Une approche Monte Carlo par Chaînes de Markov pour la classification des potentiels d'action. Application à l'étude des corrélations d'activité des cellules de Purkinje.

Matthieu Delescluse

THESE DE DOCTORAT DE L'UNIVERSITE PARIS VI

Spécialité

NEUROSCIENCES

Présentée par

Matthieu DELESCLUSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE PARIS VI

Sujet de la thèse

Une approche Monte Carlo par Chaînes de Markov pour la classification des potentiels d'action.

Application à l'étude des corrélations d'activité des cellules de Purkinje.

Soutenue le 25 Novembre 2005

devant le jury composé de :

Jean-Michel DENIAU, examinateur

Alain MARTY, directeur de thèse

Pierre MEYRAND, rapporteur

Khashayar PAKDAMAN, examinateur

Judith ROUSSEAU, rapporteur

Remerciements

Cette thèse s'est entièrement déroulée dans le laboratoire de Physiologie Cérébrale. Je tiens à remercier en tout premier lieu Christophe Pouzat qui a supervisé ce travail. Je le remercie pour la confiance qu'il m'a accordée, la ténacité dont il fait preuve en permanence et qu'il sait communiquer, l'exigence scientifique qu'il maintient toujours au plus haut. Le travail présenté ici s'inscrit dans la perspective, tracée par Christophe, d'un projet vaste et ambitieux de développement de méthodes d'analyse de données en électophysiologie. Je veux lui souhaiter ici, à lui ainsi qu'à tous ses autres collaborateurs, présents et à venir, tout le succès et la reconnaissance qu'ils méritent dans cette entreprise.

Je remercie Alain Marty qui m'a accueilli et conseillé tout au long de cette thèse. Je le remercie pour son soutien bienveillant et sans faille, la confiance qu'il m'a toujours témoignée et l'autonomie qu'il a donnée à ce projet.

Je tiens à remercier Isabel Llano pour sa direction efficace de la gestion du matériel expérimental et de l'approvisionnement courant du laboratoire, ainsi que le Gourou, Luc Tamisier, pour sa disponibilité et sa précieuse assistance d'expert Linux.

Merci à Philippe Ascher pour ses conseils et remarques toujours sages et limpides, pour son humour et sa simplicité.

Je salue avec reconnaissance Thibault Collin pour les conseils qu'il a su me donner et les nombreuses discussions à battons rompus qu'il anime toujours avec enthousiasme.

Je salue également la constante bonne humeur et l'amitié de David, Marie-Gabrielle, Sheyla, Antoine, Federico, Alex, Chiara et Brandon. Je garde dans mon souvenir les innombrables Mabillon et Vesuvio... Je leur souhaite à tous bon courage pour la poursuite de leurs thèses et post-docs respectifs.

Merci beaucoup à Catherine pour la gestion administrative des dossiers, en particulier des "missions", et à Patrick pour le confort qu'il assure quotidiennement à tout le laboratoire avec une remarquable efficacité.

J'exprime ma reconnaissance à Judith Rousseau et Pierre Meyrand qui ont accepté d'être les rapporteurs de ce travail, ainsi qu'à Jean-Michel Deniau et Khashayar Pakdaman qui ont accepté de participer à mon jury de thèse.

Enfin, s'ils n'ont pris part ni à la réalisation de ce travail, ni à la vie quotidienne du laboratoire, mes parents n'ont pas moins tenu un rôle majeur durant ces années de thèse qu'ils ont accompagnées avec séreinité, sagesse et pertinence. Je leur témoigne donc ici mon intime reconnaissance.

Il en va de même pour Ksenia, dont je partage désormais la vie. C'est à elle, que j'adresse mes ultimes remerciements. Elle a accompagné au plus près les développements quotidiens de ce travail, avec attention, justesse et perspicacité. Je veux terminer en lui disant ma reconnaissance affectueuse.

Au XXe siècle, il y aura une nation extraordinaire.
Cette nation sera grande ce qui ne l'empêchera pas d'être libre.
Elle sera illustre, riche, pensante, pacifique, cordiale.
Cette nation : elle s'appellera l'Europe.
Victor Hugo, 1849.

Le hasard n'est que la mesure de notre ignorance.
Henri Poincaré, *La science et l'hypothèse*.

Glossaire

Par souci de clarté et pour ne pas déroger aux dénominations couramment employées, j'ai choisi de conserver les acronymes anglais dans la rédaction de cette thèse. Par précaution, dans chacun des chapitres, le premier emploi d'un acronyme est accompagné de l'expression complète qu'il représente.

BIC : critère d'information bayésien (*Bayesian Information Criterion*)
CC : histogramme de corrélation croisée (*Cross-Correlogram*)
DHPG : (S)-3,5-dihydroxyphenylglycine
EM : *Expectation-Maximization*
HMM : modèle de Markov caché (*Hidden Markov Model*)
IPSC : courant post-synaptique inhibiteur *(Inhibitory Post-Synaptic Current*)
ISI : intervalle de temps entre deux potentiels d'action (*Inter-Spike Interval*)
LFP : champ local de potentiel (*Local Field Potential*)
MCMC : Monte Carlo par Chaînes de Markov (*Markov Chain Monte Carlo*)
PA : Potentiel d'Action
PCA : analyse en composantes principales (*Principal Component Analysis*)
PC : cellule de Purkinje (*Purkinje Cell*)
SD : déviation standard (*Standard Deviation*)
TTX : tétrodotoxine

Par ailleurs, pour certains mots et expressions conservés en Français dans le texte, il m'a semblé utile de préciser leurs équivalents en Anglais entre parenthèses et en italique, dans la mesure où seuls ces derniers ont pleinement acquis leur sens scientifique et sont non ambigus. Certains des équivalents anglais sont utilisés couramment, en italique, tout au long de ce manuscrit.

Table des matières

Chapitre 1

Introduction

Enregistements extracellulaires multisites

Si l'enregistrement extacellulaire est très certainement la plus ancienne technique utilisée en électrophysiologie, il n'en connaît pas moins un essor nouveau et de grande ampleur depuis quelques années, avec le développement de multiples dispositifs d'enregistrements multisites élaborés dans de nombreux laboratoires du monde entier. Ces dispositifs permettent d'enregistrer les activités de plusieurs neurones individuels simultanément, avec une excellente résolution temporelle, dans de nombreuses préparations : en culture (Gross et al., 1993, 1997), sur tranche (Oka et al., 1999 ; Egert et al., 2002 ; Yvert et al., 2004) et *in vivo* (Drake et al., 1988 ; Nicolelis et al., 1997 ; Baker et al., 1999, Csicsvari et al., 2003).

Enregistrements de populations neuronales distribuées

Il existe actuellement deux types d'approche dans le domaine des enregistrements multiples de neurones individuels. La première vise à associer, autant que faire se peut, une électrode extracellulaire à un *neurone unique*. Dans cette approche, très utilisée *in vivo*, l'expérimentateur utilise des matrices de micro-électrodes (également dénommée *Micro-Electrode Arrays* ou *MEA* en Anglais) qu'il implante dans une ou plusieurs structures cérébrales d'un animal anesthésié, ou accomplissant une tâche sensorielle, motrice ou cognitive (Nicolelis et al., 1997). L'espacement entre les extrémités des électrodes de telles matrices est typiquement de $200-400\,\mu m$. L'expérimentateur ne considère ensuite les données fournies par une électrode de la matrice que lorsque celle-ci ne "voit" qu'un seul (ou deux) neurone(s). Ceci se produit lorsque l'électrode se trouve suffisamment proche du soma d'une cellule particulière : les potentiels d'action (PAs) issus de cette cellule sont de grande amplitude sur l'électrode, ils sont donc facilement isolables par simple seuillage. Dans cette perspective, certains laboratoires ont développé des matrices dont les électrodes sont manipulables individuellement et peuvent donc être placées une à une à proximité d'une cellule (Vos et al., 1999, Baker et al., 1999). Cette approche "une électrode/un neurone", décrite dans une revue récente (Chapin, 2004), a pour objectif de comprendre comment des populations distribuées de neurones (éventuellement dans des structures cérébrales différentes) représentent l'information. Il s'agit de caractériser

la relation entre la réponse d'ensemble d'une population de neurones et un stimulus, et/ou de caractériser les relations entre les activités respectives des neurones sur de *grandes échelle spatiales*. Cette approche est par ailleurs résolument tournée vers la réalisation de prothèses neurales, d'interfaces cerveau-machine (Wessberg et al., 2000 ; Serruya et al., 2002).

Cette démarche expérimentale est mise en oeuvre également *in vitro*, sur tranche. Plusieurs laboratoires ont en effet récemment développé des matrices planes d'électrodes (*MEAs* plans) sur lesquelles peuvent être déposées des tranches de tissu (Oka et al., 1999 ; Egert et al., 2002), voire un tissu tout entier comme la rétine (Meister et al., 1994). Les sites d'enregistrement de ces matrices, de l'ordre d'une soixantaine, sont en général espacés de plus $100\,\mu m$ et permettent de couvrir une surface importante de tranche. Il est nécessaire de garder à l'esprit que la distance maximale à laquelle il est possible d'enregistrer un PA d'un neurone donné dans la plupart des situations expérimentales est de l'ordre de $50 - 80\,\mu m$. De ce fait, deux électrodes, même voisines, dans toutes les matrices évoquées ici, n'enregistrent jamais l'activité d'un même neurone.

Enregistrements de populations neuronales locales

L'approche expérimentale résumée ci-dessus ne permet d'échantillonner l'activité de populations de neurones que sur de grandes échelles spatiales (un ou deux neurones tous les $200\,\mu m$, d'après ce qui précède) : elle ne permet pas d'enregistrer simultanément plusieurs neurones proches, situés dans un petit volume de tissu. La multiplicité des électrodes et le volume qu'elles représentent causent par ailleurs de lourds dommages tissulaires. Une autre approche est nécessaire pour enregistrer *localement* une population de neurones voisins. Le développement d'électrodes extracellulaires multisites a permis de répondre à cette exigence de localisation des enregistrements (McNaughton et al., 1983 ; Drake et al., 1988 ; Wilson et McNaughton, 1993 ; Gray et al., 1995). Comme leur nom le laisse entendre, ces électrodes possèdent plusieurs sites d'enregistrements distincts, faiblement espacés (typiquement $25 - 50\,\mu m$) et disposés selon des géométries variables. La Fig. 1.1 montre un exemple d'électrode multisite : il s'agit d'une électrode de l'université de Michigan[1], du type de celles utilisées au cours de cette thèse. Lorsque les sites sont géométriquement groupés par quatre, on parle de tétrodes. Quelle que soit la disposition des sites d'enregistrement, ces électrodes multisites présentent un intérêt majeur : la faible distance entre les sites d'enregistrement et leurs dispositions relatives permettent d'enregistrer les PAs d'un même neurone sur *plusieurs sites*. L'amplitude d'un PA sur un site d'enregistrement étant fonction de la distance qui sépare ce dernier du lieu d'émission du PA, les PAs de deux neurones situés à égale distance de ce site y seront de même amplitude, ce qui rend leur distinction et leur séparation pour le moins difficiles. En revanche, les amplitudes des PAs de ces deux neurones seront nécessairement différentes sur un autre site situé dans un proche voisinage (Fig. 1.2). En fournissant plusieurs points de vue pour un même PA, l'électrode multisite facilite grandement la reconnaissance de PAs issus de neurones proches ; elle permet en ce sens de détailler les activités neuronales individuelles dans de petits volumes de tissu.

[1]Center for Neural Communication Technology

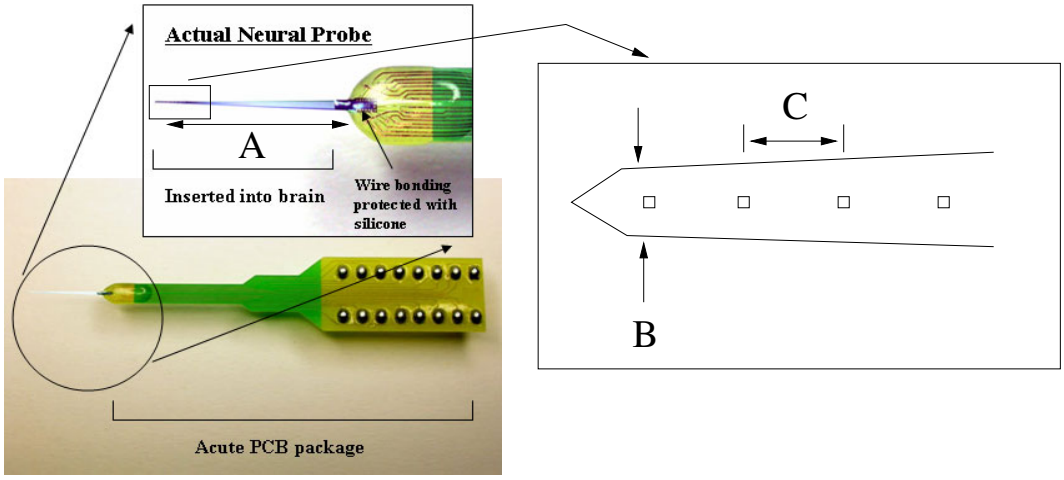

Figure 1.1: **Electrode multisite de l'université de Michigan produites par NeuroNexus Technologies** (`http://www.neuronexustech.com/Products/probe.aspx`). A: $3\,mm$. A droite, schéma de la pointe de l'électrode représentant les 4 premiers des 16 sites de l'électrode. B: $33\,\mu m$. C: $50\,\mu m$.

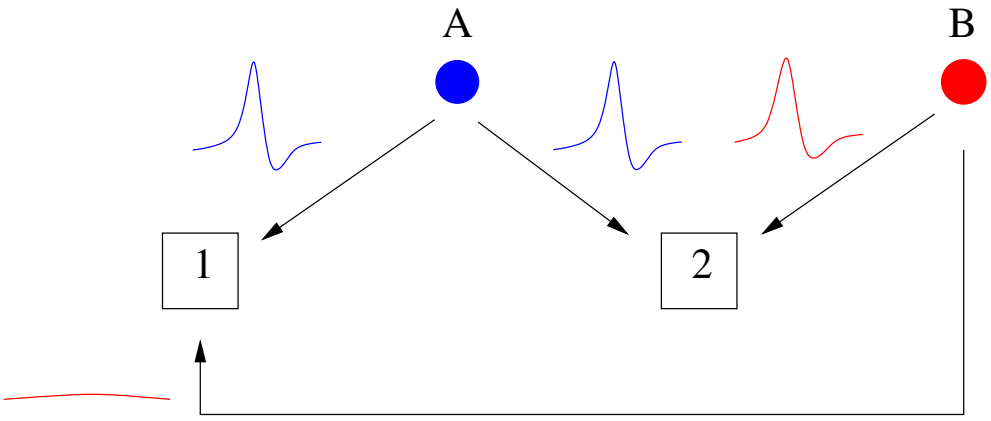

Figure 1.2: **Intérêt d'une électrode multisite pour la classification des potentiels d'action.** Les neurones A et B sont à la même distance du site 2 de l'électrode: les formes de leurs potentiels d'action sont semblables et ne peuvent être distinguées sur ce site. En revanche, sur le site 1, les formes des potentiels d'action des neurones A et B sont très différentes. La présence du site 1 permet de séparer les activités des deux neurones.

L'électrophysiologie a apporté de très nombreuses connaissances fondamentales sur la physiologie des neurones, des cellules gliales et des synapses, ainsi que sur les phénomènes biologiques dont ils sont le siège. L'un des grands enjeux des neurosciences contemporaines est de caractériser le comportement d'ensemble de *populations neuronales locales* et de le comprendre à la lumière de la somme de connaissances acquises sur les propriétés individuelles des cellules et de la transmission synaptique. Accompagnées des développements adéquats des méthodes d'analyse de ces donnnées - classification des PAs et outils mathématiques d'analyse de trains multiples de PAs (Brown et al., 2004) - ces nouvelles techniques d'enregistrement multisite de populations de neurones doivent permettre de relever ce défi (Buzsaki, 2004). C'est dans cette perspective que se situe le travail expérimental effectué au cours de cette thèse, *in vitro*, sur des groupes de *cellules de Purkinje voisines du cortex cérébelleux de rat*.

Classification des potentiels d'action

Pour être réellement exploitables, les données de ces enregistrements multiples doivent faire l'objet d'un traitement préalable visant à isoler les activités neuronales individuelles. En effet, les données brutes fournies par ces enregistrements sont, après détection des PAs, un mélange de trains de PAs issus de plusieurs neurones simultanément (Fig. 1.3 ; pour un exemple de données brutes, voir Fig. 4.2). Il faut donc commencer par déterminer le nombre de neurones présents dans l'enregistrement, et reconstituer les trains de PAs tels qu'ils sont émis par les neurones individuels. Ceci passe par le *tri*, ou *classification*, des PAs selon leur neurone d'origine : à chaque PA détecté est attribué une étiquette (un label), à savoir l'index du neurone qui l'a supposément émis. Cette procédure de classification, ou encore de labélisation, des PAs est connue en Anglais sous le terme de *spike-sorting* (*spike* étant l'équivalent raccourci anglais de "PA"). J'emploierai le terme anglais tout au long de cette thèse, au même titre que "classification des PAs". Une fois terminé, le *spike-sorting* fournit un ensemble de trains de PAs dont les corrélations temporelles peuvent faire l'objet d'analyses ultérieures.

De très nombreuses méthodes de *spike-sorting* ont été proposées et sont utilisées par les laboratoires effectuant des enregistrements multiples. Beaucoup sont des solutions partielles ou *ad hoc* au problème. Aucune n'est entièrement satisfaisante et aucune ne s'est réellement imposée à l'heure actuelle. Quel que soit leur niveau d'automaticité, autrement dit quel que soit le degré d'intervention de l'expérimentateur dans la procédure, ces méthodes fondent toutes leur classification sur la *forme des PAs* (leur décours temporel) uniquement : l'attribution d'un neurone d'origine à un PA se fait sur la seule base de la forme de celui-ci. L'expérimentateur fait donc l'hypothèse fondamentale de *stationnarité* des PAs émis par un neurone donné. Dans cette hypothèse, seul le bruit est à l'origine de la variabilité des PAs d'un neurone. Cette stabilité des PAs d'un neurone est souvent observée expérimentalement et il est légitime de s'appuyer sur cette reproductibilité pour identifier le neurone d'origine d'un PA. Cependant, il existe de nombreuses exceptions à cette règle générale. En particulier, un neurone émet plusieurs PAs de *formes différentes* au cours de salves de PAs (*bursts*). C'est le cas, par exemple, des cellules de Purkinje du cervelet que nous avons enregistrées : ces cellules

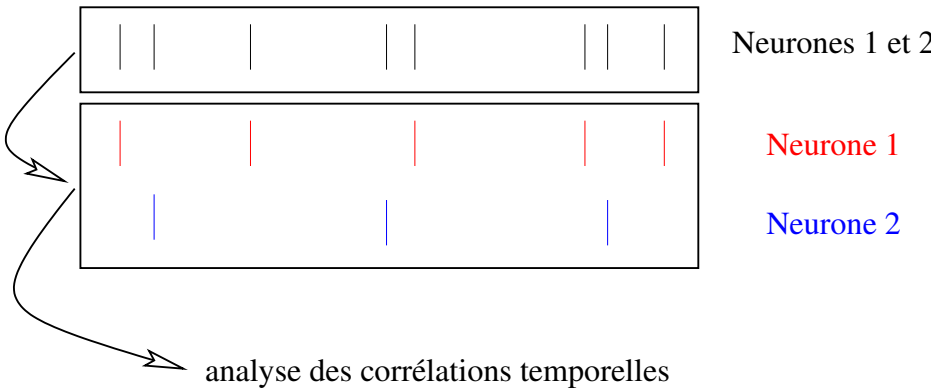

Figure 1.3: **La classification des potentiels d'action (*spike-sorting*).** Les données recueillies sont un mélange de trains de potentiels d'action (barres verticales) issus de plusieurs neurones disctincts. Le *spike-sorting* vise à séparer chacun des trains. Une fois cette séparation accomplie, les trains de potentiels d'action peuvent faire l'objet d'une analyse de leurs corrélations temporelles. Dans ce schéma, ainsi que dans tous les schémas de trains de potentiels d'action qui suivent, les barres verticales successives symbolisent les potentiels d'actions émis au court du temps.

sont capables d'émettre des *bursts* de PAs d'amplitudes décroissantes (un exemple de *burst* est donné dans la Fig. 2.4). Lorsqu'a débuté ce travail, aucune des méthodes de *spike-sorting* proposées ne permettait de traiter automatiquement de telles données.

Le cortex cérébelleux

Les cellules de Purkinje (PCs) sont les cellules principales du cortex cérébelleux. Ce sont des neurones gabaergiques inhibiteurs dont les axones en constituent la seule sortie. Dans un plan sagittal, leurs corps cellulaires forment une monocouche qui porte leur nom (*la couche des cellules de Purkinje*). Cette couche unicellulaire sépare deux autres couches caractérisées par la présence de types neuronaux distincts : la *couche des grains* et la *couche moléculaire* (Ramon y Cajal, 1911) (Fig. 1.4). La première, interne à la couche des PCs, contient les corps cellulaires des cellules en grain, excitatrices, ceux des cellules de Golgi et des cellules de Lugaro, inhibitrices, ainsi que les cellules dites en Anglais "unipolar brush" dans la partie vestibulaire du cervelet. La seconde couche, externe à celle des PCs, contient les corps cellulaires des interneurones inhibiteurs, dont on peut distinguer deux types : les cellules dites en panier, situées dans la partie la plus interne de cette couche, proche des PCs, et les cellules étoilées, plus périphériques.

Le cortex cérébelleux reçoit deux types d'afférences, toutes deux excitatrices : les *fibres grimpantes*, issues de l'olive inférieure (dans le tronc cérébral) et les *fibres moussues* dont les origines sont diverses au sein du système nerveux central. Les premières forment de multiples synapses sur l'arbre dendritique des PCs, en particulier dans sa partie proximale. Chaque PC n'est contactée que par une seule fibre grimpante, mais l'axone d'un neurone donné de l'olive inférieure se scinde en différentes fibres grimpantes. Les fibres moussues se terminent quant à elles dans la couche des grains, au

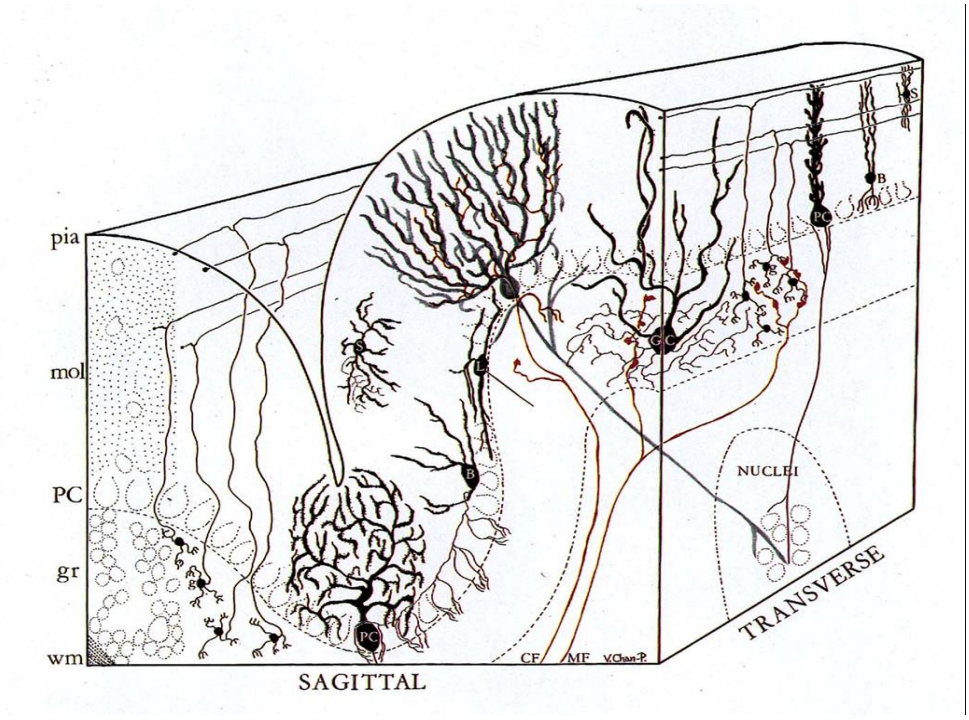

Figure 1.4: **Coupe schématique du cervelet selon les plans sagittal et frontal** (transverse)(Eccles et al., 1967). *pia*: pie. *mol*: couche moléculaire. *PC*: cellules de Purkinje. *gr*: couche des cellules en grains. *wm*: matière blanche. *CF*: fibre grimpante. *MF*: fibre moussue. *L*: cellule de Lugaro. *B*: cellule en panier. *S*: cellule étoilée. *GC*: cellule de Golgi. *g*: cellule en grain.

niveau de glomérules où elles forment des synapses avec les cellules en grains. Les cellules de Golgi contactent, et inhibent, les cellules en grains dans ces mêmes glomérules. Les axones des cellules en grain montent jusqu'à la couche moléculaire où, après une bifurcation en T, ils sont connus sous le nom de "fibres parallèles" et forment des synapses excitatrices sur les arbres dendritiques des multiples PCs qu'elles traversent. Les fibres parallèles contactent également les interneurones de la couche moléculaire. Ces interneurones exercent leur action inhibitrice entre eux et sur les PCs, un interneurone pouvant notamment contacter plusieurs PCs. Ces dernières ont une action inhibitrice sur les neurones des noyaux cérébelleux profonds. Une revue détaillée de l'organisation du cortex cérébelleux est disponible en Français (Pouzat, 1998).

Objectifs de la thèse

Objectif expérimental

Un grand nombre de travaux, *in vitro* en particulier, ont permis de caractériser de nombreuses propriétés individuelles des neurones du cortex cérébelleux et de leurs synapses. Mais comprendre ce réseau aux composantes bien connues nécessite également d'étudier son activité au niveau d'un groupe de cellules.

L'objectif expérimental de ce travail est de mettre en évidence et de caractériser *l'influence de l'activité des interneurones de la couche moléculaire sur le patron d'activité d'un ensemble de PCs enregistrées simultanément*, dans les tranches de cervelet de rat. Il s'agit ici de savoir si l'inhibition commune de PCs contactées par un même interneurone est visible dans l'analyse des corrélations de leurs trains de PAs et si, en particulier, les interneurones sont capables de synchroniser des PAs de PCs voisines dans les tranches.

Ces enregistrements ont été réalisés dans différentes conditions pharmacologiques, à l'aide d'électrodes multisites permettant de recueillir l'activité spontanée de plusieurs PCs voisines. Le réseau des PCs présente plusieurs atouts. Tout d'abord, ce sont les cellules principales du circuit décrit plus haut ; elles en constituent l'unique sortie, de sorte que l'on peut considérer que leur activité résume le traitement de l'information réalisé dans le cortex cérébelleux. D'autre part, leurs signaux extracellulaires sont de grande amplitude ; on peut les enregistrer relativement loin de leur soma. Leur arrangement géométrique linéaire particulier en monocouche est par ailleurs parfaitement adapté à certaines de nos électrodes multisites, dont les sites d'enregistrement sont disposés linéairement. Enfin, l'action inhibitrice des interneurones de la couche moléculaire sur les PCs a fait l'objet, dans ce laboratoire, de nombreuses études dont profite le présent travail.

Une présentation condensée des résultats obtenus nous permet d'illustrer en outre l'idée de "recherche reproductible" (*reproducible research*) développée dans d'autres domaines scientifiques depuis quelques années. Au coeur de cette démarche scientifique se trouve la volonté de publier les analyses sous une forme qui leur permette d'être littéralement re-produites par tout un chacun sur sa propre machine.

Objectif méthodologique

L'absence de méthode automatique de *spike-sorting* satisfaisante, ainsi que la non-stationnarité de la forme des PAs émis par les PCs, ont requis le développement d'une nouvelle approche de ce problème. Le second objectif de mon travail est donc méthodologique : il est de contribuer à l'élaboration et au *développement d'une méthode automatique de spike-sorting* capable de traiter la non-stationnarité des PAs émis par un neurone. Il est aussi d'en réaliser la validation expérimentale. La méthode proposée, radicalement nouvelle dans ce domaine, implémente un algorithme dit de *Monte Carlo par Chaînes de Markov* (MCMC) qui permet de réaliser une inférence statistique sur un modèle de génération de données plus réaliste et plus complexe que ceux habituellement considérés. Y est inclus, en particulier, un modèle de statistique de décharge des neurones enregistrés qui permet de prendre en compte automatiquement, pour la première fois, l'information temporelle des trains de PAs pour réaliser la classification. Cette prise en compte permet de résoudre le problème de la non-stationnarité de la forme des PAs des PCs. Ces développements méthodologiques ont été initiés et supervisés par Christophe Pouzat. Leur validation expérimentale a été effectuée sur le système des PCs décrit plus haut.

Plan de la thèse

L'ensemble de ce travail de thèse s'articule donc autour de ces deux volets, expérimental et méthodologique. Le chapitre 2 décrit l'activité spontanée des PCs et expose les aspects saillants de l'action inhibitrice des interneurones sur ces cellules ; cette inhibition est susceptible de modeler les décharges spontanées de PCs. Ce chapitre passe également en revue l'ensemble des travaux réalisés jusqu'à présent sur la synchronisation et les corrélations d'activités spontanées des PCs. Le chapitre 3 présente les méthodes de *spike-sorting* les plus répandues parmi celles disponibles quand a débuté ce travail. Dans le chapitre 4 sont décrites les principales méthodes utilisées et *élaborées* au cours de cette thèse. La première partie de ce chapitre donne les détails techniques des expériences réalisées ; la seconde expose les méthodes de *spike-sorting* utilisées, et en particulier celle que nous avons développée ; la troisième partie de ce chapitre décrit les méthodes utilisées pour analyser les corrélations entre trains de PAs. Le chapitre 5 présente, par le biais des articles, les résultats obtenus concernant les développements méthodologiques réalisés et leur validation expérimentale (trois premiers articles), et concernant la biologie du cervelet et l'idée de recherche reproductible (quatrième article). Enfin, le chapitre 6 discute et met en perspective l'ensemble de ce travail.

Chapitre 2

Les cellules de Purkinje

Les cellules de Purkinje (PCs, Fig. 2.1) sont le seul lien entre le cortex cérébelleux et les noyaux profonds du cervelet. Les patrons de décharge des PCs inhibant les neurones de ces noyaux, ainsi que les patrons de décharge de ces derniers, contribuent largement à la fonction de coordination motrice du cervelet. L'activité des PCs est le fruit d'un jeu subtil entre leurs propriétés membranaires intrinsèques et l'activité synaptique intense dont leur gigantesque arbre dendritique est l'objet. La *panoplie de courants somatiques et dendritiques* dont les PCs sont pourvues les rend capables de *produire spontanément des potentiels d'action* (PAs), même en l'absence de toute activité synaptique (section 2.1). Néanmoins, les *entrées synaptiques inhibitrices* que les PCs reçoivent ont une influence notable sur la forme de cette activité spontanée et sont potentiellement capables d'induire des corrélations entre décharges de cellules voisines (section 2.2). La *synchronisation* des décharges spontanées de PCs a fait l'objet de travaux dans plusieurs préparations *in vivo*, mais jamais *in vitro* (section 2.3).

2.1 La décharge des cellules de Purkinje

La partie expérimentale de ce travail de thèse est fondée sur l'*activité spontanée des PCs* dans les tranches de cervelet de rat. Après avoir rappelé brièvement les deux types de PAs que peuvent produire les PCs (paragraphe 2.1.1), cette section décrit cette activité spontanée dans les différentes préparations dans lesquelles elle a été étudiée. Les travaux passés en revue montrent comment des *somas* de PCs fraîchement dissociées (ou isolées) peuvent produire une activité spontanée (paragraphe 2.1.2) comprenant également des salves de PAs de très haute fréquence (*bursts*) (paragraphe 2.1.3). Le paragraphe 2.1.4 s'intéresse à cette activité spontanée dans les PCs intactes des tranches, et se focalise sur le rôle et l'influence des *dendrites* dans le façonnement de cette activité spontanée. Enfin, le paragraphe 2.1.5 résume très brièvement les éléments de cette activité spontanée qui n'apparaissent qu' *in vivo*.

2.1.1 Les potentiels d'action de la cellule de Purkinje

Sans entrer dans une description détaillée de l'électrophysiologie des PCs (revue par Llinas et Sugimori, 1992), il est nécessaire d'en donner schématiquement les éléments

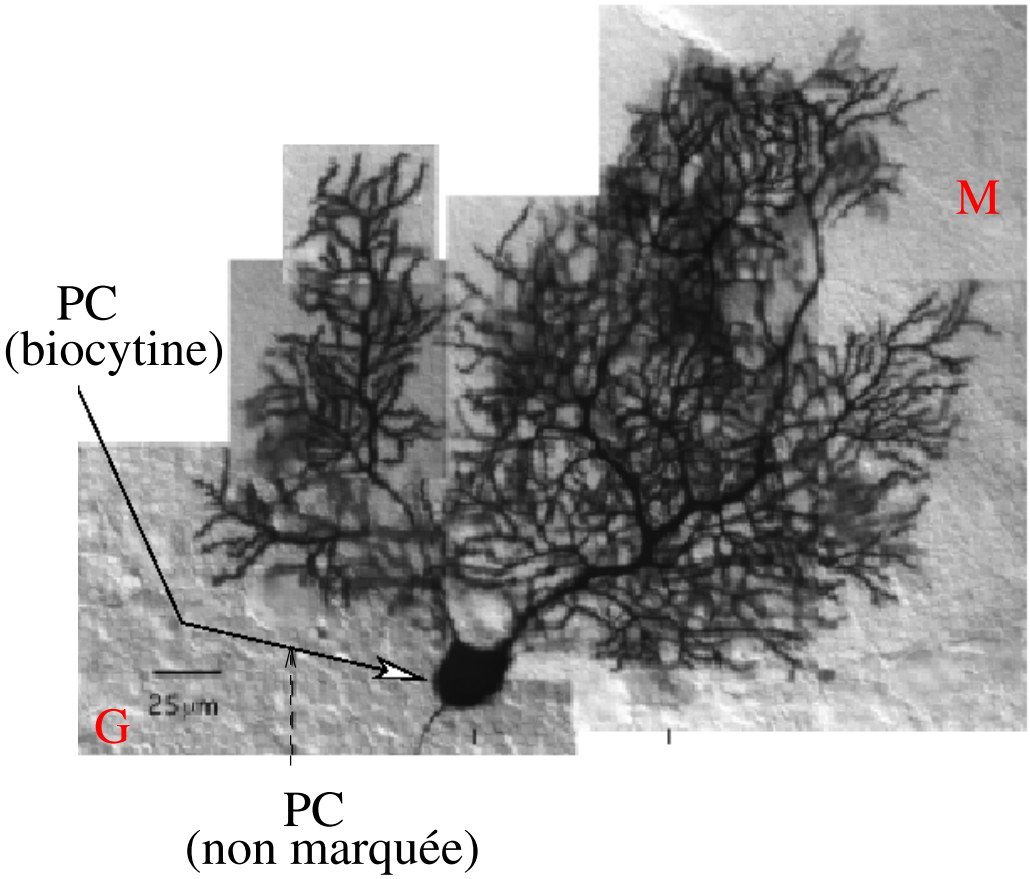

Figure 2.1: **Cellule de Purkinje d'un rat de 32 jours, remplie de biocytine** (Pouzat, 1998). PC: cellule de Purkinje. M: couche moléculaire. G: couche des grains.

Figure 2.2: **Les potentiels d'action de la cellule de Purkinje:** potentiels d'action simples (flèche et suivants) et potentiel d'action dendritique calcique (encadré) enregistrés extracellulairement. Activité spontanée en présence de DHPG ($40\,\mu M$). Barres d'échelles horizontale et verticale: $100\,ms$ et $0.5\,mV$ respectivement

principaux. Les PCs peuvent produire deux grands types de PAs (Fig. 2.2) : des PAs sodiques classiques, dits "simples" (*simple spikes*), au niveau du soma et de l'axone, et des PAs calciques au niveau des dendrites, que l'on rencontre moins fréquemment dans d'autres types cellulaires (Llinas et Sugimori, 1980,a,b).

Par ailleurs, la réponse remarquable de la PC à une excitation de sa fibre grimpante est caractéristique de cette cellule ; cette réponse, véritable signature électrophysiologique, permet une identification certaine de la PC lors des enregistrements *in vivo*. L'entrée excitatrice de la fibre grimpante provoque une dépolarisation massive des dendrites de la PC. Cette dépolarisation est surmontée d'un PA simple initial, suivi de plusieurs petits PAs, calciques dans les dendrites, sodiques au niveau du soma et de l'axone. Cette succession rapprochée de plusieurs PAs en réponse à l'excitation d'une fibre grimpante est connue sous le nom de PA complexe (*complex spike*). La durée d'un PA complexe est de l'ordre de $10-15\,ms$. Dans les tranches de cervelet, les fibres grimpantes ne sont actives que si on les stimule. Par conséquent, l'activité spontanée d'une PC dans cette préparation ne comporte pas de PA complexe. Les travaux consacrés à la décharge de PAs complexes (*in vivo*) ne seront donc abordés que de façon marginale, à la fin de ce chapitre.

2.1.2 La décharge spontanée tonique

Les PCs sont capables d'émettre spontanément des PAs à haute fréquence et de manière relativement régulière *in vivo* (Bell et Grimm, 1969 ; Latham et Paul, 1971 ; Jäger et Bower, 1994). Cette capacité de décharge spontanée tonique est conservée *in vitro*, dans les tranches (Crépel, 1972 ; Hounsgaard, 1979 ; Llinas et Sugimori, 1980b), même lorsque toute activité synaptique est bloquée (Häusser et Clark, 1997 ; Womack et Khodakhah, 2002, 2004), ainsi que dans les cultures de PCs (Gruol et Franklin, 1987). Plus encore, cette activité spontanée a été constatée dans des cultures de PCs *avant la formation des dendrites* (Gruol et al., 1991), ainsi que sur des somas de PCs fraîchement isolées (Nam et Hockberger, 1997 ; Raman et Bean, 1997 ; Swensen et Bean, 2003). Dans ce dernier cas, l'activité spontanée des PCs est très similaire à celle des PCs intactes des tranches, en l'absence d'entrées synaptiques (Raman et Bean, 1999). Ces travaux

suggèrent que cette capacité des PCs d'émettre spontanément des PAs vient de leurs propriétés membranaires intrinsèques et que leurs dendrites ne sont pas indispensables à la génération de cette activité spontanée, même si elles jouent incontestablement un rôle important dans la forme finale qu'elle peut prendre (voir section 2.1.4).

Pour fixer les idées, il peut être utile de donner les fréquences typiques des décharges toniques spontanées des PCs dans les différentes préparations, dans les conditions contrôle. *In vivo*, chez le chat anesthésié, ces fréquences vont de 30 à plus de $50\,Hz$ (Bell et Grimm, 1969 ; Latham et Paul, 1971). Dans les tranches de cervelet de rat, elles sont environ de $40\,Hz$ à $34^{\circ}C$ (Häusser et Clark, 1997) et $10\,Hz$ à $20^{\circ}C$ (observations personnelles) ; elles sont de $30\,Hz$ dans les PCs de souris fraîchement dissociées à $22^{\circ}C$ (Khaliq et al., 2003). La différence de fréquence notable observée à la même température entre les tranches et les PCs dissociées vient de l'activité inhibitrice des interneurones présente dans les tranches uniquement.

L'un des courants les plus remarquables de l'activité de décharge des PCs est sans doute le courant sodium dit "résurgent" découvert par Raman et Bean (1997) dans des PCs fraîchement dissociées (Fig.2.3). Ce courant est sensible à la tétrodotoxine (TTX), molécule bloquant spécifiquement les canaux sodium sensibles au voltage. Après une dépolarisation de la membrane de la PC ayant engendré le courant sodium transitoire typique des PAs, la repolarisation vers des potentiels encore dépolarisés situés entre -60 et $-20\,mV$ déclenche un courant sodium (Fig. 2.3B), inattendu pour les canaux conventionnels (Fig. 2.3C). En particulier, la repolarisation de chaque PA est capable d'activer ce courant résurgent. Ce dernier assure donc une nouvelle dépolarisation progressive immédiatement après un PA.

L'amplitude de ce courant sous le seuil est beaucoup plus petite que celle d'un transitoire sodique de PA, mais sa cinétique est beaucoup plus lente. Par ailleurs, le temps de récupération de l'inactivation est remarquablement court pour ces canaux. Il s'agit là de leur autre propriété capitale pour la décharge à haute fréquence : ils sont très vite à nouveau disponibles pour le PA suivant. Les auteurs cités ci-dessus montrent que les canaux sodium responsables de ce courant résurgent génèrent également des transitoires sodiques de PA. Il semble néanmoins que tous les canaux sodium de la PC ne soient pas en mesure de le produire. Les mécanismes moléculaires permettant l'existence de ce courant inhabituel commencent à être compris. La présence de certaines sous-unités α, telles que la sous-unité $Na_V1.6$, apparaît nécessaire (Raman et al., 1997). Celles-ci intéragissent avec une protéine qui bloque le canal dans son état ouvert lors de la dépolarisation mais se détache lors de la repolarisation. D'après Grieco et al. (2005) cette protéine pourrait être la queue cytoplasmique de la sous-unité β_4 des canaux sodium.

Ce courant particulier joue un rôle crucial dans l'émission spontanée de PAs par les PCs fraîchement isolées et leur capacité à décharger à haute fréquence (Khaliq et al., 2003). Son absence rend l'activité spontanée des PCs beaucoup plus rare et réduit drastiquement la fréquence de décharge de leur activité évoquée. Contrairement à d'autres neurones *pacemaker*, le courant cationique activé par hyperpolarisation, I_h, bien que présent dans les PCs (Crépel et Penit-Soria, 1986), est très faible entre deux PAs et ne joue pratiquement aucun rôle dans l'activité spontanée (Raman et Bean, 1999).

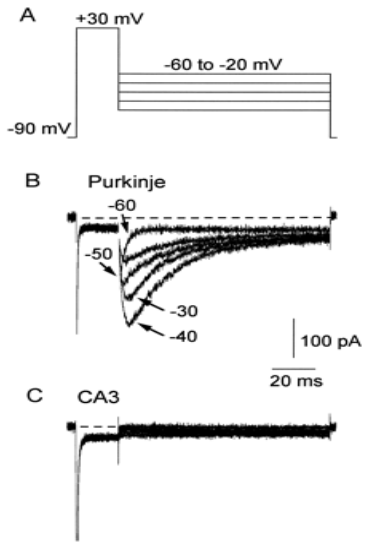

Figure 2.3: **Courant sodium résurgent** (Raman et Bean, 1997). A, un courant sodium est évoqué par une dépolarisation de -90 à $+30\,mV$ d'une durée de $20\,ms$. La membrane est ensuite repolarisée jusqu'à des voltages situés entre -20 et $-60\,mV$. B, courant sodium sensible à la TTX apparaissant dans une cellule de Purkinje lors de ce protocole. Le courant transitoire à $30\,mV$ n'est pas à l'échelle (pic de $2\,nA$). Les courants de fuite et capacitifs dans $300\,nM$ de TTX ont été soustraits. C, courant sodium sensible à la TTX apparaissant dans une cellule pyramidale de l'hippocampe (région CA3) lors de ce même protocole. Le pic du courant à $30\,mV$ est de $1.3\,nA$ (hors échelle). Les barres d'échelles sont valables pour les deux ensembles de traces.

Ce courant sodium résurgent n'explique néanmoins pas à lui seul la génération spontanée et tonique de PAs à haute fréquence. Il est tout d'abord associé à un autre courant sodium sous le seuil : le courant sodium dit "persistant", lui aussi sensible à la TTX et d'inactivation lente (Llinas et Sugimori, 1980a,b) participe également à la dépolarisation progressive entre deux PAs. Raman et Bean (1999) ont étudié les autres courants importants de cette activité dans des somas de PCs fraîchemement isolées. En particulier, les conductances potassium activées par le calcium, BK et SK (Gähwiler et Llano, 1989), ainsi que celles dépendant du potentiel, sont massives et ont une déactivation rapide. La repolarisation rapide qu'elles induisent ne ramène pas la membrane à des potentiels très hyperpolarisés, et l'hyperpolarisation au terme du PA (*afterhyperpolarisation*) est peu prononcée. Le calcium nécessaire à l'activation des conductances potassium dépendant du calcium entre majoritairement par les canaux calciques de type P dépendant du potentiel. Les courants des canaux calciques de type T participent, comme le courant sodium résurgent, à la dépolarisation entre deux PAs, mais de manière secondaire.

2.1.3 L'émission spontanée de *bursts*

Les PCs ne sont pas seulement capables de décharger spontanément des PAs individuels de façon tonique, elles sont aussi capables d'émettre spontanément des salves de PAs (*bursts*) de très haute fréquence (de 100 jusqu'à plus de 200 Hz). C'est l'autre grande caractéristique de leur activité (Fig. 2.4). Là encore, ces *bursts* peuvent être émis de façon spontanée *in vivo* (Jäger et Bower, 1994), dans les tranches (Womack et Khodakhah, 2002) et par des somas de PCs fraîchement dissociées (Swensen et Bean, 2003). Dans cette dernière préparation, des *bursts* peuvent être émis également en réponse à de brèves injections de courant dépolarisant (Raman et Bean, 1997). On peut donc ici aussi expliquer leur génération par les propriétés membranaires intrinsèques du soma.

Les *bursts* spontanés des PCs fraîchement dissociées sont brefs et ne contiennent pas plus de deux ou trois PAs. Swensen et Bean (2003) ont disséqué les courants qui en sont responsables. Les résultats sont similaires à ceux de l'émission tonique de PAs dans la même préparation, décrite dans le paragraphe précédent. Ici aussi, le rôle des courants sodium sous le seuil sensibles à la TTX, et en particulier celui du courant sodium résurgent, est majeur. Ils sont responsables, avec le courant calcium des canaux de type T et dans des proportions comparables, de la dépolarisation entre deux PAs qui porte le potentiel de membrane vers le seuil de déclenchement du potentiel d'action. Le courant I_h ne joue pratiquement aucun rôle dans cette dépolarisation et la génération de *bursts* de très haute fréquence n'est pas perturbée par son absence. Le courant calcium des canaux de type P est grand immédiatement après le PA mais décroît lentement (il est bien inférieur au courant des canaux de type T au milieu de l'intervalle entre PAs). Le calcium qui entre alors active les canaux potassium de grande conductance dépendant du calcium, BK, qui repolarisent rapidement la membrane et sont à l'origine de la brièveté des PAs. Les canaux potassium dépendant du potentiel participent également à la repolarisation. La déactivation rapide des canaux potassium responsable de la repolarisation empêche ceux-ci de ramener la membrane à des potentiels très hyperpolarisés et on n'observe qu'une hyperpolarisation limitée après le PA. C'est une propriété cru-

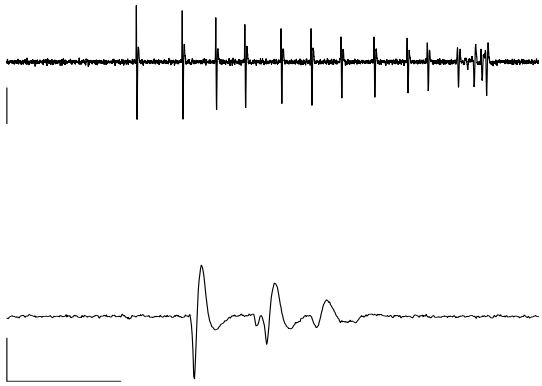

Figure 2.4: **Exemples de *bursts* de potentiels d'action d'amplitudes décroissantes.** *Haut*: *burst* long. Barres d'échelle horizontale et verticale: $100\,ms$ et $0.5\,mV$ respectivement. *Bas*: triplet de potentiels d'action. Barres d'échelle horizontale et verticale: $10\,ms$ et $0.5\,mV$ respectivement.

ciale des canaux potassium pour la génération de *bursts* à haute fréquence. Toujours selon Swensen et Bean (2003), les canaux potassium de petite conductance dépendant du calcium, SK, jouent également un rôle au cours de ces *bursts*, en particulier dans le contrôle de leur durée : c'est l'augmentation cumulative de ce courant au cours du *burst*, accompagnée d'une inactivation cumulative des canaux sodium, qui met un terme à celui-ci.

Selon ces mêmes auteurs, l'ensemble de ces courants sodium, calcium et potassium sont présents simultanément et se compensent partiellement entre deux PAs au sein d'un *burst*. Leur résultante nette est un courant entrant d'amplitude bien inférieure à chacun de ces courants individuels. Ceci rend possibles des mécanismes de compensation : un changement important de l'une des conductances peut ne pas remettre en cause la capacité de la PC à émettre des *bursts*, s'il est accompagné de changements adéquats dans d'autres conductances. Et de fait, on observe qu'une réduction substantielle des conductances sodium (en présence de TTX ou par mutation de certains canaux sodium) n'empêche pas la formation des *bursts*, parce que celle-ci est compensée soit par une réduction similaire des conductances potassium, soit par une augmentation sensible des courants calcium (Swensen et Bean, 2005). Ces auteurs mettent donc en évidence une grande flexibilité des modes d'émission de *bursts* par la PC. Cette robustesse suggère que ce type particulier de décharge est important pour la fonction physiologique de la PC.

On peut s'interroger sur ce qui détermine le type d'émission de PAs dans la PC : émission tonique de PAs individuels ou émission de *bursts*. Le courant sodium résurgent semble être de première importance dans les deux cas. D'après Swensen et Bean (2003) la disponibilité des courants calcium de type T est sans doute un facteur crucial. La

pente de la courbe d'inactivation de ce courant est forte à des potentiels de l'ordre de $-70\,mV$ et le niveau d'hyperpolarisation atteint entre deux PAs détermine le niveau d'inactivation de ce courant. Les courants SK également sont vraisemblablement capitaux à ce niveau : lors d'une décharge de type tonique, le niveau de calcium peut demeurer élevé dans la cellule, ce qui augmente le courant SK et favorise l'émission de PAs individuels.

Terminons en précisant que l'une des grandes caractéristiques des PAs de ces *bursts* est *la forte décroissance de leurs amplitudes* : l'amplitude du premier PA est égale à celle d'un PA individuel d'une décharge tonique, mais les suivants voient leurs amplitudes drastiquement réduite (elles peuvent être divisées par deux à chaque nouveau PA dans le *burst*, Fig. 2.4). L'ampleur de cette décroissance croît avec la fréquence interne du *burst*. Elle est due à une inactivation cumulative des canaux des transitoires sodiques des PAs.

2.1.4 Le rôle des dendrites dans la décharge spontanée

Bien que l'activité tonique spontanée et la génération de *bursts* soient possibles en l'absence des dendrites de la PC, il est évident que celles-ci, ainsi que l'activité synaptique qu'elles reçoivent (voir section 2.2), exercent une influence importante sur le patron final de cette activité. Les dendrites de la PC possèdent notamment une forte densité de canaux calcium dépendant du potentiel (Usowicz et al., 1992) qui peuvent donner naissance à de massifs PAs calciques (Llinas et al., 1969 ; Llinas et Sugimori, 1980b). Comme le soma, elles possèdent également des conductances potassium BK et SK activées par le calcium (Gruol et al., 1991 ; Cingolani et al, 1992 ; Edgerton et Reinhart, 2003), ainsi que des conductances potassium dépendant du potentiel (Martina et al., 2003).

L'activité spontanée des PCs intactes a été étudiée avec soin dans les tranches de souris (Womack et Khodakhah, 2002). Deux types de décharge spontanée ont été mis en évidence : une décharge tonique relativement régulière (à environ $50\,Hz$ à $35^{\circ}C$) et une décharge "trimodale" reproduisant plusieurs cycles de trois périodes successives : une période de décharge tonique, une période de décharge en *bursts* réguliers et une période de silence (Fig.2.5). Ce second mode de décharge se généralise au cours du développement et est observé dans la majorité des PCs de souris âgées de plus de 16 jours. D'autre part, le blocage de toute transmission synaptique rapide favorise ce mode trimodal de décharge : il fait passer certaines PCs du mode tonique au mode trimodal, l'inverse n'étant pas observé. Enfin, la baisse de la température réduit linéairement la fréquence de décharge des cellules en mode tonique et rend silencieuses les cellules en mode trimodal. Ce dernier est donc favorisé par trois facteurs : la maturation de la PC, et en particulier le développement de son arbre dendritique, le blocage de toute transmission synaptique rapide et la température physiologique. Par ailleurs, les cycles de la décharge trimodale sont de durées variables d'une cellule à l'autre et aucune synchronisation des cycles n'a été constatée dans des PCs voisines. De plus, une PC en mode trimodal peut être voisine d'une PC en mode tonique. L'hypothèse que le mode trimodal de décharge est contrôlé par la diffusion cyclique de neuromodulateurs est donc très peu probable. Il s'agit bien plutôt d'un cycle renouvelé par les *propriétés*

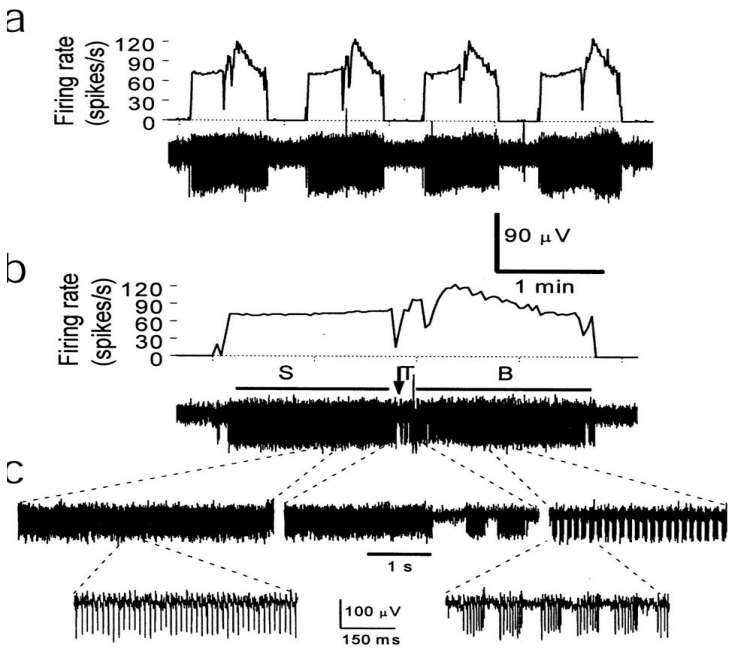

Figure 2.5: **Décharge spontanée d'une cellule de Purkinje dans une tranche de cervelet de souris à** $35^{\circ}C$ (Womack et Khodakhah, 2002). La cellule parcourt un cycle trimodal de décharge: modes tonique, en *bursts*, et silencieux. *a*, quatre cycles du patron de décharge. La fréquence moyenne de décharge est tracée au-dessus de la trace brute. Chaque cycle se termine par une période de silence. *b*, enregistrement de la même cellule à une échelle de temps étendue. Le cycle est composé d'une décharge tonique (S), d'une décharge en bursts (B), séparées par une période de transition (T). *c*, les périodes S, T et B décrites en *b* sont détaillées sur une échelle de temps étendue.

intrinsèques de cette cellule.

Womack et Khodakhah (2002) ont montré que les dendrites jouent un rôle majeur dans l'établissement de la décharge trimodale des PCs, en particulier dans le contrôle de la partie en *bursts* du cycle (Womack et Khodakhah, 2004). Le nombre de PAs présents dans un *burst* est généralement compris entre 3 et 20 ; il est donc à peu près trois à quatre fois plus élevé que dans les PCs fraîchement dissociées. En raison d'une dépolarisation progressive et continue, la fréquence de décharge augmente au cours du *burst* et l'amplitude des derniers PAs est dramatiquement réduite. Mais l'observation la plus remarquable faite par Womack et Khodakhah (2004) est qu'un *burst est systématiquement terminé par un PA calcique dendritique dépendant des canaux calciques* P/Q. Les *bursts* des PCs intactes des tranches, même en l'absence de toute transmission synaptique rapide, sont donc relativement différents de ceux des PCs fraîchement dissociées décrits dans la section précédente (Swensen et Bean, 2003, 2005).

Comme dans cette dernière préparation, les canaux calcium de type T et les canaux potassium SK et BK dépendant du calcium contribuent à la formation des *bursts* des PCs dans les tranches, mais aucun d'eux n'est indispensable : la suppression des courants qu'ils supportent modifie le déroulement des *bursts* et leur enchaînement (intervalles

entre PAs dans le *burst*, intervalles entre *bursts*, durée d'un *burst*, nombre de PAs dans un *burst*), mais n'abolit pas leur génération régulière. Le courant I_h quant à lui n'y contribue pas. En fait, seuls les canaux calcium de type P/Q des dendrites sont indispensables pour soutenir l'activité de *bursts*. De plus, le cycle d'émission des PAs calciques dendritiques réguliers dus à ces canaux demeure même lorsque les courants sodium sensibles à la TTX sont bloqués. Lorsque les canaux calcium P/Q des dendrites sont bloqués, les PCs passent de leur décharge en *bursts* réguliers à une décharge de type tonique ou à une décharge en *bursts* irréguliers, beaucoup plus courts, sans PA calcique à leur terme, et très semblables à ceux que les PCs fraîchement dissociées émettent (Swensen et Bean, 2003). La capacité des PCs à émettre des PAs avec leurs seules conductances somatiques n'est donc pas remise en cause. Cependant, ce sont bien les conductances calciques dendritiques qui contrôlent la forme de cette activité de *bursts*, en particulier sa régularité et sa robustesse.

L'intervalle entre *bursts* est de l'ordre de $40\,ms$. Dans la mesure où le blocage des courants potassium dépendant du calcium n'en supprime pas la présence, il semble que la fin d'un *burst* soit due à l'inactivation des canaux sodium par le PA calcique dendritique plutôt qu'aux conductances potassium BK et SK, activées par le calcium entré au cours de ce PA calcique. Les effets de ces conductances sur l'activité intrinsèque des PCs intactes ont été étudiés dans les tranches et le détail par Edgerton et Reinhart (2003).

La description de ce mode particulier de décharge spontané trimodal des PCs est compatible avec les descriptions qui en avaient été faites antérieurement. Des enregistrements intracellulaires de PCs dans des tranches de cochons d'Inde adultes avaient déjà révélé un patron de décharge cyclique alternant périodes d'activité et périodes quiescentes (Llinas et Sugimori, 1980a,b ; Jäger et Bower, 1994). Ce patron cyclique a été observé sur cette préparation avec des enregistrements extracellulaires également (Llinas et Sugimori, 1980a,b) ; il a aussi été suggéré dans les tranches de cervelet de rat (Jäger et Bower, 1999). Womack et Khodakhah (2002) précisent ce résultat en démontrant que le patron cyclique de décharge alternant activité et quiescence demeure en l'absence de transmission synaptique rapide, et que la période d'activité consiste en la succession d'une décharge tonique et d'une décharge en *bursts*.

2.1.5 L'activité des cellules de Purkinje *in vivo*

La décharge spontanée en *bursts* n'est pas observée *in vivo*, mais l'alternance entre périodes de décharge de PAs simples à haute fréquence et périodes de quiescence est toujours de mise (Granit et Philips, 1956 ; Bell et Grimm, 1969), de même que la capacité des PCs à émettre des PAs simples intrinsèquement, en l'absence de toute entrée synaptique excitatrice des fibres parallèles (Cerminara et Rawson, 2004). Comme signalé au début de ce chapitre, un élément nouveau apparaît dans l'activité spontanée des PCs *in vivo* : les PAs complexes émis en réponse à l'excitation par les fibres grimpantes issues de l'olive inférieure (Bell et Grimm, 1969). Sans stimulation particulière, ces PAs complexes apparaissent à une fréquence moyenne de $1\,Hz$, avec une autorythmicité d'environ $10\,Hz$, reflétant celle de la décharge des neurones de l'olive inférieure (Latham et Paul, 1971 ; Bell et Kawasaki, 1972 ; Lang et al., 1999). Des résultats contra-

dictoires ont été observés concernant l'effet de ces PAs complexes sur la décharge des PAs simples : les PA complexes ont été associés à une suppression momentanée des PAs simples (Latham et Paul, 1971) ou au contraire à une augmentation temporaire de leur fréquences de décharge (Sato et al., 1992).

Plus récemment, Loewenstein et al. (2005) ont mis en évidence la bistabilité du potentiel de membrane et de la décharge des PCs *in vivo*, chez l'animal anesthésié, qui alterne entre décharge tonique et quiescence. Ils montrent aussi que les transitions d'un état à l'autre (actif ou "up" *vs* quiescent ou "down") peuvent être contrôlées par l'excitation de la PC produite par sa fibre grimpante. Cette observation explique les résultats contradictoires évoqués ci-dessus : un PA complexe supprime la décharge de PAs simples s'il fait passer la PC de l'état actif à l'état quiescent (Armstrong et Rawson, 1979), mais il peut causer le phénomène inverse.

2.2 L'inhibition des cellules de Purkinje

L'activité synaptique dont fait l'objet la PC est susceptible d'influencer considérablement la décharge spontanée décrite dans la section précédente. Dans les tranches, en l'absence de toute stimulation, il n'y a que très peu d'activité synaptique excitatrice dans les PCs. Par contre, l'activité synaptique inhibitrice des interneurones est importante. Si, en outre, certaines entrées inhibitrices sont concomitantes dans plusieurs PCs voisines, elles peuvent contribuer à synchroniser, ou tout au moins à corréler, les décharges de ces PCs. En 1981, Ebner et Blödel spéculent : "The axons of individual basket and stellate cells are distributed transversely and contact several Purkinje cells. Through their inhibitory action on Purkinje cells, these interneurons may produce correlations between the activity of transversely oriented Purkinje cells". Ils ajoutent, concernant les synpases PC-PC : "The axon collateral system of Purkinje cells also could contribute to the correlations in the discharge of neighboring neurons through their action on inhibitory interneurons and Purkinje cells". Cette section reprend les éléments les plus saillants de l'inhibition des PCs, que ce soit par les interneurones (paragraphes 2.2.1, 2.2.2, 2.2.3) ou par les PCs voisines (paragraphe 2.2.4). Elle montre ainsi en quoi les spéculations d'Ebner et Blödel citées plus haut sont fondées et méritent notre intérêt. Les résultats présentés dans ce chapitre ont tous été obtenus dans les tranches de cervelet de mammifères.

2.2.1 Effets de l'inhibition des interneurones sur la décharge des cellules de Purkinje

Comme les PCs, les interneurones de la couche moléculaire sont spontanément actifs dans les tranches de cervelet (Llano et Gerschenfeld, 1993a), même en l'absence de toute excitation glutamatergique (Häusser et Clark, 1997). A température ambiante, la fréquence de décharge typique des interneurones est de $2-3\,Hz$; elle est de l'ordre de $12\,Hz$ à $34^{\circ}C$.

Leurs PAs sont responsables de courants postsynaptiques inhibiteurs (IPSCs) dans les PCs qu'ils contactent. Vincent et Marty (1996) ont montré la très grande variabilité

des amplitudes des IPSCs d'une PC postsynaptique causés par un interneurone présynaptique chez le jeune rat, en particulier lorsque cet interneurone est une cellule en panier. Cette dernière est capable de provoquer des IPSCs de très grande amplitude dans la PC, de l'ordre de $1\,nA$.

Pouzat et Hestrin (1997) ont montré que l'amplitude de ces IPSCs chutait d'un facteur 11 au cours du développement, entre 11 et 31 jours après la naissance. Cette baisse spectaculaire de l'amplitude moyenne des IPSCs s'accompagne également d'une baisse notable de leur variabilité. En revanche, le nombre d'interneurones présynaptiques à une PC augmente sur cette période. Ainsi, la fréquence des IPSCs dans une PC augmente au cours du développement. Elle est de 5 à $20\,Hz$ chez le jeune rat ($9-15$ jours), à température ambiante.

Ces courants inhibiteurs ne sont pas sans influence sur la décharge des PCs. Midtgaard (1992) a montré qu'une succession de PAs issus d'une cellule étoilée pouvaient inhiber la décharge d'une PC dans le cervelet de tortue. Häusser et Clark (1997) montrent quant à eux qu'un PA unique d'un interneurone est capable de retarder significativement le PA d'une PC postsynaptique : lorsqu'un PA présynaptique intervient entre deux PAs postsynaptiques, l'intervalle entre ces derniers est en moyenne 12% plus long (Fig. 2.6A). Vincent et Marty (1996) avaient anticipé ce résultat au vu de l'amplitude de nombre d'IPSCs enregistrés, notamment ceux dus aux cellules en panier. Ainsi, un interneurone individuel a un effet significatif sur la décharge d'une PC. La durée des intervalles de temps entre PAs (*inter-spike intervals*, ISIs) d'une PC est donc en partie contrôlée par la présence ou non d'un IPSC dans cet intervalle, et par l'amplitude de cet IPSC s'il est présent. Corollairement, la variabilité des ISIs d'une PC est liée à celle des amplitudes de ses IPSCs. Les ISIs les plus longs sont quant à eux sans doute dus à la sommation des courants inhibiteurs causés par plusieurs interneurones. Dans la perspective du travail de Häusser et Clark (1997), la régularité de la décharge intrinsèque d'une PC est perturbée par les PAs des interneurones qui la contactent. L'activité inhibitrice de ces derniers transforme le patron de décharge régulier des PCs en un patron de décharge irrégulier. Par ailleurs, la suppression de l'inhibition par les interneurones provoque une augmentation de la fréquence de décharge spontanée des PCs d'environ 40%.

Que les entrées inhibitrices très variables des interneurones sur une PC participent à la variabilité de ses ISIs ne fait guère de doute. Cependant, la grande régularité et l'exclusivité de la décharge tonique de toutes les PCs enregistrées dans les tranches par Häusser et Clark (1997) fait contraste avec les résultats expérimentaux décrits par Womack et Khodakhah (2002) et Llinas et Sugimori (1980a,b) et les résultats théoriques de Jäger et al. (1997) (voir section 2.1.4). Ces travaux mettent en effet en évidence des décharges spontanées en *bursts* ainsi que des silences prolongés, alternant avec ces périodes de décharge tonique. Womack et Khodakhah (2002) suggèrent que cette différence expérimentale s'explique par la durée beaucoup plus courte des enregistrements de Häusser et Clark (1997), qui auraient, de ce fait, manqué le patron trimodal des décharges de leurs PCs. Ils suggèrent aussi que les PCs en mode de décharge tonique sont en général en moins bonne santé que celles en mode trimodal.

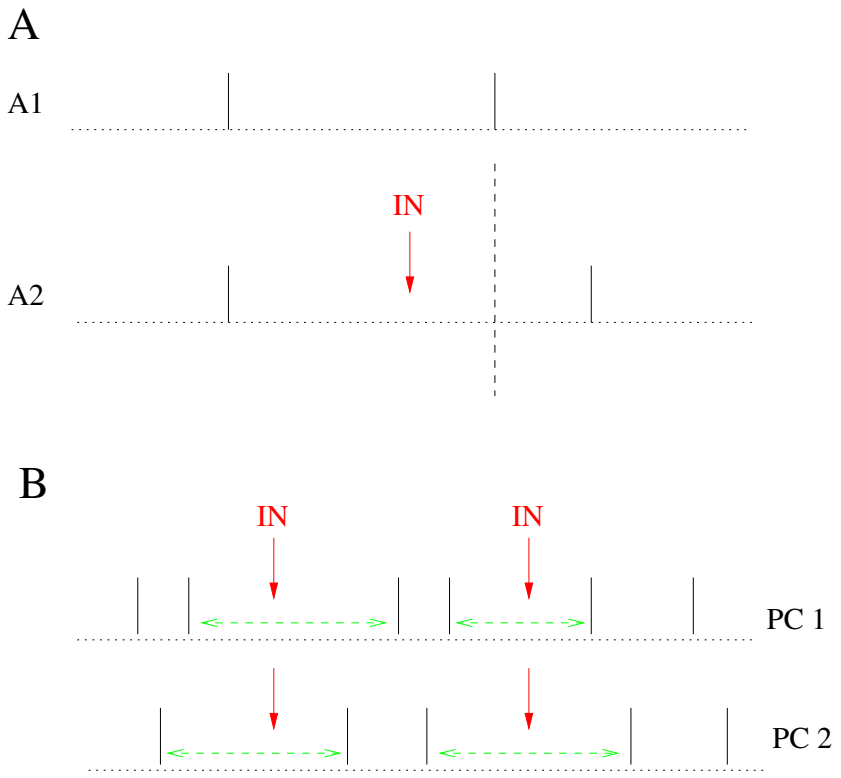

Figure 2.6: **Effet schématique de l'inhibition commune de deux cellules de Purkinje par un interneurone.** A1, ISI d'une cellule de Purkinje sans potentiel d'action présynaptique d'interneurone. A2, un potentiel d'action d'un interneurone présynaptique (IN, flèche verticale) retarde significativement le potentiel d'action de la cellule de Purkinje et prolonge l'ISI (12 % en moyenne, voir texte). B, deux cellules de Purkinje (PC 1 et PC 2) possédant un même interneurone présynaptique voient certains de leurs potentiels d'action retardés simultanément lors des potentiels d'action de cet interneurone (IN, flèches verticales). Les trains de ces deux cellules de Purkinje possèdent des ISIs prolongés concomitants (doubles flèches horizontales pointillées).

2.2.2 Existence d'interneurones communs à deux cellules de Purkinje

Les axones des cellules en panier et des cellules étoilées les plus internes de la couche moléculaire s'étendent sur 200 à 300 μm le long de la couche des PCs, transversalement aux fibres parallèles (Palay et Chan-Palay, 1974). Sur cette distance, ils forment donc potentiellement des synapses inhibitrices sur plusieurs PCs voisines, même s'ils ne contactent pas toutes les PCs qu'ils traversent (Bishop, 1993). Vincent et Marty (1993) ont montré l'existence d'IPSCs simultanés dans des paires de PCs espacées de moins de 290 μm. La proportion d'IPSCs qu'une PC partage avec une autre (*i.e* le rapport entre le nombre des IPSCs d'une PC qui sont communs à une autre et le nombre total des IPSCs de cette PCs) décroît avec la distance qui les sépare. Elle varie de 13 à 78% pour des cellules séparées de $25 - 50\,\mu m$. A ces distances, une paire sur 23 uniquement ne montre aucune corrélation entre les IPSCs. Ces IPSCs communs à deux PCs disparaissent en présence de TTX, ils requièrent donc la génération de PAs présynaptiques. D'autre part, ils ne sont pas éliminés par la suppression de toute transmission glutamatergique excitatrice par le CNQX et l'APV. Ceci indique que la synchronie de ces IPSCs a pour origine les interneurones et non les afférences excitatrices en amont.

Il est donc très vraisemblable que ces IPSCs communs à deux PCs soient dus à l'activité d'un même interneurone présynaptique. La longueur des axones de cellules en panier est compatible avec la distance maximale de couplage des IPSCs trouvée par Vincent et Marty. Cependant, on ne peut pas exclure que certains de ces IPSCs synchrones soient dus à deux interneurones différents, couplés par des synapses électriques. Plusieurs travaux montrent en effet la présence de jonctions communicantes (*gap junctions*) entre interneurones inhibiteurs (Sotelo et Llinas, 1972 ; Mann-Metzer et Yarom, 1999). Ce point est détaillé dans la section 2.2.3. Observons simplement que le résultat est le même : les PCs voisines reçoivent une inhibition commune de la part des interneurones de la couche moléculaire. Cette inhibition commune est susceptible de jouer un rôle dans la formation de corrélations entre les décharges de PCs (Fig. 2.6B).

Chez le rat, seules 6 à 10 cellules en panier contactent une PC (Palay et Chan-Palay, 1974), et ce nombre est sans doute encore inférieur dans les tranches. La grande variabilité de la proportion d'IPSCs partagées par deux PCs est sans doute due aux grandes fluctuations relatives du nombre d'interneurones présynaptiques qui peuvent être communs à deux PCs. Par ailleurs, deux autres caractéristiques de cette inhibition commune ont été mises en évidence par les mêmes auteurs, lors d'un travail ultérieur (Vincent et Marty, 1996), et méritent d'être signalées. La première est la différence d'amplitude des IPSCs communs dans les deux PCs : l'inhibition de l'interneurone présynaptique commun est très souvent plus forte pour une PC que pour l'autre. Il est même envisageable que, dans le cas où l'une des deux PCs ne reçoit qu'une faible inhibition de cet interneurone, certains des IPSCs communs ne soient pas détectés, ou même absents, dans cette cellule. On peut d'ailleurs concevoir que certains PAs de l'interneurone présynaptique commun produisent un IPSC dans une PC et pas dans l'autre. La seconde caractéristique de cette inhibition commune est l'absence de corrélation entre les fluctuations d'amplitudes des IPSCs communs de deux PCs postsynaptiques : les fluctuations d'amplitudes de ces IPSCs dans une cellule ne sont pas corrélées à celles

de ces mêmes IPSCs dans l'autre cellules. Ces fluctuations sont donc locales et ne sont pas synchrones entre deux différentes synapses du même interneurone.

2.2.3 Synapses électriques et interneurones du cervelet

C'est dans les années 50, chez les invertébrés, qu'ont été découvertes les synapses électriques entre neurones (Furshan et Potter, 1959). L'existence de ces synapses dans le cerveau des mammifères n'a été admise que plus tard, très exactement depuis leur mise en évidence anatomique dans le cervelet (Sotelo et Llinas, 1972) et le cortex (Sloper, 1972). Depuis, de nombreux travaux ont montré l'importance de ce mode de communication neuronale, en particulier dans les réseaux d'interneurones inhibiteurs. Une revue récente fait l'inventaire de toutes les régions du cerveau dans lesquelles ces synapses ont été découvertes (Connors et Long, 2004). L'ensemble des travaux synthétisés par ces auteurs contribuent à forger l'image de réseaux d'interneurones inhibiteurs communiquant par couplage électrotonique.

Le site morphologique d'une synapse électrique est une structure spécialisée, appelée *jonction communicante* (*gap junction* en Anglais), et constitutée d'un amas de canaux joignant les cytoplasmes de deux cellules. Les synapses électriques sont beaucoup plus rapides que les synapses chimiques et sont bidirectionnelles. De plus, leur action est continue et ne dépend pas de l'émission de PA. Elles sont considérées comme un moyen puissant et efficace de synchroniser les activités des neurones, que ce soit leurs PAs ou les fluctuations de potentiel sous le seuil. C'est là la fonction principale qui leur est attribuée, et celle qui nous intéresse ici. Cette synchronisation par les synapses électriques a pu être constatée expérimentalement à de nombreuses reprises. Cependant, des études théoriques montrent qu'elle n'est pas systématique. La manière dont les synapses électriques peuvent synchroniser les neurones dépend en effet de deux facteurs : les propriétés intrinsèques des neurones (Pfeuty, 2003) et leur combinaison avec des synapses chimiques, notamment les synapses chimiques gabaergiques (Pfeuty, 2005).

Dans le cervelet, malgré leur découverte anatomique précoce, la mise en évidence électrophysiologique des synapses électriques s'est révélée délicate. Vincent et Marty (1996) rapportent que la grande majorité des interneurones enregistrés ne montrent pas de signe de couplage électrique. Par ailleurs, l'injection de neurobiotine dans un interneurone ne s'étend pas à un interneurone voisin, comme ce devrait être le cas en présence de jonctions communicantes (Pouzat et Hestrin, 1997). En revanche, Mann-Metzer et Yarom (1999) montrent que 40% des interneurones de la couche moléculaire enregistrés sont couplés électriquement. De plus, l'injection de biocytine dans un interneurone révèle souvent des groupes d'interneurones teints indirectement. Selon ces derniers auteurs, il est très probable que ces différences d'observation soient dues au fait que les jonctions communicantes sont encore peu développées aux âges des rats utilisés par les premiers auteurs (9 à 15 jours), alors qu'elles le sont pleinement chez leurs cochons d'Inde matures. Chez les animaux juvéniles, les dendrites des cellules étoilées superficielles, ainsi que leurs connexions électriques, sont seulement en cours de formation.

D'après Mann-Metzer et Yarom (1999), le couplage électrique qu'ils mettent en

évidence est assez fort pour synchroniser les décharges des interneurones mais la fenêtre temporelle de cette synchronisation est relativement grande, de l'ordre de la dizaine de *ms*. La rôle physiologique de cette synchronisation des interneurones inhibiteurs n'est pas clair. Ces auteurs avancent l'hypothèse qu'une inhibition efficace des dendrites distales d'une PC, notamment celle des PAs calciques évoqués par la fibre grimpante, peut nécessiter un "effort collaboratif", selon l'expression qu'ils emploient, d'un groupe de cellules étoilées. D'autre part, la chute spectaculaire, au cours du développement, de l'amplitude des courants synaptiques d'une PC évoqués par un PA d'un interneurone laisse penser que les activités d'interneurones, pour demeurer efficaces, doivent être mieux coordonnées chez l'animal mature (Pouzat et Hestrin, 1997).

Les cellules en panier quant à elles exercent individuellement une forte inhibition au niveau proximal et somatique et sont, de fait, beaucoup moins souvent couplées électriquement. Cette observation constitue d'ailleurs une seconde explication à l'absence de couplage électrique constatée par Vincent et Marty (1996) dont les enregistrements proviennent préférentiellement des cellules en panier. Celles-ci ne sont donc pas les seules à pouvoir donner naissance à des IPSCs synchrones dans deux PCs postsynaptiques voisines. La synchronisation, par es jonctions communicantes, d'interneurones plus périphériques de la couche moléculaire est un autre moyen d'obtenir une inhibition commune à plusieurs PCs, tout au moins chez l'animal mature.

2.2.4 L'inhibition des cellules de Purkinje par les cellules de Purkinje

Un axone de PC, après avoir traversé la couche des grains et rejoint la matière blanche, atteint les neurones des noyaux profonds du cervelet. Cependant, il possède de multiples collatérales récurrentes visibles dans la couche moléculaire et dans la couche des PCs (Ramon y Cajal, 1911 ; King et Bishop, 1982). Ces collatérales demeurent dans le plan sagittal mais couvrent une épaisseur plus importante que l'arbre dendritique de la PC. Des observations morphologiques ont montré qu'elles pouvaient contacter des PCs voisines et former des synapses axo-somatiques et axo-dendritiques chez le rat et le chat (Palay et Chan-Palay, 1974 ; King et Bishop, 1982). Chez la souris, Larramendi et Lemkey-Johnston (1970) n'ont observé de telles synapses de collatérales que sur les dendrites de PCs, jamais sur leurs somas. Dans cette étude, les principales cibles postsynaptiques des collatérales de PCs sont les cellules en panier.

Les synapses PC-PC ont également été mises en évidence fonctionnellement (Eccles et al., 1966). Selon cette étude, l'inhibition exercée par ces collatérales sur les PCs voisines, bien que réelle, reste faible, surtout si on la compare à celle exercée par les interneurones de la couche moléculaire. Dans l'ensemble, peu de travaux ont été consacrés à cette inhibition PC-PC à ce jour.

2.2.5 Conclusion

Les paragraphes précédents montrent que tous les éléments sont réunis pour que les trains de PAs de PCs ne soient pas indépendants, même au cours de leur activité spontanée et en l'absence d'entrées excitatrices des fibres parallèles. En effet, si un PA

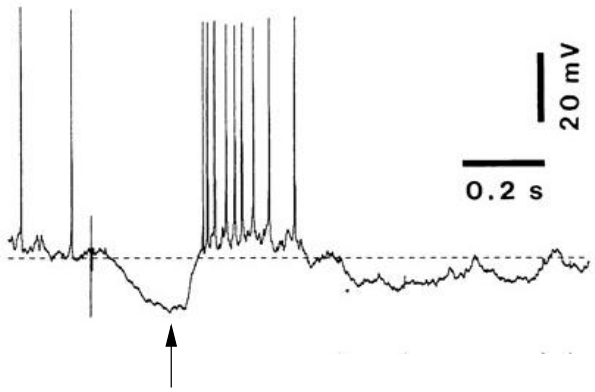

Figure 2.7: **Potentiel d'action en rebond: exemple d'excitation postinhibitrice** (Grenier et al., 1998). Un IPSC (flèche) est suivi d'un *burst* de potentiels d'action. Enregistrement intracellulaire d'un neurone cortical de l'aire motrice 4 chez le chat anesthésié.

présynaptique d'un seul interneurone peut retarder significativement celui d'une PC postsynaptique (Häusser et Clark, 1997) et si un même interneurone peut contacter plusieurs PCs voisines (Vincent et Marty, 1993), ou encore si plusieurs interneurones contactant plusieurs PCs voisines se synchronisent grâce à leurs jonctions communicantes (Mann-Metzer et Yarom, 1999), alors les PAs de ces PCs postsynaptiques peuvent être significativement retardés *simultanément*. Autrement dit, les IPSCs communs à plusieurs PCs voisines, quelle que soit la cause de cette simultanéité, doivent se traduire par des ISIs prolongés concomitants dans ces cellules (Fig. 2.6B). La question se pose alors de savoir si cette inhibition commune - ces pauses communes - est perceptible et détectable sur des trains de PAs de PCs enregistrées simultanément.

Si, de plus, ces IPSCs communs sont capables de produire un PA *en rebond* dans chaque cellule postsynaptique où ils se produisent, alors une synchronisation étroite des PAs de ces cellules est possible. De nombreux travaux ont en effet mis en évidence l'existence d'excitations postinhibitrices dans différents types neuronaux (Grenier et al., 1998). L'inhibition est alors suivie d'une dépolarisation remarquable causant l'émission d'un ou plusieurs PAs au terme de l'IPSC (Fig. 2.7). Il est envisageable que l'inhibition commune reçue par deux PCs donne lieu à une excitation commune synchrone qui lui succède (Fig. 2.8). Ce mécanisme de synchronisation a souvent été avancé dans la littérature, notamment pour les PCs (Ebner et Blödel, 1981 ; Isope et al., 2002). Il est donc intéressant de déterminer s'il est effectif dans les tranches de cervelet.

C'est dans cette perspective que se place mon travail expérimental. Il tente de répondre à trois questions, en relation aux différentes conséquences que peut avoir l'inhibition sur les décharges des PCs : (i) une synchronisation des PAs de PCs par les interneurones est-elle visible dans des enregistrements de leurs activités spontanées simultanées dans les tranches de cervelet ? (ii) à défaut, l'inhibition commune reçue par des PCs voisines peut-elle être détectée dans ces mêmes enregistrements ? (iii) enfin, si les synapses PC-PC sont fonctionnelles et suffisamment fortes, elles doivent induire des corrélations négatives entre trains de PAs de PCs, que l'on souhaite mettre en évidence.

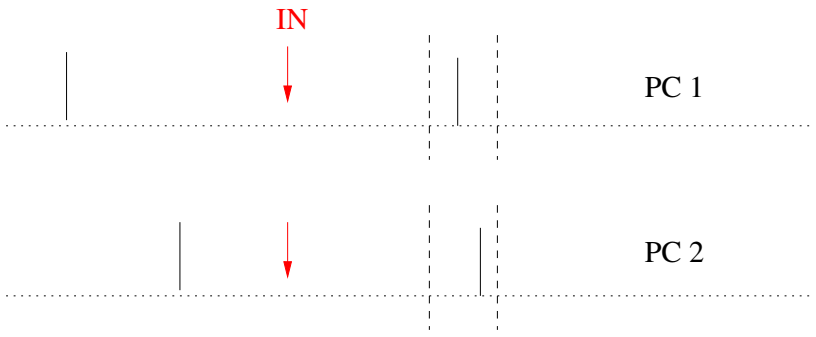

Figure 2.8: **Synchronisation schématique des potentiels d'action de deux cellules de Purkinje par une possible excitation postinhibitrice commune.** Si deux IPSCs reçus simultanément dans deux cellules de Purkinje (PC1 et PC2) causent dans chacune d'elles un potentiel d'action en rebond, alors ceux-ci sont émis dans une fenêtre étroite après le potentiel d'action présynaptique (flèche verticale) de l'interneurone commun (IN). Cette fenêtre de synchronisation est schématisée par deux lignes verticales pointillées.

2.3 La synchronisation des potentiels d'action simples (*simple spikes*)

Dans les tranches de cervelet, à température ambiante et en l'absence de stimulation, les PCs ne reçoivent pas d'entrée excitatrice de leurs fibres grimpantes ; elles n'émettent donc pas de PAs complexes spontanés dans ces conditions. Par conséquent, seuls les travaux relatifs aux relations temporelles entre trains de PAs simples de PCs sont abordés dans cette section. Ceux qui sont consacrés à l'autorythmicité et la synchronisation des PAs complexes constatés dans différentes préparations *in vivo* (animaux anesthésiés, décérébrés, éveillés) ne sont pas revus ici. Cette rythmicité et cette synchronie des PAs complexes (Bell et Kawasaki, 1972) sont le reflet de l'activité oscillante à $8-12\,Hz$ des neurones de l'olive inférieure, largement couplés par des jonctions communicantes (Llinas et Yarom, 1986).

2.3.1 Corrélations PA à PA

L'étude de la synchronisation des PAs simples des PCs a fait l'objet de plusieurs travaux, sur différentes préparations, au cours des dernières décennies. Le premier article à ce sujet est celui de Bell et Grimm (1969). Ces auteurs ont enregistré extracellulairement l'activité spontanée de paires de PCs, à l'aide de microélectrodes de verre remplies de KCl, *in vivo*, sur le chat anesthésié. Ils limitent leur analyse des corrélations croisées des trains aux paires de PCs dont chaque cellule est enregistrée sur *l'une des deux électrodes seulement*, celles-ci étant espacées de 50 à $100\,\mu m$, le long de l'axe des fibres parallèles. Sur les cinq paires analysées, trois présentent un pic, centré sur $0\,ms$ et large de quelques ms, dans leur histogramme de corrélation croisée (*cross-correlogram*, CC). Sur ces trois pics, l'un seulement semble réellement significatif (l'analyse de la significativité des pics n'est pas présentée par leurs auteurs, laissant le choix au lecteur d'apprécier).

Dans un travail légèrement postérieur, Bell et Kawasaki reproduisent ces résultats sur le chat *in vivo*, anesthésié et non anesthésié (Bell et Kawasaki, 1972). Cinq paires de PCs sur quinze présentent un pic centré sur 0, et large de $3-4\,ms$, dans leurs CCs entre PAs simples, en activité spontanée. Ces cellules sont également situées sur un axe parallèle aux fibres parallèles et sont espacées de moins de $65\,\mu m$.

De même, avec une technique identique à Bell et Grimm, Ebner et Blödel (1981) ont noté une corrélation positive, sur une courte échelle de temps (de l'ordre de quelques ms) entre PAs simples spontanés de PCs espacées de moins de $100\,\mu m$, dans l'axe des fibres parallèles, chez le chat *in vivo* non anesthésié. Deux des cinq paires de PCs enregistrées sur les deux microélectrodes montrent une corrélation positive semblable de leurs PAs simples. Tous ces auteurs attribuent la synchronisation de ces paires de PCs à leur excitation commune par les fibres parallèles qu'elles partagent. Le degré de corrélation entre deux PCs reflète la proportion de fibres parallèles qu'elles ont en commun.

Par ailleurs, Bell et Grimm (1969) d'une part, Ebner et Blödel (1981) d'autre part, montrent des corrélations positives entre PCs enregistrées sur une *même microélectrode* (respectivement deux paires et six paires sur onze). Dans le cas de ces paires, les pics dans les CCs ne sont pas centrés sur 0 mais sont décalés de quelques ms. Ces CCs sont obtenus aussi bien en activité spontanée qu'après stimulation, en l'occurrence un léger étirement du muscle gastrocnemius-soleus (Ebner et Blödel, 1981).

Nous avons vu que les PCs étaient capables d'émettre des doublets de PAs de très haute fréquence (section 2.1.3). Bien que les auteurs affirment que les PAs qu'ils enregistrent proviennent de PCs voisines distinctes, l'hypothèse selon laquelle les PAs qu'ils enregistrent sur une même électrode sont des doublets émis par *une même PC* ne semble pas pouvoir être exclue définitivement, du moins sur la base de ce qui est publié. Dans cette hypothèse, la corrélation à $2-5\,ms$ ou $6\,ms$ entre ces PAs, observée en activité spontanée et/ou après étirement du muscle, ne reflète pas une corrélation entre deux PCs mais entre les PAs des *bursts* d'une même PC. Un premier argument en faveur de cette hypothèse des doublets s'appuie sur la valeur des amplitudes des PAs des deux PCs supposées : le rapport des amplitudes de ces PAs, approximativement égal à deux, est tout à fait comparable à celui des PAs d'un doublet. C'est d'ailleurs sur la base de cette différence d'amplitude notable des PAs des deux PCs que repose leur séparation par les auteurs.

Un deuxième argument, temporel cette fois, vient s'ajouter à cet argument d'amplitude. La position des pics dans les CCs de ces paires se situe entre 2 et $5\,ms$ dans le cas d'Ebner et Blödel, à $6\,ms$ dans le cas de Bell et Grimm. Cet intervalle de temps est également celui qui sépare les deux PAs d'un doublet d'une PC. De plus, les CCs montrés par les auteurs ne sont pas symétriques : le pic n'apparaît qu'à des temps positifs, d'où l'on conclut que c'est toujours la même prétendue cellule qui précède l'autre. Dans le travail de Bell et Grimm, ce sont les PAs de petite amplitude qui suivent toujours ceux de grande amplitude, conformément à ce que l'on attend dans l'hypothèse des doublets. Les arguments temporel et d'amplitude se rejoignent et s'associent donc ici pour soutenir cette hypothèse. Ebner et Blödel ne précisent pas si ce sont les PAs de grande amplitude qui précèdent toujours ceux de plus petite amplitude. Il serait intéressant de le savoir. Car alors ces six paires de PCs ne pourraient être en fait que six

PCs uniques et les auteurs montreraient simplement que les PCs peuvent émettre des doublets spontanément, et/ou après stimulation par étirement du muscle. Il ne s'agit plus d'un CC de deux cellules, mais d'un histogramme d'intervalles entre PAs d'une cellule émettant des doublets de PAs espacés de $2-5\,ms$ ou $6\,ms$.

Cette hypothèse des doublets n'est pas évoquée et les auteurs attribuent ces corrélations entre PCs enregistrées sur une même électrode à une excitation commune par les fibres parallèles qu'elles ont en partage, comme dans le cas de PCs enregistrées sur les deux électrodes séparément.

Jäger (2003) quant à lui n'a jamais obtenu que des CCs plats pour toutes les paires de PCs qu'il a pu enregistrer *in vivo* ($n = 30$ paires), chez le rat anesthésié, le long de l'axe des fibres parallèles, sur des distances de 0.1 à $1.5\,mm$. Contrairement aux auteurs précédents, il ne constate aucune synchronisation précise PA à PA dans ces paires, en activité spontanée et lors de stimulations périphériques (pression d'air sur la lèvre supérieure). Or, les simulations qu'il effectue par ailleurs montrent que même un très faible pourcentage des fibres parallèles activées de façon synchrone provoque une synchronisation temporelle précise des PCs, se traduisant par l'apparition d'un pic central significatif dans les CCs. Cette synchronisation PA à PA étant absente, il en conclut que ces "vollées", pour reprendre son terme, de fibres parallèles synchrones n'existent pas. Les expériences de Chéron et al. (2004) confirment ces résultats. Ces auteurs n'ont obtenu que des CCs plats pour les paires de PCs enregistrées *in vivo* ($n = 48$ paires), chez la souris sauvage éveillée, le long de l'axe des fibres parallèles, sur des distances allant jusqu'à $0.5\,mm$.

2.3.2 Corrélations de longues durées

Ebner et Blödel (1981) montrent l'existence de corrélations positives et négatives entre PCs sur des échelles de temps plus longues, de l'ordre de $[-500; +500\,ms]$, lors de leur activité spontanée. Ces corrélations ne sont pas des corrélations PA à PA, telles que celles décrites jusqu'à présent. Elles sont beaucoup plus lâches et diffuses, leur durée étant de plusieurs dizaines de ms. En effet, les CCs de ces paires présentent soit un pic, soit un creux, larges, centrés sur 0, sur un intervalle typique de $[-100; +100\,ms]$. Il en va ainsi pour 25% de leurs paires de PCs situées le long de l'axe des fibres parallèles, à des distances pouvant aller jusqu'à $750\,\mu m$.

L'application d'un stimulus périphérique sinusoïdal cutané module la fréquence de décharge de certaines PCs. Si cette modulation se fait dans le même sens pour les deux PCs (resp. dans des sens contraires), le CC de la paire présente un pic (resp. un creux) semblable à celui décrit en activité spontanée. Ces CCs reflètent une évolution, en phase ou en opposition de phase, des fréquences de décharges des PCs des paires étudiées, en réponse au stimulus. Ce phénomène peut être avancé pour expliquer les corrélations longues, positives et négatives, observées en régime spontané : des changements concomitants de fréquences de deux PCs peuvent apparaître spontanément. En revanche, les auteurs mettent en évidence l'apparition de corrélations longues, positives uniquement, pour des paires dont les fréquences de décharge ne sont pas modulées par la stimulation, ou modulées différemment, selon une structure temporelle incompatible avec la

corrélation de leur CCs. Pour ces paires, sept sur dix-huit, l'apparition de corrélations positives semble indépendante de leurs changements de fréquence de décharge. De plus, l'amplitude de ces corrélations dépend de la fréquence de la stimulation périphérique.

L'origine de ces corrélations longues particulières entre PAs simples n'est pas claire pour ces auteurs. Ces derniers favorisent l'hypothèse selon laquelle elles seraient provoquées par les fibres grimpantes, selon trois mécanismes envisageables : (i) une même fibre grimpante active plusieurs PCs, (ii) plusieurs fibres grimpantes sont activées simultanément par le stimulus externe et activent de façon synchrone plusieurs PCs, (iii) les fibres grimpantes exercent cet effet par l'intermédiaire des interneurones de la couche moléculaire. Le travail présenté par les auteurs ne permet pas de conclure sur la validité de ces hypothèses.

Jäger (2003) a également mis en évidence l'existence de corrélations de longue durée dans les CCs de PCs en activité spontanée et lors de stimulations sensorielles par pression d'air sur la lèvre supérieure. Ces CCs présentent de larges pics (100 à 200 ms) centrés sur 0. Ils correspondent, selon l'auteur à des modulations conjointes des fréquences de décharge des PCs de la paire. Selon lui, celles-ci sont dues à la forte modulation de l'activité asynchrone des cellules en grain. Contrairement à Ebner et Blödel, Jäger ne rapporte pas de corrélations longues entre PCs autres que celles-ci.

2.3.3 Cervelet et oscillations

La synchronisation neuronale est souvent associée à l'émergence de rythmes de l'activité cérébrale. Des oscillations de champs local de potentiels (*local field potential*, LFP) ont été mises en évidences dans de nombreuses structures et sur une large gamme de fréquences, allant de quelques Hz à plusieurs centaines de Hz. Ces oscillations reflètent l'activité synchrone de nombreux neurones dans ces structures.

Pellerin et Lamarre (1997) ont pu mettre en évidence de telles oscillations de LFP dans le cortex cérébelleux chez le singe éveillé. Un certain nombre d'arguments indirects mènent les auteurs à la conclusion que ces oscillations à $13-18\,Hz$ sont générées dans la couche des grains. Elles sont par ailleurs modulées par le comportement de l'animal. Cependant, ces auteurs n'observent aucune synchronisation des PAs de PCs sur ces oscillations.

Hartmann et Bower (1998) ont également enregistré des oscillations de LFP dans la couche des grains du cortex cérébelleux de rats éveillés, libres de leurs mouvements. La fréquence de ces oscillations, de $7-8\,Hz$, est légèrement inférieure à celle de Pellerin et Lamarre (1997). Ces deux études montrent que ces oscillations sont présentes si l'animal est immobile. Elles sont interrompues par un mouvement et lors d'une stimulation sensorielle. Hartmann et Bower (1998) considèrent ces oscillations comme une propriété fondamentale du système ; elles en constituent en quelque sorte l'état de base. Le modèle détaillé du réseau de la couche des grains élaboré par Maex et De Schutter (1998) vient appuyer ces résultats expérimentaux : il est capable de générer des oscillations de 10 à $40\,Hz$. Cette étude théorique montre donc que le réseau des cellules de Golgi et des grains est particulièrement propice à la génération de tels rythmes dans le cortex cérébelleux.

Par ailleurs, Lu et al. (2005) montrent, chez le rat anesthésié, que les cellules en grain

émettent spontanément des *bursts* avec une fréquence variant de 2 à 7 Hz. Surtout, ils montrent que les PAs simples d'une PC et les *bursts* d'activité des cellules en grain *situées juste en-dessous* sont corrélés sur de courtes échelles de temps (de l'ordre de quelques ms de latence entre un *burst* dans la couche des grain et le PA simple de la PC). La nature de cette corrélation entre une PC et les cellules en grain sous-jacentes est variable. Elle est positive dans la plupart des cas, mais elle peut être négative ou inexistante. Dans le cas où la corrélation est positive, elle peut être très brève (pic étroit dans l'histogramme de corrélation croisée) ou bien de longue durée (pic large de plusieurs dizaines de ms). La nature de cette corrélation ne change jamais pour une PC donnée. Les auteurs montrent également que la force de cette corrélation décroît avec la fréquence de décharge des PCs, ce qui n'est pas étonnant puisqu'alors la différence de fréquence de décharge entre PCs et grains augmente. De plus, ils constatent que la corrélation est maximale lorsque les grains émettent leurs *bursts* à environ $5Hz$. Aucune hypothèse ne permet d'expliquer cette dernière observation. Les auteurs attribuent ces résultats à l'excitation directe de la PC par les *segments ascendants* des axones des cellules en grain juste sous-jacentes. A aucun moment, ils n'étudient une éventuelle synchronisation de plusieurs PCs relativement à ce phénomène.

Adrian (1935) avait quant à lui enregistré, à la surface du cervelet, des oscillations de LFP de nature différente, puisque de fréquence beaucoup plus élevée (150 à 250 Hz), chez le chat anesthésié. Aucun travail récent ne reproduit ces résultats. En revanche, des oscillations de LFP de fréquence similaire (160 Hz) ont été mises en évidence dans le cortex cérébelleux de souris éveillées *knock-out* pour la calbindine et/ou la calrétinine, deux protéines liant le calcium et impliquées dans le façonnement des signaux calciques intracellulaires (Chéron et al., 2004). Ces oscillations sont interrompues par toute stimulation sensorielle. Dans cette étude, les auteurs montrent que la fréquence de décharge des PCs augmente dramatiquement chez ces souris, en lien avec une excitabilité accrue des cellules en grain. De plus, les décharges des PCs, le long des fibres parallèles, sont étroitement synchronisées et en phase avec les oscillations rapides de LFP ; elles en sont vraisemblablement le principal générateur. Le blocage de la transmission gabaergique ou celui des jonctions communicantes supprime ces oscillations ; les auteurs attribuent donc un rôle central au réseau des interneurones de la couche moléculaire central dans leur émergence. Chéron et al. (2005) ont fait des observations similaires en tous points sur d'autres souris mutantes, ataxiques, atteintes d'un syndrome d'Angelman. Ces deux travaux constituent, à ma connaissance, *les seuls exemples de synchronisation massive et robuste des PCs*. A ce stade des investigations, il n'est pas possible d'établir un lien entre ces résultats et ceux d'Adrian.

2.3.4 Conclusion

Deux types de corrélations entre PAs simples de PCs ont été donc mis en évidence *in vivo*, pour des paires situées dans l'axe des fibres parallèles : une *synchronisation précise PA à PA* et une *corrélation plus lâche de longue durée*.

Concernant la synchronisation PA à PA, il semble difficile de tirer une conclusion définitive au vu du peu de données dont on dispose. La significativité des quelques pics

de CCs montrés dans la littérature n'est pas établie par les auteurs. Par ailleurs certains sont centrés sur $0\,ms$ et symétriques, d'autres sont centrés sur $\sim 3-6\,ms$ et ne sont pas symétriques. Ces derniers apparaissent pour des PCs enregistrées sur une même électrode. Il semble que nombre de ces paires pourraient en fait être une seule et même cellule émettant des doublets. Rien dans les données telles qu'elles sont publiées ne permet d'exclure cette hypothèse de façon certaine. Enfin, Jäger (2003) et Chéron et al. (2004) montrent très clairement une totale absence de ces synchronisations de PCs dans leurs enregistrements, jetant un doute notable sur leur existence réelle. Si elles existent, les auteurs les attribuent unanimement à une excitation commune des PCs d'une paire par les fibres parallèles qu'elles partagent. Selon Isope et al. (2002), cette synchronie de l'excitation de PCs par les fibres parallèles, remise en cause par Jäger (2003) et Chéron et al. (2004), est l'un des mécanismes possibles expliquant les oscillations à très haute fréquence ($\sim 180\,Hz$) des potentiels de champ enregistrés par Adrian (1935) dans le cortex cérébelleux chez le chat anesthésié.

Quoiqu'il en soit, aucune corrélation PA à PA n'a jamais été mise en évidence *in vivo* entre deux PCs séparées par plus de $100\,\mu m$ le long des fibres parallèles, comme on l'attendrait si ces dernières étaient en mesure de contrôler la sortie des PCs. Cette constatation, associée à nombre d'études physiologiques, anatomiques et théoriques, mène certains auteurs à la conclusion que *les fibres parallèles ne provoquent pas elles-mêmes l'émission de PAs par les PCs* (Bower et Woolston, 1983). Ce serait plutôt le rôle des synapses associées aux *segments ascendants des axones des grains*, celui des synapses entre fibres parallèles et PCs étant, lui, de contrôler la dynamique des dendrites (Lu et al., 2005 ; Santamaria et Bower, 2005).

Les corrélations de longues durées semblent en revanche solidement établies. Elles apparaissent en régime spontané et pendant des stimulations périphériques. Elles reflètent généralement des changements simultanés de fréquence de décharge de deux PCs. Les quelques cas rapportés par Ebner et Blödel (1981) où ces corrélations semblent indépendantes de ces modulations de fréquence restent inexpliqués.

Les oscillations lentes observées *in vivo* dans le cortex cérébelleux, en l'absence de mouvement et de stimulation sensorielle, ne semblent pas synchroniser les PCs, contrairement aux oscillations rapides des souris *knock out* et mutantes de Chéron et collaborateurs. Cependant, ces oscillations rapides remarquables n'apparaissent que lorsque les signaux calciques intracellullaires du cervelet sont perturbés par l'absence de protéines liant le calcium comme la calrétinine et la calbindine. Se pose donc la question de leur pertinence et de leur signification physiologique. A l'heure actuelle, il s'agit de la seule démonstration robuste et incontestée de synchronisation des PAs simples de PCs, toutes préparations confondues (*in vivo* et dans les tranches en particulier).

Il n'y a pas de réelle étude de la synchronisation PA à PA des PCs *in vitro*. La seule mention qui en est faite apparaît dans un article méthodologique décrivant la mise au point d'une matrice plane de microélectrodes pour l'enregistrement de l'activité d'une tranche sagittale de cervelet (Egert et al., 2002). Tous les CCs des paires de cellules enregistrées en activité spontanée par ces auteurs sur des tranches sagittales de rats sont plats ($n = 40$).

Chapitre 3

La classification des potentiels d'action (*spike-sorting*)

Le problème du *spike-sorting* est ancien puisqu'il se pose à l'expérimentateur dès lors qu'il effectue un enregistrement extracellulaire de l'activité neuronale (Krüger, 1983). De nombreuses solutions, souvent partielles ou *ad hoc*, ont été proposées. Les revues de référence en la matière sont celles de Schmidt (1984a,b) et Lewicki (1998). Aucune de ces solutions ne résoud définitivement le problème et aucune ne s'est réellement imposée à l'heure actuelle (Brown et al., 2004). En 1990 déjà, Jansen écrit à propos du problème du *spike-sorting* : "The magnitude of the problem is probably best illustrated by the number of different methods that have been described" (Jansen, 1990).

Les décours temporels, ou encore les *formes*, des potentiels d'action (PAs) enregistrés extracellulairement diffèrent selon le type et la distribution spatiale des courants dans la cellule. Ils diffèrent également selon la position et la géométrie de l'électrode. La distance séparant la cellule d'un site d'enregistrement est un élément déterminant de la forme des PAs de cette cellule sur ce site. Rappelons ici que les variations de potentiels extracellulaires, de l'ordre de la centaine de μV, sont très inférieures à celles que l'on peut enregistrer dans la cellule, en raison de la très faible résistance du milieu extracellulaire au regard de celle de la membrane cellulaire ; ces variations de potentiels extracellulaires s'atténuent à mesure que l'on s'éloigne des courants transmembranaires qui en sont l'origine. Gardons à l'esprit qu'un PA extracellulaire est, en règle générale, visible jusqu'à $50 - 80\,\mu m$ du neurone qui l'a émis. Si les formes des PAs de deux neurones diffèrent sur un site d'enregistrement, c'est en premier lieu parce qu'ils sont situés à des distances différentes de ce site (Fig. 1.2). Par conséquent, si la distance du site aux neurones qu'il enregistre n'est pas modifiée, et si un neurone émet toujours le même PA, celui-ci peut être considéré comme la signature de ce neurone sur ce site.

Toutes les méthodes de *spike-sorting* présentées dans ce chapitre se fondent sur ces considérations et effectuent la classification des PAs selon leurs formes. Le dogme est donc : "une forme de PA=un neurone (au bruit près)". Sont ainsi considérés comme ayant été émis par un même neurone les PAs dont les différences de forme ne sont dues qu'au bruit. Deux PAs dont la différence de forme ne peut être expliquée par le bruit sont considérées comme issus de deux neurones distincts.

Les méthodes actuelles de *spike-sorting* sont, pour la plupart, des méthodes dites

de *clustering* : les PAs, représentés par des vecteurs dans un espace judicieux, forment, dans cet espace, des nuages de points qui sont ensuite identifiés et dont on délimite les contours. A chaque nuage est associé un neurone. L'hypothèse sous-jacente est qu'un neurone donné émet des PAs de forme unique (*i.e* un point dans l'espace de représentation des PAs) à laquelle s'ajoute un bruit indépendant, à l'origine du nuage de points. Après détection des PAs, les méthodes actuelles de *spike-sorting* procèdent donc généralement en deux temps (Fig. 3.1) :

1. Extraction des caractéristiques distinctives des formes de PAs (*feature extraction*, section 3.2) : cette étape vise à décrire chaque PA par un nombre réduit m de caractéristiques discriminantes, formant un vecteur de dimension m.
2. Délimitation des nuages de points (*clustering*, section 3.3) : dans l'espace déterminé à l'étape précédente, les PAs issus d'un même neurone forment un nuage de points. Les nuages de points correspondant aux différents neurones enregistrés sont identifiés lors de cette étape.

Fondamentalement, les méthodes existantes, à l'exception des plus anciennes qui font l'objet de la première section de ce chapitre, ne diffèrent que dans leur manière de réaliser ces deux étapes. Ce chapitre dresse un panorama des grandes classes de méthodes couramment utilisées. Le but ici n'est pas de rentrer dans le détail d'implémentations particulières publiées dans la littérature, les références citées permettent de s'y reporter. Il n'est pas non plus de développer les théories mathématiques sur lesquelles certaines de ces méthodes se fondent. Il existe sur ces sujets des ouvrages de référence que le lecteur intéressé pourra consulter. C'est le principe et l'architecture de chaque méthode qui sont exposés dans ce chapitre. Ce dernier doit permettre avant tout de comprendre comment les électrophysiologistes ont abordé et traité le problème du *spike-sorting* jusqu'à ce jour. Il doit permettre de saisir quels en sont les tenants et les aboutissants, et quelles sont les grandes difficultés auxquelles nous devons faire face.

3.1 Les premières méthodes

3.1.1 Fenêtre d'amplitude

La caractéristique la plus évidente d'un PA est sans aucun doute son amplitude. La manière la plus élémentaire d'isoler l'activité du neurone dont les PAs sont les plus grands sur une électrode extracellulaire est de sélectionner les PAs dont l'amplitude dépasse un seuil judicieusement déterminé (seuil C dans la Fig. 3.2A). Ce procédé élémentaire ne fournit l'activité que d'un seul neurone. Si on poursuit un peu plus loin cette idée, on peut également sélectionner comme provenant d'un même neurone tous les PAs dont l'amplitude se situe dans un intervalle d'amplitudes pertinemment choisi (*window discriminator*) (Fig. 3.2A). Cette méthode rudimentaire de tri des PAs est effectuée au niveau du *hardware*. Elle permet de séparer, sur une électrode, un ou deux neurones dont les PAs ont des amplitudes bien distinctes, situées dans des fenêtres d'amplitudes séparées (Schmidt, 1984a).

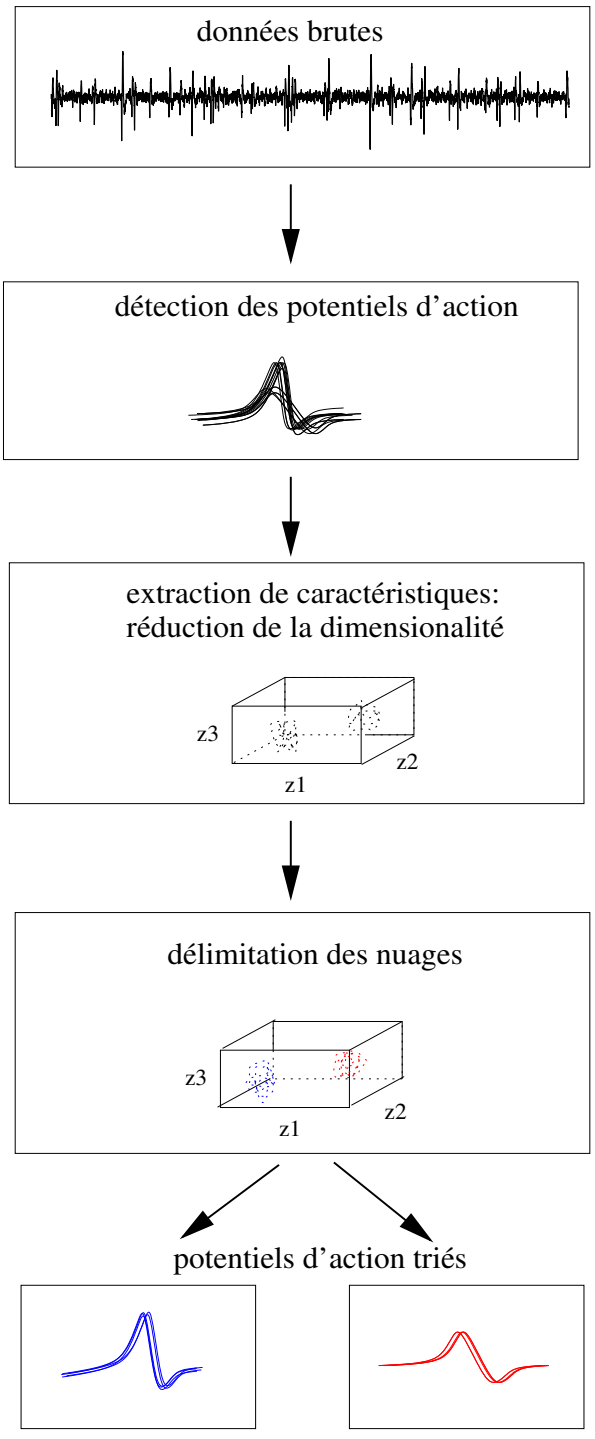

Figure 3.1: **Etapes générales de la procédure de classification des potentiels d'action (*spike-sorting*).** Après détection sur les données brutes, les potentiels d'action sont représentés par un ensemble de caractéristiques discriminantes (*feature extraction*) formant des nuages de points qui sont ensuite séparés les uns des autres (*clustering*). Les potentiels d'action d'un même nuage sont considérés comme venant d'un même neurone.

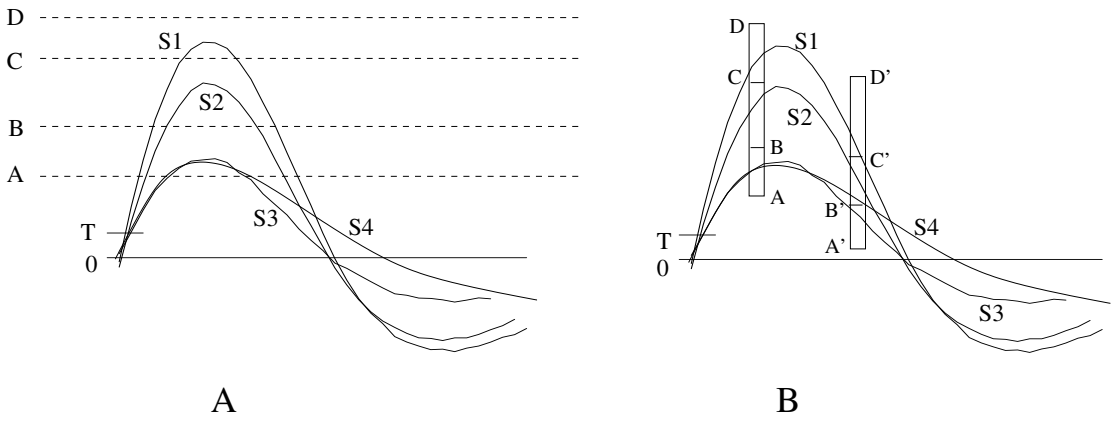

Figure 3.2: **Discrimination des formes de potentiel d'action par seuillage et par fenêtres temps-amplitude.** A, quatre formes de potentiels d'action (S1-S4) sont superposées et horizontalement alignées de sorte que toutes franchissent le seuil de détection T au même instant. Les fenêtres d'amplitude C-D et B-C permettent d'isoler le train de potentiels d'action de formes S1 et S2 respectivement. La fenêtre A-B ne permet pas de séparer les formes S3 et S4. B, même formes de potentiels d'action avec deux ensembles de trois fenêtres temps-amplitude séparés d'un intervalle de temps choisi. Chaque fenêtre admet le(s) potentiel(s) d'action qui la traverse(nt) uniquement. La fenêtre A-B ne peut séparer S3 et S4, en revanche, la fenêtre A'-B' située peu après isole S3 de S4.

3.1.2 Fenêtre d'amplitude et fenêtre temporelle

L'étape suivante dans la sophistication de la procédure de tri ajoute à ce critère d'amplitude un critère sur le décours temporel du PA : pour être sélectionné, un PA doit franchir une première fois la limite inférieure d'amplitude fixée ; puis il doit le franchir une deuxième fois, et ce dans un intervalle de temps déterminé et tout en restant en dessous de la limite supérieure d'amplitude choisie. En d'autres termes, la durée du PA dans l'intervalle d'amplitudes choisi est aussi contrainte et doit appartenir à une fenêtre de temps prédéterminée. L'expérimentateur choisit donc une double fenêtre, d'amplitude et de temps, pour chaque neurone (*time-amplitude window discriminator*). Un catalogue complet des variations sur ce thème et des multiples dispositifs électroniques conçus et utilisés pour réaliser ces opérations lors de l'acquisition des données a été dressé par Schmidt (1984a). La Fig. 3.2B donne un exemple d'utilisation de deux ensembles de fenêtres temps-amplitude pour discriminer les formes de potentiels d'action plus finement que ne peuvent le faire de simples fenêtres d'amplitude. Ce type de procédure n'est en aucun cas obsolète : il est encore largement employé, sous une forme ou sous une autre, par les tenants de l'approche une électrode/un neurone évoquée dans l'introduction générale.

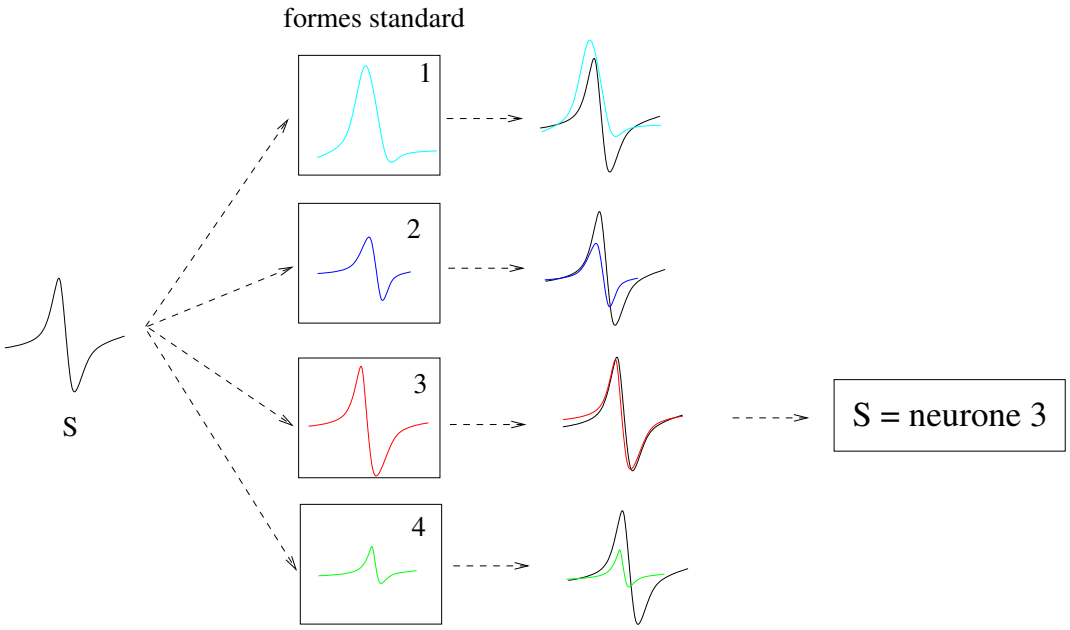

Figure 3.3: **Principe du *template-matching*.** Lorsque l'on dispose d'un ensemble de K formes standard de potentiel d'action (ici, $K = 4$) correspondant à K neurones d'origine, on compare chacune d'elles à tout potentiel d'action S des données. On attribue à S la forme la plus proche, au sens défini dans le texte.

3.1.3 Comparaison à une forme prototypique (*template matching*)

La troisième méthode utilisée dès les débuts du *spike-sorting* compare les formes des PAs détectés à des formes prototypiques (ou encore "standard"), appelées *templates* en Anglais. Contrairement aux dispositifs électroniques précédents (*hardware oriented*), il s'agit ici de la première méthode numérique (*software oriented*) développée pour effectuer le *spike-sorting*. La distance d'un PA à une forme de référence est définie comme la somme des carrés des différences sur les d points d'échantillonnage définissant un PA. A chaque PA est attribuée la forme standard qui minimise cette distance (Fig. 3.3). Tous les PAs associés à une même forme standard sont considérés comme émis par le même neurone. La difficulté de cette méthode, connue en Anglais sous le nom de *template matching*, est évidemment de définir les formes standard de PAs auxquelles comparer les PAs enregistrés. Les premières méthodes de *template matching* nécessitaient le choix d'un petit nombre de formes standard de PAs par l'expérimentateur lui-même (Schmidt, 1984b). Les méthodes de *clustering*, exposées dans les deux prochaines sections, permettent de choisir ces formes prototypiques automatiquement.

3.2 Extraction des caractéristiques (*feature extraction*)

3.2.1 Représentation initiale des PAs

La forme de chaque PA détecté, notée $\mathbf{x}^j$, est décrite par son échantillonnage numérique, c'est-à-dire par un vecteur de dimension d : $\mathbf{x}^j = (x_1^j, ..., x_d^j)$. Compte tenu des fréquences d'échantillonnage généralement adoptées (typiquement $15\,kHz$) et de la fenêtre de temps nécessaire pour décrire entièrement un PA (typiquement $4\,ms$), la dimension d de l'espace de départ est de l'ordre de 60. Lorsqu'un PA est décrit par sa forme sur quatre sites d'enregistrement (dans le cas des tétrodes), cette dimension est de l'ordre de 240 (mise bout à bout des quatre vecteurs). Les données initiales soumises au *spike-sorting* sont donc constituées d'un ensemble de n vecteurs $\{\mathbf{x}^j\}\,(j = 1, \ldots, n)$ de dimension d représentant les n PAs détectés. De plus, afin de minimiser la variabilité des formes de PAs, la position du maximum de tous ces vecteurs, c'est-à-dire des pics des PAs, est la même.

Si l'on fait l'hypothèse qu'un neurone donné émet un PA de forme fixe à laquelle s'ajoute un bruit indépendant, les PAs de ce neurone forment, dans cet espace de grande dimension, un nuage de points (en Anglais : *cluster*). Les PAs d'un autre neurone formeront idéalement un autre nuage de points et l'objectif du *spike-sorting* est d'isoler ces différents nuages (*clustering*) et de déterminer à quel nuage (*i.e* neurone) appartient chaque point (*i.e* PA). Néanmoins, la grande dimension de l'espace de départ rend cette étape de *clustering* très coûteuse, voire inextricable. Il est possible d'être beaucoup plus économe dans la représentation des PAs et de réduire celle-ci à quelques paramètres discriminants.

La première étape du *spike-sorting* consiste donc à trouver un espace de représentation des PAs de dimension réduite $m < d$ avant d'effectuer le *clustering* proprement dit (Fig. 3.1). Cette réduction de dimensionalité doit être accompagnée d'une perte minimale d'information par rapport à la description complète de départ. Le problème est donc de trouver - d'extraire - un petit nombre de caractéristiques des formes de PAs qui soient les plus discriminantes possibles. Il s'agit de déterminer la représentation minimale qui conserve le mieux la séparation des PAs. Pour ce faire, diverses méthodes ont été proposées dont voici les plus répandues. Les méthodes décrites dans cette section sont autant de manières de réaliser cette étape d'extraction des caractéristiques ; elles constituent en ce sens différentes alternatives possibles pour réaliser la transformation d'un PA $\mathbf{x}^j = (x_1^j, ..., x_d^j)$ en un vecteur $\mathbf{z}^j = (z_1^j, ..., z_m^j)$, $m < d$. Ces notations seront utilisées dans toute la suite : $\mathbf{x}^j$ et d pour l'espace de départ, $\mathbf{z}^j$ et m pour l'espace réduit des caractéristiques.

3.2.2 Choix *a priori* des paramètres caractéristiques des PAs

La méthode la plus simple est de ne décrire chaque PA que par un petit nombre de grandeurs cruciales qui le caractérisent, telles que son amplitude au pic, son amplitude pic à pic ou sa largeur à mi-hauteur (Schmidt, 1984b). Pour les enregistrements avec tétrodes, et plus généralement pour les enregistrements avec électrodes multisites, on

peut par exemple ne conserver que les amplitudes au pic des PAs sur quatre sites pour les décrire. Des paramètres temporels tels que le temps entre le pic et le minimum du PA, ou le temps entre le pic et le prochain zéro, sont également souvent très pertinents pour caractériser la forme d'un PA. On réduit ainsi la représentation des PAs à quelques paramètres choisis *a priori* ($m = 2, 3$ ou 4 typiquement). Si ces paramètres sont effectivement discriminants, les PAs forment, dans cet espace de petite dimension, des nuages de points qu'un algorithme de *clustering* permet d'isoler.

3.2.3 Sélection des points d'échantillonnage pertinents : *Reduced Feature Set* (RFS)

On peut aussi faire le choix de représenter un PA par un sous-ensemble de m points d'échantillonnage, choisis parmi les d points de la description initiale. Dans cette approche, il s'agit de sélectionner les m points d'échantillonnage (appelés *Reduced Feature Set*, ou RFS, dans la littérature) les plus discriminants dans la forme des PAs enregistrés. Cette sélection se fait à l'aide d'un ensemble $\mathcal{S}$ prédéfini de PAs pris dans l'enregistrement. Le point d'échantillonnage le plus discriminant est évidemment celui dont la variance est maximale sur $\mathcal{S}$. Dinning et Sanderson (1983), Salganicoff et al. (1988) ont décrit des algorithmes pour sélectionner des points d'échantillonnage supplémentaires. Les procédés de sélection décrits par ces auteurs favorisent naturellement les points d'échantillonnage ayant une grande variance sur $\mathcal{S}$; ils favorisent également une répartition uniforme de ces points sur les régions de grande variance. Tout nouveau PA de l'enregistrement est ensuite décrit par ce seul sous-ensemble de m points d'échantillonnage sélectionnés.

Dans le cas d'enregistrements obtenus avec des tétrodes, la sélection du point d'échantillonnage de plus grande variance sur chaque site fournit tout naturellement les quatre points les plus discriminants de la forme des PAs enregistrés sur ces quatres sites.

3.2.4 Analyse en composante principale (PCA)

L'une des méthodes les plus répandues dans la réduction de la dimensionalité est sans aucun doute l'analyse en composantes principales (*Principal Component Analysis*, ou PCA)(Glaser and Marks, 1968 ; Eggermont, 1983). Il s'agit d'une transformation linéaire des PAs $\mathbf{x}^j = (x_1^j, ..., x_d^j)$ dans l'espace de départ en vecteurs $\mathbf{z}^j = (z_1^j, ..., z_m^j)$ dans un espace de dimension $m < d$. L'idée générale de cette approche est de trouver les m directions orthogonales de l'espace de départ, selon lesquelles les variances d'un ensemble $\mathcal{S}$ prédéfini de n PAs pris dans l'enregistrement sont les plus grandes. Ces directions, appelées *directions* - ou *composantes* - *principales*, sont de ce fait les plus discriminantes pour l'ensemble $\mathcal{S}$. Tous les PAs enregistrés sont ensuite systématiquement projetés sur ces m directions principales.

Les m composantes principales orthogonales sont celles qui minimisent le carré de l'erreur moyenne de la représentation des PAs de l'ensemble $\mathcal{S}$. Soit $\mathbf{x}^j = (x_1^j, ..., x_d^j)$ l'un des PAs de cet ensemble. Sans perte de généralité, on peut représenter $\mathbf{x}^j$ comme une combinaison linéaire de d vecteurs orthonormés $\mathbf{u}_i$:

$$\mathbf{x}^j = \sum_{i=1}^{d} z_i^j \mathbf{u}_i \tag{3.1}$$

Il s'agit ici d'une simple rotation du système de coordonnées par rapport à la représentation initiale du vecteur $\mathbf{x}^j$. Les coordonnées z_i^j du vecteur $\mathbf{x}^j$ dans ce nouveau système de coordonnées sont données par :

$$z_i^j = \mathbf{u}_i^T \mathbf{x}^j \tag{3.2}$$

L'objectif est de ne conserver qu'un sous-ensemble $m < d$ des vecteurs de base $\mathbf{u}_i$, de façon à représenter chaque vecteur $\mathbf{x}^j$ par m coordonnées z_i^j uniquement. Les coefficients restants sont remplacés par des coefficients constants b_i identiques pour tous les vecteurs, de sorte que chaque vecteur $\mathbf{x}^j$ est approximé par une expression de la forme :

$$\widetilde{\mathbf{x}^j} = \sum_{i=1}^{m} z_i^j \mathbf{u}_i + \sum_{i=m+1}^{d} b_i \mathbf{u}_i \tag{3.3}$$

Il s'agit bien ici d'une réduction de dimensionalité puisque le vecteur $\mathbf{x}^j$, qui contenait d degrés de liberté, est désormais approximé par un vecteur ayant $m < d$ degrés de liberté. Comment déterminer la base de vecteurs orthonormés $\mathbf{u}_i$? De façon naturelle, on choisit la base qui fournit la meilleure approximation, en moyenne, des n vecteurs $\{\mathbf{x}^j\}$ de notre ensemble $\mathcal{S}$ de PAs. L'erreur introduite par la réduction de dimensionalité pour un vecteur $\mathbf{x}^j$ de cet ensemble $\mathcal{S}$ s'écrit :

$$\mathbf{x}^j - \widetilde{\mathbf{x}^j} = \sum_{i=m+1}^{d} (z_i^j - b_i)\mathbf{u}_i \tag{3.4}$$

On définit la meilleure approximation des n vecteurs de l'ensemble $\mathcal{S}$ comme étant celle qui minimise la somme des carrés des erreurs sur cet ensemble. On minimise donc la quantité E_m suivante :

$$E_m = \frac{1}{2} \sum_{j=1}^{n} \|\mathbf{x}^j - \widetilde{\mathbf{x}^j}\|^2 = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=m+1}^{d} (z_i^j - b_i)^2 \tag{3.5}$$

En annulant la dérivée de E_m par rapport à b_i, on obtient :

$$b_i = \frac{1}{n} \sum_{kj=1}^{n} z_i^j = \mathbf{u}_i^T \overline{\mathbf{x}} \tag{3.6}$$

où $\overline{\mathbf{x}}$ est le vecteur moyen de $\mathcal{S}$:

$$\overline{\mathbf{x}} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}^j \tag{3.7}$$

En utilisant 3.2 et 3.6, on peut réécrire E_m :

$$E_m = \frac{1}{2} \sum_{i=m+1}^{d} \sum_{j=1}^{n} \left(\mathbf{u}_i^T \left(\mathbf{x}^j - \overline{\mathbf{x}} \right) \right)^2 = \frac{1}{2} \sum_{i=m+1}^{d} \mathbf{u}_i^T \mathbf{\Sigma} \mathbf{u}_i \tag{3.8}$$

où $\mathbf{\Sigma}$ est la matrice de covariance de l'ensemble $\mathcal{S}$ de vecteurs $\{\mathbf{x}^j\}$, donnés par :

$$\mathbf{\Sigma} = \sum_j \left(\mathbf{x}^j - \overline{\mathbf{x}} \right) \left(\mathbf{x}^j - \overline{\mathbf{x}} \right)^T \tag{3.9}$$

Un recours aux multiplicateurs de Lagrange, que je ne détaille pas, permet de minimiser E_m par rapport à la base orthonormée $\{\mathbf{u}_i\}$ (Bishop, 1995). Ce minimum est atteint lorsque les vecteurs de base $\mathbf{u}_i$ satisfont :

$$\mathbf{\Sigma} \mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{3.10}$$

Le minimum de E_m est donc atteint avec les vecteurs propres de la matrice de covariance des vecteurs de l'ensemble $\mathcal{S}$. Cette matrice étant réelle et symétrique, ses vecteurs propres peuvent êtres choisis orthonormés. En remplaçant 3.10 dans 3.8, et en utilisant l'orthogonalité des vecteurs $\mathbf{u}_i$ on obtient la valeur de E_m au minimum :

$$E_m = \frac{1}{2} \sum_{i=m+1}^{d} \lambda_i \tag{3.11}$$

Ainsi, l'erreur E_m est minimisée si l'on choisit les $d-m$ plus petites valeurs propres (et les vecteurs propres correspondants) de la matrice de covariance des vecteurs de l'ensemble $\mathcal{S}$, comme celles à ne pas considérer dans la représentation des vecteurs de $\mathcal{S}$. Autrement dit, on minimise l'erreur E_m en représentant les vecteurs $\mathbf{x}^j$ de $\mathcal{S}$ par leur projection sur les m vecteurs propres orthonormés de leur matrice de covariance qui ont les plus grandes valeurs propres. Ces valeurs propres sont égales aux variances respectives des vecteurs $\mathbf{x}^j$ selon ces directions. La PCA choisit donc de représenter les vecteurs $\mathbf{x}^j$ de $\mathcal{S}$ selon les directions orthogonales de plus grande variance pour cet ensemble : les directions principales.

En pratique, l'algorithme d'une PCA commence par calculer le vecteur moyen $\overline{\mathbf{x}}$ des vecteurs $\mathbf{x}^j$ de $\mathcal{S}$, desquels il est ensuite soustrait. L'algorithme calcule alors la matrice de covariance de ces vecteurs, ainsi que leurs vecteurs propres et leurs valeurs propres. Les vecteurs propres correspondant aux m plus grandes valeurs propres sont conservées et les vecteurs $\mathbf{x}^j$ de $\mathcal{S}$ y sont projetés pour obtenir les vecteurs $\mathbf{z}^j = (z_1^j, ..., z_m^j)$ dans l'espace à m dimensions. En règle générale, 3 ou 4 composantes principales sont suffisantes pour rendre compte de plus de 90% de la variance de $\mathcal{S}$. Une fois ces directions principales déterminées sur l'ensemble $\mathcal{S}$, tout nouveau PA $\mathbf{x}$ proposé y est ensuite systématiquement projeté.

3.2.5 Transformée en ondelettes

Récemment, plusieurs méthodes fondées sur la transformée en ondelettes des PAs ont été proposées (Letelier et Weber, 2000 ; Hulata et al., 2002 ; Quian Quiroga et al.,

2004). Le principe de ces méthodes est de décomposer chaque PA en une somme finie d' "ondelettes" et de ne garder qu'un sous-ensemble d'entre elles. Dans cette analyse, un PA est représenté par le vecteur dont les composantes sont les m coefficients des ondelettes de la décomposition qui sont conservés. Cette section décrit le principe général de cette analyse. L'objectif n'est pas ici d'exposer tous les développements mathématiques de la théorie. Il s'agit plutôt de comprendre en quoi la transformée en ondelettes des PAs peut être utile à leur classification.

Transformée en ondelettes continue

Une ondelette est une fonction $\psi(t)$, intégrable et suffisamment oscillante pour être d'intégrale nulle. L'analyse continue par ondelettes lui associe une famille de copies d'elle-même, translatées et dilatées :

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}) \tag{3.12}$$

Les paramètres a et b sont dits paramètres d'échelle et de translation respectivement.

La transformée en ondelettes continue d'une fonction de carré intégrable $f \in L^2$ est définie par :

$$W_\psi f(a,b) = \int_{-\infty}^{+\infty} \psi^*_{a,b}(t) f(t) dt \tag{3.13}$$

La transformée inverse représente le signal f comme une superposition d'ondelettes translatées et dilatées :

$$f(t) = \frac{1}{C_\psi} \int_{a=0}^{+\infty} \int_{b=-\infty}^{+\infty} W_\psi f(a,b)\psi_{a,b}(t) \frac{dadb}{a^2} \tag{3.14}$$

où C_ψ ne dépend que de l'ondelette $\psi(t)$ et est donné par $C_\psi = \int_0^{+\infty} \frac{|\hat{\psi}(\omega)|}{\omega} d\omega$, avec $\hat{\psi}(\omega) = \int_{-\infty}^{+\infty} \psi(t) \exp(-i\omega t) d\omega$ la transformée de Fourier de ψ.

Transformée en ondelettes discrète

Très souvent, c'est la transformée en ondelettes discrète du signal qui est utilisée. Les paramètres a et b prennent les valeurs discrètes $a_j = 2^{-j}$et $b_{j,k} = 2^{-j}k$ $(j,k \in \mathbb{Z})$. La famille d'ondelettes s'écrit alors :

$$\psi_{j,k}(t) = \frac{1}{2^{j/2}}\psi(2^j t - k) \tag{3.15}$$

Cette famille forme une base orthonormée de L^2. La transformée en ondelettes discrète représente la fonction f (un PA dans le cadre qui nous occupe) comme une combinaison linéaire des fonctions $\psi_{j,k}$:

$$f(t) = \sum_{j,k} c_{j,k}\psi_{j,k}(t) \tag{3.16}$$

où les coefficients $c_{j,k}$ sont donnés, comme en 3.13, par :

$$c_{j,k} = \int_{-\infty}^{+\infty} \psi^*_{j,k}(t) f(t) dt \tag{3.17}$$

Les coefficients d'ondelettes $c_{j,k}$ fournissent donc une représentation alternative de la fonction f, sans perte d'information.

L'analyse en ondelettes fournit plusieurs choix possibles pour l'ondelette ψ (dite *mother wavelength* en Anglais), moyennant certaines contraintes mathématiques (continuité, support compact, moyenne nulle). C'est d'ailleurs l'enjeu de la transformée en ondelettes que de choisir une ondelette adaptée à f pour qu'un minimum de coefficients $c_{j,k}$ soient non nuls. Le signal f représenté par une seule variable indépendante, le temps, est donc maintenant décrit par une fonction de deux variables indépendantes, j et k. L'index j change le comportement de $\psi_{j,k}$ dans l'espace des fréquences, tandis que k translate l'ondelette le long de l'axe des temps.

Sélection des coefficients $c_{j,k}$

Une fois obtenue la décomposition en ondelettes d'un ensemble $\mathcal{S}$ de PAs, un sous-ensemble de m coefficients $c_{j,k}$ est sélectionné. Ces coefficients caractérisent les formes des PAs à différentes échelles de fréquences et à différents temps. Il s'agit de ne garder, pour représenter un PA, que les quelques coefficients qui séparent le mieux les différentes classes de PAs. De ce point de vue, les coefficients les plus discriminants doivent avoir une distribution multimodale sur l'ensemble $\mathcal{S}$ de PAs (plusieurs classes de PAs). La sélection des coefficients dont la distribution dévie le plus d'une distribution normale (comme il est possible d'en juger par un test de Kolmogorov-Smirnov) s'avère particulièrement pertinente (Quian Quiroga et al., 2004).

Fondées sur les principes d'analyse ci-dessus, les méthodes proposées dans le cadre du *spike-sorting* diffèrent selon le type de décomposition en ondelettes effectuée (transformée en ondelettes simple, Letelier et Weber, 2000 ; transformée en paquets d'ondelettes, Hulata et al., 2002), et selon le mode de sélection des coefficients pour la représentation finale d'un PA (Quian Quiroga et al., 2004 ; Hulata et al., 2002). La très grande capacité de cette méthode à séparer les classes de PAs a été démontré par Quian Quiroga et al. (2004). Ces auteurs montrent que la séparation de trois classes de PAs de formes très proches (en particulier de même amplitude) réalisées par les trois meilleurs coefficients d'ondelettes de leur sélection sépare beaucoup mieux ces trois classes que ne le font les trois premières composantes principales.

3.3 Délimitation des nuages de points (*clustering*)

Une fois l'étape d'extraction de caractéristiques terminée, on dispose de n vecteurs $\{\mathbf{z}^j\}$ de dimension m (typiquement $m < 10$) répartis en nuages (*clusters*) plus ou moins distincts, qu'il s'agit maintenant de séparer (*clustering*) (Fig. 3.1). Il faut souligner ici l'importance de l'étape d'extraction : si les caractéristiques extraites sont peu discriminantes, les PAs seront peu séparés dans l'espace de ces caractéristiques et les nuages

peu marqués. Quelle que soit la performance de la méthode de *clustering* utilisée, le *spike-sorting* sera mauvais.

Là encore, toute une gamme de méthodes sont disponibles pour réaliser cette étape. Les paragraphes qui suivent décrivent les méthodes les plus utilisées dans le cadre du *spike-sorting*.

3.3.1 Délimitation visuelle

La méthode la plus répandue est sans doute la délimitation visuelle (*i.e* "à la main") des nuages. Les n points $\{\mathbf{z}^j\}$ dans l'espace de dimension m sont visualisés dans différents plans simultanément ; l'expérimentateur trace lui-même les contours des nuages dans ces différents plans. Cette méthode est longue, fastidieuse et sujette à de nombreuses erreurs (Harris et al., 2000), notamment lorsque les recouvrements entre nuages sont importants.

3.3.2 L'algorithme des *k-means*

L'isolation des nuages peut être effectuée automatiquement. Pour ce faire, l'algorithme le plus simple est l'algorithme dit des *k-means* (Hartigan, 1975). Ici, le nombre K de classes doit être fixé *a priori*. Cet algorithme cherche la partition des n points $\{\mathbf{z}^j\}$ en K sous-ensembles $\mathcal{C}_k$ disjoints, contenant n_k points ($k = 1, ..., K$), qui minimise la quantité J ci-dessous :

$$J = \sum_{k=1}^{K} \sum_{j \in \mathcal{C}_k} \| \mathbf{z}^j - \mu_k \|^2 \tag{3.18}$$

où μ_k est le vecteur moyen du sous-ensemble $\mathcal{C}_k$:

$$\mu_k = \frac{1}{n_k} \sum_{j \in C_k} \mathbf{z}^j \tag{3.19}$$

On commence par réaliser une première partition aléatoire des n PAs en K sous-ensembles. Le vecteur moyen μ_k de chaque sous-ensemble $\mathcal{C}_k$ est calculé à partir de cette partition initiale. Puis on réassigne chaque point au sous-ensemble dont le vecteur moyen est le plus proche de ce point, au sens de la distance euclidienne. On re-calcule alors les vecteurs moyens de ces nouveaux sous-ensembles. Cette procédure est répétée jusqu'à ce qu'il n'y ait plus de changement dans la partition effectuée (Lloyd, 1982).

En utilisant la distance euclidienne dans l'assignation d'un point à un nuage, on fait ici l'hypothèse que les nuages sont sphériques. Cet algorithme est donc nettement moins performant, voire inappropié, quand les nuages ne le sont pas ou quand leurs recouvrements sont importants.

3.3.3 L'algorithme des *fuzzy c-means*

Il s'agit d'une extension de l'algorithme précédent. Ici, un point $\mathbf{z}^j$ des données est considéré comme appartenant à plusieurs sous-ensembles simultanément, avec un certain *degré d'attribution* $t_{jk} \in [0, 1]$ pour chaque sous-ensemble $\mathcal{C}_k$. Là encore, le

nombre K de sous-ensemble $\mathcal{C}_k$ doit être fixé *a priori*. Cet algorithme est fondé sur la minimisation de la quantité J_s (Bezdek, 1981) :

$$J_s = \sum_{j=1}^{n} \sum_{k=1}^{K} t_{jk}^s \parallel \mathbf{z}^j - \mu_k \parallel^2, 1 \leq s < \infty \tag{3.20}$$

où s est un nombre réel plus grand que 1, et μ_k est le vecteur moyen du sous-ensemble $\mathcal{C}_k$. Plus s est grand, plus le caractère "diffus" (*fuzzy*) de la partition est important. On a bien évidemment :

$$\sum_{k=1}^{K} t_{jk} = 1, \forall j$$

$$0 < \sum_{j=1}^{n} t_{jk} < N, \forall k$$

L'algorithme commence avec une partition initiale $\mathbf{t}^{(0)} = [t_{jk}^{(0)}]$ aléatoire. La minimisation de J_s se fait de façon itérative. L'itération l de l'algorithme est constituée de deux étapes :

– Calcul des centres $\mu_k^{(l)}$ des sous-ensembles $\mathcal{C}_k$, étant donné $\mathbf{t}^{(l-1)}$:

$$\mu_k^{(l)} = \frac{\sum_{j=1}^{n} (t_{jk}^{(l)})^s \mathbf{z}^j}{\sum_{j=1}^{n} (t_{jk}^{(l)})^s} \tag{3.21}$$

– Calcul de la partition $\mathbf{t}^{(l)}$, étant donné les $\mu_k^{(l)}$:

$$t_{jk}^{(l)} = \sum_{r=1}^{K} \left(\frac{\|\mathbf{z}^j - \mu_k^{(l)}\|}{\|\mathbf{z}^j - \mu_r^{(l)}\|} \right)^{-\frac{2}{s-1}} \tag{3.22}$$

On peut par exemple arrêter l'algorithme lorsque $max_{jk}\{|t_{jk}^{(l+1)} - t_{jk}^{(l)}|\} < \epsilon$. Cette procédure converge vers un minimum local de J_s. Pour obtenir une classification forcée des n points $\{\mathbf{z}^j\}$ on peut choisir d'associer à chaque $\mathbf{z}^j$ le sous-ensemble $\mathcal{C}_k$ dont le degré d'attribution $t_{jk}^{(derniere)}$ est le plus grand à la dernière itération.

3.3.4 Les modèles de mélanges et l'algorithme d'*Expectation-Maximization* (EM)

Modèles de mélanges

Dans un modèle de mélange, chacun des n points $\{\mathbf{z}^j\}$ des données $\mathbf{z}$ est issu d'une loi de densité :

$$p(\mathbf{z}^j|\theta) = \sum_{k=1}^{K} \pi_k \phi(\mathbf{z}^j|\mathbf{a}_k) \tag{3.23}$$

où ϕ est la densité, de paramètres $\mathbf{a}_k$, de la composante k du mélange, π_k est la proportion de cette composante ($\pi_k \in [0,1]$, $\sum_k \pi_k = 1$) et K est le nombre de composantes dans le mélange. Ce nombre K est là encore choisi *a priori*. Dans 3.23, on a posé $\theta = (\pi_1, ..., \pi_{K-1}, \mathbf{a}_1, ..., \mathbf{a}_K)$. Le but est ici d'associer chaque point $\mathbf{z}^j$ des données à l'une des composantes k du mélange, c'est-à-dire d'estimer la partition des données. Ceci peut être fait en estimant les paramètres θ du modèle de mélange, par exemple en maximisant la fonction de vraisemblance (ou son logarithme).

Sous l'hypothèse d'indépendance des n PAs $\mathbf{z}^j$, la probabilité des données $\mathbf{z}$ étant donné le modèle, la *vraisemblance*, s'écrit :

$$P(\mathbf{z}|\theta) = \prod_{j=1}^{n} p(\mathbf{z}^j|\theta) \tag{3.24}$$

Le logarithme de la vraisemblance, noté $\mathcal{L}$, est donc donné par :

$$\mathcal{L} = \sum_{j=1}^{n} \ln(\sum_{k=1}^{K} \pi_k \phi(\mathbf{z}^j|\mathbf{a}_k)) \tag{3.25}$$

La log-vraisemblance $\mathcal{L}$ peut être maximisée avec l'algorithme dit d'*Expectation-Maximization* (EM) (Dempster et al., 1977). Cet algorithme a été introduit en électrophysiologie par Ling et Tohlburst (1983). Il est décrit plus bas.

En règle générale, le modèle de mélange considéré est un modèle de mélange gaussien multivarié (Pouzat et al., 2002). Les densités ϕ sont de la forme :

$$\phi(\mathbf{z}^j|\mathbf{a}_k) = \phi(\mathbf{z}^j|\mathbf{u}_k, \mathbf{\Sigma}_k) = \frac{1}{(2\pi)^{m/2} \cdot |\Sigma_k|} \exp\left(-\frac{1}{2}(\mathbf{z}^j - \mathbf{u}_k)^T \mathbf{\Sigma}_k^{-1} (\mathbf{z}^j - \mathbf{u}_k)\right) \tag{3.26}$$

où m est, rappelons-le, la dimension de l'espace considéré. $\mathbf{u}_k$ et $\mathbf{\Sigma}_k$ sont respectivement le vecteur moyen et la matrice de covariance de la densité $\phi(.|\mathbf{a}_k)$. L'hypothèse sous-jacente est donc que le bruit indépendant qui s'ajoute aux PAs est gaussien.

L'algorithme EM

Cet algorithme itératif commence avec une valeur arbitraire $\theta^{(0)}$ des paramètres θ du modèle de mélange et construit une séquence d'estimations $\theta^{(1)}, ..., \theta^{(l)}$ de θ telles que la log-vraisemblance $\mathcal{L}$ croît de façon monotone. L'itération l de l'algorithme est divisée en deux étapes :

- Etape E (*Expectation*) : calcul de la matrice $\mathbf{t}^{(l)} = [t_{jk}^{(l)}]$ des probabilités conditionnelles $t_{jk}^{(l)} = t_k(\mathbf{z}^j|\theta^{(l-1)})$, où $t_k(\mathbf{z}^j|\theta^{(l-1)})$ est la probabilité que le point $\mathbf{z}^j$ vienne de la composante k du mélange.
- Etape M (*Maximization*) : calcul des paramètres $\theta^{(l)}$ du mélange, avec les probabilités $\mathbf{t}^{(l)}$ fixées.

L'algorithme estime donc la partition des points à partir des valeurs présentes des paramètres des densités du mélange et réciproquement. L'arrêt de l'algorithme peut se faire en fixant le nombre d'itérations ou en utilisant un seuil minimal de croissance de $\mathcal{L}$ par itération. Le EM ne converge pas nécessairement vers l'estimateur du maximum de

vraisemblance et le résultat dépend de la valeur initiale $\theta^{(0)}$ choisie pour les paramètres. Il est donc indispensable de lancer l'algorithme plusieurs fois, avec différentes valeurs initiales, et de ne garder que l'estimation obtenue pour l'essai ayant conduit à la valeur maximale de $\mathcal{L}$.

Dans le cas du mélange gaussien multivarié, si on note $n_k^{(l)} = \sum_{j=1}^{n} t_{jk}^{(l)}$ la taille du sous-ensemble k, l'étape M est donnée par :

$$\hat{\pi}_k^{(l)} = \frac{n_k^{(l)}}{n} \tag{3.27}$$

$$\hat{\mu}_k^{(l)} = \frac{1}{n_k^{(l)}} \sum_{j=1}^{n} t_{jk}^{(l)} \mathbf{z}^j \tag{3.28}$$

$$\hat{\mathbf{\Sigma}}_k^{(l)} = \frac{1}{n_k^{(l)}} \sum_{j=1}^{n} t_{jk}^{(l)} (\mathbf{z}^j - \hat{\mu}_k^{(l)})(\mathbf{z}^j - \hat{\mu}_k^{(l)})^T \tag{3.29}$$

L'étape E de l'itération de l'algorithme fournit la partition "non forcée" $\mathbf{t}^{(l)} = [t_{jk}^{(l)}]$ des n points $\{\mathbf{z}^j\}$ pour l'itération l. On force la classification finale en associant chaque point $\mathbf{z}^j$ au nuage k dont la probabilité $t_{jk}^{(derniere)}$ à la dernière itération est la plus grande.

Le modèle de mélange gaussien multivarié n'est pas le seul à avoir été considéré dans le cadre du *spike-sorting*. Le EM a également été utilisé dans le cadre d'un modèle de mélange multivarié de distributions t (Shoham et al., 2003). Pour les données analysées par ces auteurs, la distribution t multivariée rend mieux compte de la statistique des formes de PA.

Le EM peut être performant si la statistique du bruit n'est pas trop éloignée du modèle considéré et si les recouvrements entre nuages ne sont pas trop importants. Dans le cas contraire, son utilisation est inadéquate. Comme les deux algorithmes précédents (*k-means* et *fuzzy c-means*), cet algorithme travaille avec un nombre de neurones K fixé par l'expérimentateur. En ce sens, ces méthodes sont dites supervisées.

3.3.5 *Superparamagnetic clustering*

Un algorithme de *clustering* superparamagnétique, fondé sur la simulation d'interactions entre les points $\mathbf{z}^j$ des données et ses L plus proches voisins, a récemment été proposé (Quian Quiroga et al., 2004). Il s'agit d'un algorithme non supervisé qui ne nécessite pas la détermination *a priori* du nombre de nuages.

Plus précisément, les auteurs implémentent un algorithme introduit par Wolf pour la simulation des modèles d'Ising à température critique utilisant un modèle d'interaction entre deux points "proches voisins" $\mathbf{z}^i$ et $\mathbf{z}^j$. Un point $\mathbf{z}^i$ possède L proches voisins : les L points dont les distances euclidiennes à ce point sont les plus petites. L'interaction entre deux proches voisins, notée J_{ij} est une fonction exponentiellement décroissante du carré de la distance euclidienne entre ces deux points. Initialement, chaque point $\mathbf{z}^j$ des données est dans un état s^j, parmi q possibles. Un point $\mathbf{z}^i$ est choisi aléatoirement et son état s^i est remplacé par un nouveau, s^{new}, pris aléatoirement entre 1 et q. On

essaie ensuite de changer l'état des proches voisins de $\mathbf{z}^i$. La probabilité qu'un proche voisin $\mathbf{z}^j$ de $\mathbf{z}^i$ passe également dans l'état s^{new} est donnée par :

$$p_{ij} = 1 - \exp\left(-\frac{J_{ij}}{T}\delta_{s^i,s^j}\right) \tag{3.30}$$

où T est la température (il s'agit d'une température formelle introduite par analogie avec les systèmes magnétiques en physique statistique) et δ_{s^i,s^j} est le symbole de Kroenecker ($\delta_{s^i,s^j} = 1$ si $s^i = s^j$ et 0 sinon). Ainsi, seuls les proches voisins qui étaient dans le même état que $\mathbf{z}^i$ sont candidats à un changement de leur état vers l'état s^{new}. Un proche voisin de $\mathbf{z}^i$ ne peut changer d'état qu'une seule fois au cours de cette itération de l'algorithme. Au cours de cette même itération, on essaie de changer, selon la même probabilité 3.30, les états des proches voisins des proches voisins de $\mathbf{z}^i$ qui ont changé d'état (*i.e* les proches voisins "au carré"), jusqu'à ce que plus aucun changement ne soit possible. On sélectionne alors au hasard un autre point $\mathbf{z}^{i'}$ et on réitère la procédure de mise à jour en cascade des états des proches voisins pour ce nouveau point. On réalise N itérations selon cette procédure, à différentes températures. Les points qui sont proches, au sens de la distance euclidienne (J_{ij} grand et p_{ij} proche de 1), ont tendance à changer d'état simultanément et correspondent à un nuage. Il suffit donc de quantifier la corrélation $\left\langle \delta_{s^i,s^j} \right\rangle$ et d'associer $\mathbf{z}^i$ et $\mathbf{z}^j$ dans un même nuage si $\left\langle \delta_{s^i,s^j} \right\rangle \geq \theta$, où θ est un seuil prédéterminé.

L'effet de la température sur cette dynamique est évident. A basse température, la probabilité de changer simultanément les états des proches voisins est grande, quelle que soit la force de leur interaction (*i.e* quel que soit leur éloignement) ; tous les points des données ont tendance à changer d'état en même temps. C'est analogue à la phase ferromagnétique d'un verre de spins. Tous les points sont considérés comme appartenant au même nuage. A l'inverse, à haute température, la probabilité de changer simultanément les états de deux proches voisins est faible ; les points des données changent d'état aléatoirement, quelle que soit la force de leurs interactions. Cela correspond à la phase paramagnétique d'un verre de spin. Les données sont alors partagées en de nombreux nuages ne contenant que quelques points chacun. Dans une gamme intermédiaire de température, seuls les points groupés en nuages changent d'état simultanément (phase superparamagnétique). En réalisant la simulation décrite ci-dessus sur tout une gamme de température, on peut explorer les trois phases mentionnées. Sur un certain intervalle de température, la taille des nuages, ainsi que son nombre, restent stables. C'est la phase superparamagnétique. On peut utiliser la partition des donnée obtenue dans cette phase pour déterminer le nombre de nuages (*i.e* neurones) et l'appartenance des points (*i.e* PAs) à ces nuages.

Cet algorithme de *clustering* présente l'avantage de ne pas nécessiter la fixation du nombre de nuages *a priori*. Cependant, la détermination de ce nombre, une fois les simulations à différentes températures réalisées, nécessite, en pratique, un choix de la part de l'expérimentateur. En effet, celui-ci doit déterminer la température T_0 de simulation qui a réalisé la bonne partition des données. Ce choix repose en grande partie sur la taille des nuages (*i.e* le nombre de points - PAs - par nuage) qui reflète la fréquence de décharge des neurones. La partition à retenir est celle dont les tailles de nuages correspondent aux fréquences de décharge que l'expérimentateur attend. On

voit donc poindre la faiblesse de la méthode dans le cas où des neurones déchargent à des fréquences très différentes : pour pouvoir autoriser une faible fréquence de décharge (un nuage contenant peu de points), l'expérimentateur doit considérer une simulation à haute température, interdisant de fait la présence de nuages contenant beaucoup de points (haute fréquence de décharge). Autrement dit, l'algorithme tend à homogénéiser les tailles des nuages à une température donnée. Cette méthode de délimitation des nuages est donc relativement faible lorsque des neurones de fréquences très différentes sont présents dans les données. Le comportement de cette méthode dans un tel cas reste à être démontré puisque les données simulées par les auteurs pour la tester comportent trois neurones ayant des décharges Poisson de même fréquence.

3.4 Réseaux de neurones artificiels

Les réseaux de neurones sont très utilisés dans de nombreux problèmes de reconnaissance de formes (Bishop, 1995). Ils le sont tout naturellement dans le domaine du *spike-sorting* (Jansen, 1990 ; Oghalai et al., 1994 ; Chandra et Optican, 1997 ; Garcia et al., 1998 ; Kim et Kim, 2000). Sans entrer dans le détail de la théorie mathématique sous-jacente, ni dans le détail d'une implémentation particulière, voici le principe de la classification réalisée par un réseau de neurones, dans le cadre qui nous occupe.

Les réseaux utilisés sont des réseaux multi-couches de neurones artificiels interconnectés. Ils sont composés d'une couche d'entrée, d'une ou plusieurs couches intermédiaires (dites "cachées") et d'une couche de sortie (Fig. 3.4). Le nombre de neurones dans la couche d'entrée est égal au nombre de points d'échantillonnage décrivant un PA (ou au nombre de caractéristiques utilisées pour décrire un PA dans le cas où l'on commence par réduire la dimension de sa représentation, Garcia et al., 1998) ; le nombre de neurones dans la couche de sortie est égal au nombre de neurones présents dans l'enregistrement, tel qu'il peut être déterminé *a priori* par l'expérimentateur. A une forme de PA donnée en entrée, le réseau devra associer un neurone de sortie. Le nombre de couches cachées et le nombre de neurones par couche cachée varient suivant les implémentations de la méthode.

La procédure de classification se fait en deux temps : apprentissage et classification. Les connexions entre neurones de deux couches successives sont caractérisées par leurs poids "synaptiques" qui peuvent être modifiés au cours de l'apprentissage. Une fois celui-ci terminé, les poids synaptiques des connexions sont maintenus fixés durant toute la classification. De façon générale, l'apprentissage a pour but de déterminer les poids du réseau tels que celui-ci associe à une entrée donnée (ici : un PA) la bonne sortie, *i.e* la sortie souhaitée par l'expérimentateur (ici : le bon neurone).

Durant l'étape d'apprentissage, l'expérimentateur apprend au réseau à reconnaître un certain nombre de formes de PAs. Pour ce faire, il choisit, dans l'enregistrement, un échantillon de PAs *déjà labélisés* pour constituer un ensemble d'apprentissage (*training set*) qu'il soumet au réseau. Il faut donc, d'une manière ou d'une autre, procéder à une première classification d'un ensemble de PAs de l'enregistrement. Cette classification peut se faire "à la main", par simple comparaison visuelle des formes des PAs sélectionnés pour constituer l'ensemble d'apprentissage (Jansen, 1990). Elle peut se faire au moyen

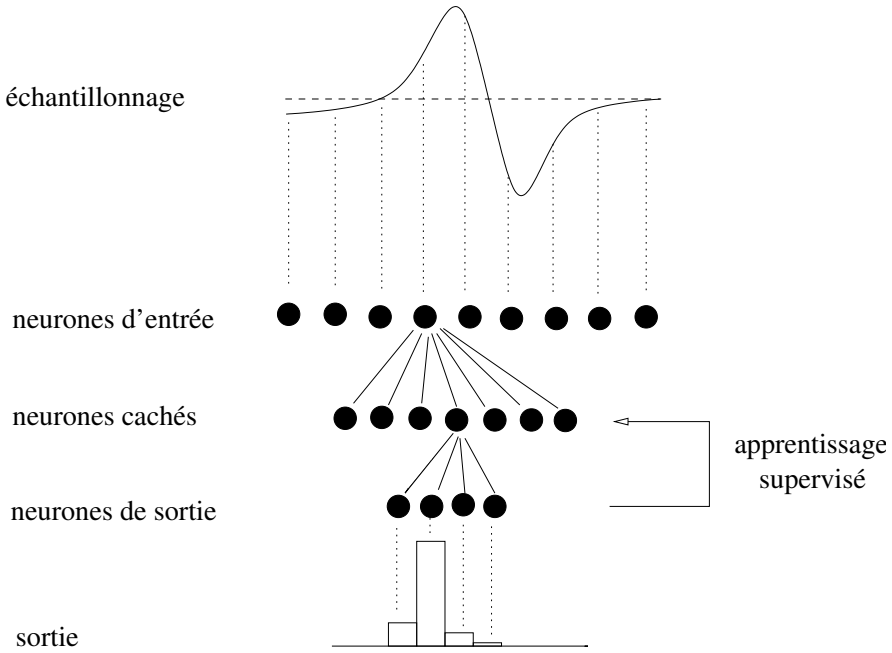

Figure 3.4: ***Spike-sorting* avec un réseau de neurones artificiels** (d'après Jansen, 1990). Les potentiels d'action sont digitalisés et l'amplitude de chaque point d'échantillonnage est passé à l'un des neurones de la couche d'entrée du réseau. Tous les neurones d'entrée sont connectés à tous les neurones de la couche cachée, et chaque connexion a un poids "synaptique" qui peut être ajusté. Tous les neurones cachés sont également connectés aux neurones de la couche de sortie. Il y a autant de neurones de sortie qu'il y a de classes de potentiels d'action. Dans un premier temps, le réseau apprend à reconnaître un jeu de potentiels d'action sélectionnés et labélisés (dit "ensemble d'apprentissage": apprentissage supervisé). Dans un second temps, on soumet au réseau les potentiels d'action non classifiés. Le niveau de sortie de chaque neurone de sortie mesure la similarité à l'ensemble d'apprentissage de chaque potentiel d'action présenté.

d'un algorithme de *clustering* (Chandra et Optican, 1997 ; Kim et Kim, 2000) effectué sur l'échantillon de PAs choisi pour être le sous-ensemble d'apprentissage. En tout état de cause, il est impératif que l'ensemble d'apprentissage contienne plusieurs PAs pour chaque classe de PAs identifiée par l'expérimentateur. Au cours de l'apprentissage, celui-ci soumet successivement à la couche d'entrée du réseau tous les PAs de l'ensemble d'apprentissage. Initialement, les poids des connexions du réseau sont aléatoires et la sortie du réseau (un nombre entre 0 et 1 pour chaque neurone de sortie) vis-à-vis de l'un des PAs en entrée est arbitraire. La différence entre la sortie réelle du réseau et la sortie souhaitée pour ce PA sert de signal d'erreur et est utilisée pour corriger les poids synaptiques, selon une "règle d'apprentissage", de sorte que l'erreur sera plus faible lorsque le même PA sera soumis à nouveau. Cette procédure est répétée jusqu'à ce que l'erreur du réseau soit minimisée sur tous les PAs de l'ensemble d'apprentissage.

Le réseau a "appris" à reconnaître un PA de l'ensemble d'apprentissage lorsque le neurone de sortie associé à ce PA est celui souhaité (nombre proche de 1 pour ce neurone, proche de 0 pour les autres). La "connaissance" du réseau se situe dans les poids synaptiques entre les neurones des couches successives. Cette configuration du réseau est alors figée. Tout nouveau PA, pris hors de l'ensemble d'apprentissage, est soumis à la couche d'entrée du réseau et produit un patron de sortie dans la couche du même nom. Cette sortie représente la classification du réseau. Celle-ci est d'autant plus nette qu'il existe un neurone de sortie proche de 1, les autres étant proches de 0.

La principale faiblesse de cette approche réside dans la nécessité de construire un ensemble d'apprentissage avec des PAs dont on a déjà réalisé la classification. Il s'agit donc une méthode hautement supervisée. L'expérimentateur doit s'assurer que tous les neurones sont représentés dans cet ensemble, et qu'il dispose de plusieurs exemplaires de PAs par neurone. Cette méthode peut s'avérer performante pour classer automatiquement tous les PAs restants (hors de l'ensemble d'apprentissage et constituant ce qui est appelé l' "ensemble test"), mais elle ne résoud pas le problème de la classification automatique initiale. Elle réalise, d'une certaine manière, une forme de *template matching*, l'information relative à un *template* étant stockée dans les poids synaptiques du réseau qui lui sont associés.

3.5 Détermination du nombre de neurones

L'un des problèmes les plus difficiles du *spike-sorting* est la détermination du nombre de neurones (c'est-à-dire, ici, du nombre de nuages) présents dans les données. Toutes les méthodes de *clustering* exposées ci-dessus nécessitent une détermination *a priori* de ce nombre par l'expérimentateur, à l'exception du *superparamagnetic clustering*, dont on a vu cependant la faiblesse à cet égard. Les réseaux de neurones artificiels exigent même la construction d'un ensemble d'apprentissage de PAs entièrement labélisés. L'expérience de l'expérimentateur peut être un bon guide dans l'estimation du nombre de neurones. Cependant, il est plus prudent et plus satisfaisant de fonder quantitativement un tel choix.

Dans le cadre des modèles de mélange évoqués, on procède généralement à un ajustement (*fit*), par maximum de vraisemblance, des paramètres des modèles de mélange

de différentes dimensions, c'est-à-dire comprenant différents nombres de neurones (cet ajustement est le plus souvent réalisé avec l'algorithme EM, voir section 3.3.4). On compare ensuite ces différents modèles ajustés. Les critères de comparaison utilisés sont généralement la vraisemblance du modèle ajusté, pénalisée pour sa dimension. On choisit alors le modèle optimal pour le critère retenu. Deux critères de comparaison de modèles sont couramment utilisés : le *Akaike Information Criterion* (AIC) (Akaike, 1973) et le *Bayesian Information Criterion* (BIC)(Schwarz, 1978). Dans ce qui suit, M_K ($K = K_1, \ldots, K_r$) désigne un modèle de mélange avec K composantes (*i.e* neurones) et $\hat{\theta}_K$ les paramètres du maximum de vraisemblance de ce modèle, ajusté sur les données $\mathbf{z}$.

Le critère AIC

L'enjeu est de sélectionner le *meilleur modèle approximant* la vraie loi des données $P_0(\mathbf{z})$. Cette loi ne peut évidemment pas être déterminée ; il s'agit donc de choisir, parmi les différentes lois $P(\mathbf{z} \mid M_K, \hat{\theta}_K) = P_{M_K}(\mathbf{z})$, celle qui minimise une certaine mesure de dissimilarité entre elle et P_0, à savoir la divergence de Kullback-Leibler :

$$KL(P_0, P_{M_K}) = \int P_0(\mathbf{z}) \ln(\frac{P_0(\mathbf{z})}{P_{M_K}(\mathbf{z})}) d\mathbf{z} \tag{3.31}$$

Dans cette approche, aucun des modèles comparés ne contient P_0 et le but est de déterminer le modèle le plus proche de P_0 au sens de la divergence de Kullback-Leibler. Comme l'on désire sélectionner un modèle qui ait une bonne capacité prédictive, on choisit comme critère de sélection la minimisation de la valeur moyenne de la quantité 3.31 pour des réplications indépendantes de $\tilde{\mathbf{z}}$ (émises selon la loi P_0), c'est-à-dire la minimisation de son espérance par rapport à la vraie loi P_0 :

$$E_{P_0}\left[KL(P_0, P_{M_K})\right]$$

On peut réécrire la divergence de Kullback-Leibler 3.31 comme la différence de deux termes, le premier ne dépendant pas du modèle M_K :

$$KL(P_0, P_{M_K}) = \int P_0(\mathbf{z}) \ln(P_0(\mathbf{z})) d\mathbf{z} - \int P_0(\mathbf{z}) \ln(P_{M_K}(\mathbf{z})) d\mathbf{z} = constante - \int P_0(\mathbf{z}) \ln(P_{M_K}(\mathbf{z})) d\mathbf{z}$$

Le critère à minimiser se réécrit alors :

$$E_{P_0}\left[KL(P_0, P_{M_K})\right] = constante - E_{P_0}\left[\int P_0(\mathbf{z}) \ln(P_{M_K}(\mathbf{z})) d\mathbf{z}\right] \tag{3.32}$$

Akaike (1973) a montré que la quantité suivante était un estimateur asymptotiquement non biaisé du deuxième terme du membre de droite, $E_{P_0}\left[-\int P_0(\mathbf{z}) \ln(P_{M_K}(\mathbf{z})) d\mathbf{z}\right]$:

$$AIC(M_K) = -\ln\left(P(\mathbf{z} \mid M_K, \hat{\theta}_K)\right) + \nu(M_K) \tag{3.33}$$

Il s'agit donc de la log-vraisemblance du modèle M_K ajusté aux données $\mathbf{z}$ selon le maximum de vraisemblance, pénalisée par la dimension $\nu(M_K)$ de ce modèle, c'est-à-dire le nombre de ses paramètres libres. Le critère utilisé en pratique est souvent le double de 3.33.

Ainsi, la minimisation de ce critère permet de déterminer le modèle M_{K_0} qui minimise, en moyenne, parmi tous les modèles en compétition $\{M_K\}$, un estimateur de la divergence de Kullback-Leibler du modèle à la vraie loi.

Le critère BIC

Ce critère s'inscrit dans un contexte bayésien : les modèles M_K et ses paramètres θ_K sont vus comme des variables aléatoires et possèdent une distribution *a priori*. On note $P(M_K)$ la distribution *a priori* de M_K. Pour un modèle M_K donné, la distribution *a priori* de ses paramètres θ_K est notée $P(\theta_K \mid M_K)$. Le critère BIC cherche à sélectionner le modèle M_K qui maximise la probabilité *a posteriori* $P(M_K \mid \mathbf{z})$:

$$M_{K_{BIC}} = argmax_{M_{K_i}} P(M_{K_i} \mid \mathbf{z}) \tag{3.34}$$

BIC cherche donc à sélectionner le modèle le plus vraisemblable au vu des données $\mathbf{z}$. La formule de Bayes s'écrit :

$$P(M_K \mid \mathbf{z}) = \frac{P(\mathbf{z} \mid M_K)P(M_K)}{P(\mathbf{z})} \tag{3.35}$$

On suppose, dans ce qui suit, que la loi *a priori* des modèles M_K est non informative :

$$P(M_{K_1}) = P(M_{K_2}) = \ldots = P(M_{K_r}) \tag{3.36}$$

Dans cette hypothèse, aucun modèle n'est privilégié et, d'après 3.34 et 3.35, la recherche du meilleur modèle ne nécessite que le calcul de $P(\mathbf{z} \mid M_K)$. Sous l'hypothèse où n est suffisamment grand (*i.e* en régime normal pour les distributions *a posteriori*). Un développement limité du logarithme de cette distribution montre que (Lebarbier et Mary-Huard, 2004) :

$$\ln\left(P(\mathbf{z} \mid M_K)\right) \approx \ln\left(P(\mathbf{z} \mid M_K, \hat{\theta}_K)\right) - \frac{\nu(M_K)}{2}\ln(n) \tag{3.37}$$

où n est la taille de l'échantillon de données $\mathbf{z}$ et $\nu(M_K)$ la dimension du modèle M_K. Rappelons que le premier terme du membre de droite dans 3.37 est la log-vraisemblance du modèle M_K ajusté aux données $\mathbf{z}$ selon le maximum de vraisemblance. D'après 3.35 et 3.36, maximiser la probabilité *a posteriori* $P(M_K \mid \mathbf{z})$ selon M_K revient à maximiser 3.37.

Le critère BIC est égal à l'approximation de $-2\ln\left(P(\mathbf{z} \mid M_K)\right)$ et doit, de ce fait être minimisé (Schwarz, 1978) :

$$BIC(M_K) = -2\ln\left(P(\mathbf{z} \mid M_K, \hat{\theta}_K)\right) + \nu(M_K)\ln(n) \tag{3.38}$$

Ce critère est étroitement lié au facteur de Bayes, également utilisé pour la comparaison de modèles. Le facteur de Bayes B_{12} de deux modèles M_{K_1} et M_{K_2} est défini par :

$$B_{12} = \frac{P(\mathbf{z} \mid M_{K_1})}{P(\mathbf{z} \mid M_{K_2})} = \frac{P(M_{K_1} \mid \mathbf{z}) \cdot P(M_{K_2})}{P(M_{K_2} \mid \mathbf{z}) \cdot P(M_{K_1})} = \frac{P(M_{K_1} \mid \mathbf{z})}{P(M_{K_2} \mid \mathbf{z})} \tag{3.39}$$

La troisième égalité est valable sous l'hypothèse, faites ci-dessus, de probabilités *a priori* non informatives pour les modèles (aucun modèle privilégié au départ). Lorsque B_{12} est supérieur (resp. inférieur) à 1, ou son logarithme supérieur (resp. inférieur) à 0, le modèle M_{K_1} (resp. M_{K_2}) est favorisé. Lorsque $n \to \infty$, on a (Kass et Raftery, 1995) :

$$\frac{-2\ln(B_{12}) - (BIC(M_{K_1}) - BIC(M_{K_2}))}{-2\ln(B_{12})} \to 0$$

Ainsi $(BIC(M_{K_1}) - BIC(M_{K_2}))$ peut être vu comme une approximation du logarithme du facteur de Bayes, $-2\ln(B_{12})$, des deux modèles.

Conclusion

Qualitativement, ces critères sélectionnent les modèles qui réalisent les meilleurs compromis entre qualité de l'ajustement, assurée par le terme de vraisemblance, et parcimonie, assurée par $\nu(M_K)$. Il est évident que plus la dimension ν du modèle est grande (*i.e* plus le nombre de neurones du modèle est important), meilleur est son ajustement aux données $\mathbf{z}$ et plus grande est sa log-vraisemblance. Par conséquent, une sélection fondée uniquement sur la qualité de l'ajustement privilégie toujours le modèle de plus grande dimension, au risque que celui-ci soit surajusté aux données particulières $\mathbf{z}$, c'est-à-dire au bruit de ces données. Un tel modèle surajusté a une très mauvaise capacité prédictive et son ajustement à d'autres données $\tilde{\mathbf{z}}$ émises selon la même loi P_0 sera mauvais. Il est donc intuitif de vouloir pénaliser la qualité de l'ajustement d'un modèle (la vraisemblance ou son logarithme) par sa dimension. Si l'accroissement de la vraisemblance apportée par le modèle M_{K+1} par rapport à M_K, *i.e* le gain en ajustement aux données, n'est pas suffisant au regard du nombre supplémentaire de paramètres utilisés, le modèle M_{K+1} est rejeté au profit de M_K.

Le terme de pénalité n'est pas le même pour les deux critères présentés. La pénalisation du BIC est en général plus lourde que celle du AIC ($\nu(M_K)\ln(n) > 2\nu(M_K)$ pour $n \geq 8$). BIC a donc tendance à privilégier des modèles de plus petite dimension, plus parcimonieux, que AIC. En règle générale, le choix du critère à utiliser est délicat. Les résultats sur données simulées montrent que leurs performances pratiques sont fonction des données, en particulier de la complexité du vrai modèle (qui fait partie de la liste des modèles comparés dans le cas de simulations) et des modèles candidats, ainsi que de la taille de l'échantillon (Lebarbier et Mary-Huard, 2004). On peut retenir grossièrement que le AIC est meilleur pour la sélection de modèles prédictifs, c'est-à-dire dont l'ajustement reste bon pour d'autres échantillons $\tilde{\mathbf{z}}$ émis selon la même loi que l'échantillon $\mathbf{z}$ utilisé pour ajuster les modèles. Le BIC est meilleur pour la sélection de modèles explicatifs, en particulier pour la sélection du vrai modèle de génération de $\mathbf{z}$ dans le cas de simulations.

Dans la pratique du *spike-sorting*, ces critères n'apportent pas toujours de réponse définitive satisfaisante, dans la mesure où la région de leur minimum est souvent plate et couvre plusieurs nombres de neurones, souvent très surestimés d'ailleurs. Dans ce cas, c'est à l'expérimentateur que revient le choix final du nombre de neurones.

3.6 Classification des PAs d'enregistrements ultérieurs

Une fois le *spike-sorting* effectué sur une période limitée des données collectées, on peut estimer, dans l'espace de dimension d de départ, la forme moyenne des PAs d'un même nuage. Cette forme moyenne est en quelque sorte la forme idéale (*template*), prototypique, ou standard, du PA du neurone ayant généré ce nuage, au cours de cette période d'enregistrement. On peut donc utiliser les formes moyennes de chaque nuage comme jeu de PAs prototypiques pour classer chaque PA détecté dans la période suivante de l'enregistrement selon la méthode du *template matching* (Salganicoff et al., 1988 ; voir section 3.1.3, Fig. 3.3). La classification de tous ces nouveaux PAs est de ce fait très rapide. On peut de plus, sur ce nouveau segment d'enregistrement, re-calculer chaque forme standard en moyennant les formes des PAs qui lui sont attribués. On réalise ainsi une mise à jour utile des formes standard. En effet, une légère dérive de l'électrode dans le tissu est inévitable au cours du temps. Cette dérive lente modifie la distance des sites d'enregistrement aux neurones enregistrés et donc les formes de leurs PAs. Il peut donc s'avérer crucial de mettre à jour chaque forme standard avec les PAs qui lui sont attribués sur un segment d'enregistrement, afin d'utiliser ces nouvelles formes standard pour classer les PAs du segment suivant par *template matching*.

Cette façon de procéder présente néanmoins l'inconvénient de ne pouvoir prendre en compte l'apparition de toute nouvelle forme de PA au cours de l'enregistrement. Il peut en effet arriver qu'un neurone silencieux devienne actif. Pour faire face à ce problème, l'échantillon de PAs soumis au *clustering* doit être réparti sur la totalité de l'enregistrement, de manière à inclure toutes les classes de PAs, même celles apparues tardivement. On perd alors la capacité mentionnée ci-dessus à suivre l'évolution lente des PAs due à la dérive inévitable de l'électrode. Sur des périodes courtes d'enregistrement, ce changement progressif des formes de PAs peut être parfaitement négligeable devant les différences qui les séparent et l'expérimentateur peut opter pour cette seconde procédure.

Dans la perspective tracée ici, le *spike-sorting* effectué avec une méthode de *clustering* sur un sous-échantillon de l'ensemble des PAs enregistrés permet une détermination automatique de toutes les formes standard - éventuellement mises à jour - à utiliser pour classer tous les PAs de l'enregistrement, par simple comparaison. Ce sous-échantillon peut être constitué soit des PAs du premier segment d'enregistrement, soit de PAs répartis sur tout l'enregistrement analysé, suivant ce que souhaite privilégier l'expérimentateur : suivre la dérive de l'électrode ou détecter un neurone inactif au départ.

3.7 Estimation de la qualité de la classification

La procédure de *spike-sorting* ne peut être achevée sans une évaluation de la qualité de ses résultats. Une fois la classification des PAs effectuée, on dispose de trains de PAs distincts qui doivent correspondre aux décharges émises par les neurones individuels enregistrés. Afin de s'assurer de la pertinence de cette reconstruction des trains de PAs,

il est nécessaire d'estimer la qualité de la classification effectuée neurone par neurone et d'identifier ceux qui sont mal séparés. J'entends ici "classification" dans le sens de classification "forcée", attribuant un numéro de neurone à chaque PA et non une probabilité d'appartenance à un neurone (classification "douce"). Ce problème de l'estimation de la qualité de la classification n'a pas souvent été abordé quantitativement dans la littérature (Pouzat et al., 2002). Les tests présentés dans cette section doivent permettre à l'expérimentateur de détecter la présence de plusieurs types de PAs dans une même classe issue de la procédure de classification. Ce sont donc des modes d'évaluation de la classification nuage par nuage. Face à ces classes de PAs mal séparées, l'expérimentateur a deux possibilités : (i) si, par ailleurs, il a de bonnes raisons de penser que ces différents types de PAs sont émis par un même neurone, par exemple lorsqu'il émet des *bursts* (voir sections 3.8 et Méthodes), il doit évidemment garder ces PAs sous le même label. (ii) si, en revanche, il s'agit de neurones qui ne sont pas précisément isolés par le *spike-sorting*, il doit les éliminer ou les traiter comme données multi-unitaires. On trouvera une mise en oeuvre et des illustrations de ces tests dans le chapitre 4 (Méthodes).

3.7.1 Tests généraux

Visualisation des PAs attribués à une même classe

L'expérimentateur peut visualiser de diverses manières, toutes très immédiates, les PAs qui ont été classés comme issus du même neurone. Il peut, par exemple, superposer sur un même graphique tous les PAs d'une même classe afin de juger de la pertinence de cette classe et de la présence d'intrus éventuels (voir Fig. 4.3).

Il peut également parcourir la trace brute "labélisée" des données, c'est-à-dire la trace brute de l'enregistrement munie des numéros de neurones associés à chaque PA (code de couleurs, symboles, etc., voir Fig. 4.6). Il peut enfin visualiser les nuages dans l'espace des caractéristiques choisi (section 3.2) colorés selon leur labélisation (voir Fig. 4.7). Ceci permet de détecter les erreurs les plus évidentes et d'identifier les classes de PAs suceptibles de poser problème. Dans la mesure où la détermination du nombre de neurone est une décision qui revient souvent à l'expérimentateur (section 3.5), cet examen direct de la labélisation effectuée fournit une aide précieuse lors de ce choix.

Test de la déviation standard (SD)

Si tous les PAs attribués à une même classe ont effectivement été émis par un même neurone, la variance de ces PAs n'est due qu'au bruit et doit rester à peu près constante, au niveau de la SD du bruit, pendant tout le décours temporel du PA. Si, en revanche, la classe considérée contient plusieurs types de PAs, correspondant à des neurones différents, la SD de la classe présente des maxima aux temps où ces types de PAs diffèrent le plus. On peut donc tester la SD de chaque nuage de la classification effectuée (Pouzat et al., 2002) : tout nuage dont la SD diffère significativement de la SD du bruit contient des PAs d'au moins deux types différents.

Histogrammes des intervalles entre PAs

Une fois les trains de PAs reconstitués, il est immédiat de tracer les histogrammes des intervalles entre PAs (*inter-spike intervals*, ISIs) de ces trains (voir Fig. 4.8B). Pour qu'un train de PA reconstitué soit valide, la condition minimale est qu'il présente une nette période réfractaire. Ceci se traduit par l'absence d'ISI de durée inférieure à quelques ms (la durée de la période réfractaire dépend du type de neurone enregistré ; elle est, en général, de l'ordre de 2 à 10 ms). Dans le cas contraire, l'expérimentateur est certain que le train de PAs considéré mélange, dans des proportions non définies à ce stade, les activités de plusieurs neurones.

D'autres caractéristiques des histogrammes d'ISIs peuvent également être examinées. Il est fréquent que la statistique de décharge des neurones enregistrés soit relativement reproductible, ce qui se traduit par une forme typique de ses histogrammes d'ISIs (uni- ou multi-modalité, localisation du (des) mode(s), présence ou non d'une queue plus ou moins lourde, etc.). Ces histogrammes d'ISIs de neurones individuels peuvent être tracés à partir d'enregistrements *cell-attached* de ces neurones. L'expérimentateur peut visuellement estimer la similarité entre les histogrammes de ses trains reconstitués à partir d'enregistrements multiples et les histogrammes des trains de neurones enregistrés individuellement.

3.7.2 Tests fondés sur un modèle gaussien du bruit

Lorsque la méthode de *spike-sorting* a recours à un modèle quantitatif de génération de données (section 3.3.4), l'expérimentateur peut s'appuyer sur ce dernier pour faire des prédictions précises sur ses données classifiées. De même que le test de la SD, les deux tests présentés dans les prochains paragraphes sont tous deux dus à Christophe Pouzat (Pouzat et al., 2002).

Test du χ^2

Ce test est fondé sur l'hypothèse d'un bruit gaussien non corrélé, indépendamment ajouté aux PAs. Il peut donc être utilisé dans le cadre d'un modèle de mélange gaussien multivarié (section 3.3.4). Ce modèle suppose que chaque nuage forme une distribution gaussienne de dimension m. La distance de Mahalanobis de tout point $\mathbf{z}^j$ de ce nuage à son centre $\mathbf{u}$ (le vecteur moyen de ce nuage) est définie comme la somme des carrés des différences des composantes de ces vecteurs (distance euclidienne) :

$$d(\mathbf{z}^j, \mathbf{u}) = \sum_{i=1}^{m} (z_i^j - u_i)^2 \tag{3.40}$$

Dans le cadre des modèles de mélange gaussien, les termes (sous le carré) de cette somme sont des gaussiennes indépendantes centrées sur 0 et, si le bruit a initialement été blanchi (Pouzat et al., 2002 ; voir chapitre 4), de SD 1. Cette distance 3.40 est donc une somme de m carrés de variables aléatoires gaussiennes réduites indépendantes, c'est-à-dire d'un χ^2 à m degrés de libertés. Par conséquent, si la variance des PAs des nuages n'est due qu'au bruit, et si le bruit peut être expliqué par un modèle gaussien, alors

la distribution des distances des points appartenant à un même nuage au centre de ce nuage suit une distribution du χ^2 à m degrés de libertés.

On peut ainsi tracer, pour chaque nuage, la distribution cumulée des distances de ses points à son centre et la comparer à la distribution du χ^2 à m degrés de libertés. Si la déviation de la première par rapport à la seconde est forte, et si par ailleurs la distribution du bruit est proche de celle du χ^2, le nuage en question comprend des PAs de différents types (Pouzat et al., 2002) ; là encore, soit il s'agit d'un neurone dont les PAs ne sont pas stationnaires, soit il s'agit de plusieurs neurones mal séparés. On peut également tracer un graphique quantile-quantile (*Q-Q plot*) pour chaque neurone pour comparer la distribution des distances de Mahalanobis des événements de ce neurone à leur moyenne à la distribution du χ^2 : les quantiles de la distribution des distances de Mahalanobis sont tracés contre ceux de la distribution du χ^2. Si les deux distributions sont identiques ou proches, le graphique résultant est une droite confondue avec la première bissectrice.

Dans le graphique quantile-quantile précédent, on peut également se contenter de la distribution des carrés des distances à l'origine d'événements de bruit comme distribution de référence en lieu et place de celle du χ^2. On compare alors directement la distribution des distances de Mahalanobis des résidus d'un neurone à la distribution des distances de Mahalanobis des événements de bruit (voir Fig.4.8A). Si l'hypothèse d'une superposition indépendante et linéaire du bruit et d'un PA est correcte, ces deux distributions sont égales pour un neurone bien isolé et ce graphique doit alors se concentrer autour de la première bissectrice.

Test de projection

Comme le précédent, ce test est fondé sur l'hypothèse d'un bruit gaussien non corrélé, indépendamment ajouté aux PAs. Si l'on projette les points de deux nuages de centres $\mathbf{u}_1$ et $\mathbf{u}_2$ sur l'axe joignant ces centres, ces projections doivent former deux gaussiennes centrées sur les points $\mathbf{u}_1$ et $\mathbf{u}_2$ et, si le bruit a été initialement blanchi, de SD 1. Le taux de recouvrement de ces deux distributions reflète évidemment la distance euclidienne entre les deux centres $\mathbf{u}_1$ et $\mathbf{u}_2$. On peut donc convertir un taux de mauvaises classifications entre deux nuages (recouvrement des deux nuages) en une distance entre les centres de ces nuages (Pouzat et al., 2002). Pour que moins de 5% des PAs du nuage $\mathbf{u}_1$ soient classés comme venant de $\mathbf{u}_2$, il faut une distance minimale entre $\mathbf{u}_1$ et $\mathbf{u}_2$ supérieure à $2.5SD$ (soit 2.5 pour un bruit blanchi, de SD 1), en supposant que les nombres de points sont les mêmes dans les deux nuages. L'expérimentateur peut décider qu'au-dessus d'un tel taux de recouvrement, *i.e* pour une distance entre centres inférieure à 2.5, les nuages ne sont pas séparables.

3.7.3 Décalage des formes dû à l'échantillonnage (*sampling jitter*)

L'échantillonnage des PAs à fréquence finie est à l'origine d'une variabilité supplémentaire des PAs qu'il faut pouvoir annuler avant de réaliser certains de ces tests (Pouzat et al., 2002). Raisonnons sur la forme standard d'un neurone, sans bruit ad-

ditionnel. Dans la mesure où l'horloge de l'ordinateur, responsable de l'échantillonnage numérique, n'est pas synchronisée avec celle du neurone considéré, la localisation des points d'échantillonnage sur cette forme standard change à chaque apparition de celle-ci : l'ensemble des points d'échantillonnage est plus ou moins translaté sur la forme standard, d'une instantiation à l'autre. Par conséquent, la position du pic (ou maximum) du vecteur d'échantillonnage d'une même forme standard varie et cette variation est de l'ordre d'un ou deux point(s) d'échantillonnage. Après alignement de ces vecteurs sur leurs pics, les PAs d'une même classe sont donc légèrement translatés les uns par rapport aux autres. Cette variabilité supplémentaire peut avoir un effet significatif sur des tests comme ceux de la SD et du χ^2, comme illustré dans Pouzat et al. (2002).

Ces auteurs préconisent donc son annulation avant d'effectuer ces tests. Le principe est une application directe du théorème de Nyquist (Pouzat et al., 2002) : chaque PA fait l'objet d'une interpolation à l'aide d'un filtre, à savoir la fonction *sinc* (*i.e* $sin(x)/x$) de période égale à deux fois la fréquence d'échantillonnage. Cette procédure permet d'obtenir une forme complète du PA, qui est dès lors décrite par un échantillonnage de fréquence dix fois plus élevée. Cette forme interpolée est alignée sur le vecteur moyen (interpolé) du nuage d'appartenance de ce PA, par minimisation de leur distance (comme d'habitude, cette dernière est définie comme la somme des carrés des composantes de leur vecteur différence). Tous les PAs de la classe considérée sont donc strictement alignés et peuvent faire l'objet des tests de la SD et du χ^2.

3.8 Les grands défis

3.8.1 Les limites des méthodes actuelles de *spike-sorting*

La très vaste majorité des méthodes de *spike-sorting* publiées sont une combinaison des méthodes exposées dans les sections 3.2 et 3.3 de ce chapitre. Elles requièrent en effet le choix d'une méthode d'extraction d'un nombre limité de caractéristiques de la forme des PAs qui soient discriminantes, ainsi que le choix d'une méthode de *clustering*. Le choix de la méthode d'extraction se décompose par ailleurs en deux : choix d'une transformation de la représentation initiale des PAs et choix d'un mode de sélection des éléments les plus discriminants de cette représentation.

Il est dès lors nécessaire d'insister sur le point, évident mais essentiel, suivant : en fondant leur classification sur la forme des PAs uniquement, toutes ces méthodes font l'hypothèse de *stationnarité des PAs* émis par un neurone donné. Dans cette perspective, la variabilité des PAs n'est due qu'au bruit. Ceci appelle deux remarques (1 et 2) et pose *de facto* deux limites à la démarche (3 et 4) :

1. En négligeant tout une partie de l'information contenue dans les enregistrements, à savoir l'*information temporelle* des trains de PAs, ces méthodes se privent d'une précieuse aide à la classification. En effet, la séquence temporelle des PAs émis par un neurone (sa "décharge") possède des caractéristiques bien connues, comme, par exemple, la présence d'une période réfractaire (deux PAs d'un même neurone sont toujours séparés d'au moins 2 à 10 ms, selon le neurone). Plus encore, elle peut être décrite par une densité de probabilité des intervalles entre PAs (ISIs) qui peut être

relativement reproductible. Cette information fournie par les temps des PAs peut donc s'avérer très précieuse pour décider de leurs attributions respectives à des neurones. Notons que dans les modèles de mélanges (section 3.3.4), l'hypothèse implicite à ce propos est que les temps des PAs d'un neurone sont Poisson, avec une fréquence moyenne proportionnelle à π_k, la proportion de la composante k du mélange. C'est l'hypothèse minimale, la moins contraignante, que l'on peut faire sur la structure temporelle des trains. Mais elle est fausse la plupart du temps. Lewicki dit à ce propos : "A more powerful description, *e.g* modeling the distribution of inter-spike interval, would be obtained by incorporating this information into [the model]" (Lewicki, 1994).

2. L'information initiale sur la forme des PAs fournie par plusieurs sites d'enregistrement rapprochés étant toujours plus riche et discriminante que celle fournie par un seul site d'enregistrement, elle améliore toujours le *spike-sorting* accompli par ces méthodes. Par conséquent, quelle que soit la méthode utilisée, il est toujours avantageux d'utiliser des *électrodes multisites*. En particulier, lorsque deux neurones sont situés à égale distance d'un site d'enregistrement, il est souvent impossible de distinguer leurs PAs sur ce site. La présence d'autres sites dans un proche voisinage permet alors leur différenciation (voir Fig. 1.2). Néanmoins, on ne peut jamais éviter les *recouvrements* de nuages de points. Les points situés dans les zones de recouvrements ne peuvent pas être classifiés de façon fiable sans information supplémentaire.
3. L'hypothèse fondamentale de *stationnarité des formes de PAs* émis par un neurone interdit à toutes ces méthodes de traiter des données dans lesquelles la forme des PAs d'un neurone change. En particulier, lorsqu'un neurone émet des PAs très rapprochés dans le temps (par exemple au cours d'un *burst*), il est fréquent que l'amplitude des PAs successifs décroisse fortement. Il y a donc une réelle dépendance de l'amplitude d'un PA vis-à-vis de l'intervalle de temps le séparant du précédent (non-stationnarité des PAs) qui ne peut être prise en compte par aucune de ces méthodes (voir Fig. 2.4).
4. Lorsque deux PAs sont émis par deux neurones différents quasi-simultanément (*i.e* dans un intervalle de temps inférieur à la durée d'un PA), leur *superposition* résulte en un décours temporel particulier inédit. Les méthodes exposées ici classent en général ces superpositions comme PAs non identifiés (Fig. 3.5).

Les points 3 et 4 sont les deux grands défis posés au *spike-sorting*. Voici les réponses qui ont été apportées jusqu'ici.

3.8.2 Les réponses apportées

La non-stationnarité des PAs

Ce problème n'a pas fait l'objet de très nombreux travaux. Il est une limite intrinsèque à la démarche, commune à toutes les méthodes, de fonder la classification sur la forme des PAs uniquement. Jansen (1990) mentionne que si des PAs d'amplitudes différentes, dans l'ensemble d'apprentissage, sont associés à un même neurone

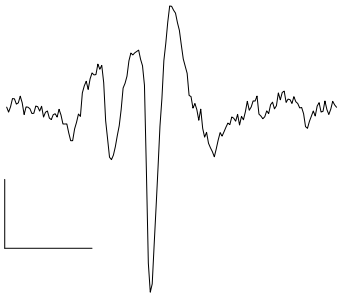

Figure 3.5: **Superposition de deux potentiels d'action.** Deux potentiels d'action sont émis quasi-simultanément par deux neurones différents. La superposition de ces deux événements donne naissance à une forme particulière souvent inédite et unique dans l'enregistrement. Barre d'échelle horizontale: $3\,ms$. Barre d'échelle verticale: $100\,\mu V$.

de sortie, le réseau de neurones artificiels les reconnaît comme venant du même neurone dans l'ensemble test. Cependant, cette labélisation identique de PAs d'amplitudes différentes est une décision que prend l'expérimentateur lui-même lorsqu'il construit son ensemble d'apprentissage. Aucune méthode de *clustering* (par ailleurs nécessaire pour construire l'ensemble d'apprentissage) ne permet de l'obtenir automatiquement. La non-stationnarité des PAs d'un neurone donne en effet naissance, pour ce neurone, à un nuage très allongé - dont la dispersion est très différente du modèle supposé par la méthode - ou même à des nuages distincts. De ce fait, ces méthodes associent nécessairement plusieurs classes de PAs au neurone en question.

Il est possible de revenir sur une première classification en regroupant *a posteriori* les nuages scindés ou séparés venant du même neurone. Fee et al. (1996) fondent ces agrégations de nuages sur deux éléments : la force de connexion entre deux nuages et leur histogramme d'ISIs. La première est définie comme une énergie d'interface calculée sur toutes les paires de points des nuages. Elle doit être grande pour que la paire soit considérée comme issue du même neurone. Le second représente tous les intervalles de temps séparant deux PAs successifs de l'ensemble des deux nuages. Il doit présenter une période réfractaire aussi nette que celle des deux nuages parents pour que ceux-ci soient attribués au même neurone. En fait, les auteurs commencent par réaliser une sur-partition des données (*overclustering*) de façon à obtenir environ dix fois plus de nuages que ce qu'ils estiment nécessaire. Ils fusionnent ensuite les paires de nuages selon les deux critères mentionnés. Les auteurs ne font aucune hypothèse sur la nature de la distribution des formes de PA au sein d'un nuage (*i.e* sur la nature du bruit). La méthode décrite est une réponse *ad hoc* au problème que posent aux méthodes de *clustering* l'anisotropie du bruit et la non-stationnarité des PAs d'un même neurone. Cette réponse passe par l'introduction, pour la première fois, d'une partie de l'information temporelle des trains de PAs (la période refractaire) dans la procédure de classification.

Les superpositions de PAs

Le problème des superpositions de PAs a fait l'objet de travaux plus nombreux. C'est dans le cadre du *template matching* qu'il est le plus simple de le traiter. Par exemple, on peut soustraire aux PAs superposés la forme standard à laquelle ils sont associés, en espérant pouvoir classer la forme résiduelle avec une autre forme standard. Ce procédé simple de soustraction nécessite une relativement bonne séparation temporelle des PAs superposés, permettant au premier PA d'être reconnu. Une approche plus robuste consiste à comparer les PAs superposés à toutes les combinaisons possibles, et toutes les translations relatives, de deux formes standard, afin de sélectionner la combinaison la plus probable (Atiya, 1992). Il s'agit ici d'explorer de façon exhaustive l'espace des superpositions des formes standard. Cette approche est computationnellement très lourde, surtout si le nombre de formes standard est grand.

Lewicki fonde sa solution sur le même principe (Lewicki, 1994). Mais il construit un algorithme de décomposition des superpositions plus efficace en restreignant l'espace des superpositions à celles qui ont une probabilité non négligeable. Il optimise également la méthode de recherche dans cet espace en ayant recours à la construction d'un arbre de recherche et à la programmation dynamique.

Les réseaux de neurones, soumis à un apprentissage adéquat, peuvent se révéler relativement performants dans la résolution de PAs superposés (Chandra et Optican, 1997). Ces auteurs incluent dans leur ensemble d'apprentissage un certain nombre de superpositions, avec translations relatives des paires de formes standard utilisées. Cette méthode souffre de la faiblesse des méthodes utilisant les réseaux de neurones : ceux-ci doivent au préalable être soumis à un apprentissage sur un ensemble de PAs déjà labélisés. Ce que nous montrent Chandra et Optican fondamentalement, c'est qu'en étoffant leur ensemble d'apprentissage avec un certain nombre de superpositions, ils rendent leur réseau capable d'identifier celles-ci. De plus, leur méthode est testée sur des données simulées à partir des trois formes standard utilisées par ailleurs dans l'ensemble d'apprentissage du réseau. Cette situation est évidemment très favorable.

3.9 Conclusion

Au terme de ce tour d'horizon, il apparaît que, si le *spike-sorting* est un problème délicat en électrophysiologie, il n'est pas totalement inextricable. Avec les méthodes exposées dans ce chapitre, les électrophysiologistes sont en mesure de traiter des données dans lesquelles les formes de PAs des neurones sont stationnaires, les superpositions pas trop fréquentes et les différences entre classes de PAs relativement marquées (pas de recouvrement de nuages). La détermination du nombre de neurones demeure un problème dans la pratique, même dans le cas où une solution théorique est disponible (section 3.5).

La méthode de *Monte Carlo par Chaînes de Markov* (MCMC) développée et présentée dans cette thèse (chapitre 4 et articles 1 et 2) offre une solution au problème de la non-stationnarité des PAs d'un neurone, ainsi qu'au problème de recouvrement des nuages de points. Ceci passe par la prise en compte de la statistique de décharge des neurones enregistrés dans la classification des PAs. Par ailleurs, la détermination du

nombre de neurones est également possible dans le cadre de cette méthode. L'approche et la méthodologie bayésienne MCMC proposées offrent ainsi un cadre de travail radicalement nouveau, cohérent et fécond, au problème du *spike-sorting*. En revanche, elles n'apportent pas d'avancée notable, du moins pour l'instant, en matière de superposition de PAs.

Chapitre 4

Méthodes

Ce chapitre décrit le protocole expérimental (section 4.1), la procédure de *spike-sorting* (section 4.2) et l'analyse des trains multiples de potentiels d'action (PAs) après *spike-sorting* (section 4.3). Le contenu de la section 4.2 *a été élaboré au cours de ce travail de thèse*. Il est fondé pour partie sur un travail théorique antérieur de Christophe Pouzat (Pouzat et al., 2002), et pour partie sur le travail théorique et de programmation présenté dans les trois premiers articles du chapitre 5. Cette section 4.2 expose de manière descriptive et linéaire l'ensemble de la démarche élaborée pour réaliser le *spike-sorting*.

4.1 Le protocole expérimental

Préparation des tranches

La préparation des tranches utilisée est celle décrite par Pouzat et Hestrin (1997). Les rats, âgés de 9 à 25 jours, sont anesthésiés avec quelques gouttes d'halothane (Sigma) déposées sur un mouchoir, puis décapités. On prélève une partie du vermis du cervelet que l'on place dans une solution saline à $4^{\circ}C$ (la composition précise de cette solution est donnée plus bas). Les tranches sagittales, d'une épaisseur de $180\,\mu m$, sont obtenues à l'aide d'un vibratome (Leica). La coupe a lieu dans la solution saline à $4^{\circ}C$. Une fois coupées, les tranches sont placées dans la même solution saline à $34^{\circ}C$, dans laquelle bulle continûment un mélange de gaz 95% O_2 and 5% CO_2, pendant environ une heure. On place alors une tranche dans la chambre d'enregistrement que l'on dispose sous le microscope.

Solution externe

La solution saline évoquée dans la préparation des tranches est la solution saline externe de base (*Bicarbonate Buffered Saline*, BBS) utilisée durant toutes les expériences. Elle contient (en mM) : 130 $NaCl$, 2.5 KCl, 2 $CaCl_2$, 1 $MgCl_2$, 1.3 NaH_2PO_4, 26 $NaHCO_3$ and 25 glucose. Elle est constamment alimentée en mélange 95% O_2 and 5% CO_2. Son pH est de 7.4. Durant l'expérience, la tranche située dans la chambre

d'enregistrement est en permanence perfusée par cette solution oxygénée, maintenue à température ambiante.

Suivant l'expérience, différents bloquants et agonistes (Tocris Neuramin Ltd, Bristol, UK) sont ajoutés à la solution externe décrite. La bicuculline, bloquant des récepteurs $GABA_A$est appliquée à $25\,\mu M$; le (S)-3,5-dihydroxyphenylglycine (DHPG), agoniste des récepteurs glutamate métabotropiques du groupe I (mGluRI) est appliqué à $40\,\mu M$. Les stocks de ces agents pharmacologiques sont conservés gelés et dissous dans la solution externe au moment de leur utilisation.

Electrodes

Les enregistrements de cellules de Purkinje (PCs) individuelles sont effectués en configuration cellule attachée lâche (*loose cell-attached*), à l'aide d'une microélectrode de verre de résistance $2-4\,M\Omega$. Cette microélectrode de *patch* est remplie d'une solution proche de la solution externe, mais dans laquelle le tampon bicarbonate est remplacé par le tampon HEPES. Cette solution, dite HBS, contient (en mM) : 145 $NaCl$, 2.5 KCl, 2 $CaCl_2$, 1 $MgCl_2$, 10 HEPES, 10 glucose.

Les enregistrements simultanés de plusieurs PCs sont effectués à l'aide d'électrodes multisites fournies par le *Center for Neural Communication Technology*, de l'université de Michigan (voir Fig. 1.1). Ces électrodes sont de fines lames d'une longueur de $3\,mm$. Leur largeur à leur extrémité est de $33\,\mu m$ et leur épaisseur d'environ $3\,\mu m$. Sur cette lame, 16 sites d'enregistrement (dépôts d'irridium) sont disposés linéairement et séparés de $50\,\mu m$. La résistance de ces sites va de 1.3 à $1.9\,M\Omega$.

Identification des cellules

Toutes les expériences ont été effectuées sur les PCs. Celles-ci sont identifiées visuellement, grâce à la taille de leurs somas (dont le diamètre, $\sim 20\,\mu m$, est bien plus grand que celui de toutes les autres cellules du cortex cérébelleux) et à leur alignement caractéristique séparant les couche moléculaire et la couche des grains dans les tranches sagittales.

L'électrode multisite est disposée le long des PCs en un endroit où leur disposition est rectiligne sur plusieurs centaines de μm, dans la direction de la lame de l'électrode (Fig. 4.1). Cette dernière s'enfonce de quelques μm dans la tranche, avec un angle d'environ $10 - 15^{\circ}$ par rapport à l'horizontale. Les somas des PCs enregistrées ne sont pas visibles puisque dissimulés sous l'électrode (dont la largeur est légèrement supérieure au diamètre des PCs). L'activité extracellulaire de ces neurones peut être enregistrée sur les 8 premiers sites de l'électrode, soit environ $350\,\mu m$.

Il est très peu probable que les PAs d'autres cellules que les PCs soient enregistrés par cette méthode. En effet, le placement de l'électrode dans la couche des grains et dans la couche moléculaire n'a jamais fourni aucun signal. L'expérience montre qu'aux âges étudiés, un PA extracellulaire d'interneurone n'est visible que lorsqu'un contact étroit, proche de la configuration cellule attachée, est établi entre une pipette de verre extracellulaire et cet interneurone. Ceci est sans doute lié à la faiblesse des courants membranaires responsables des PAs d'interneurones. Les courants en jeu dans les PCs

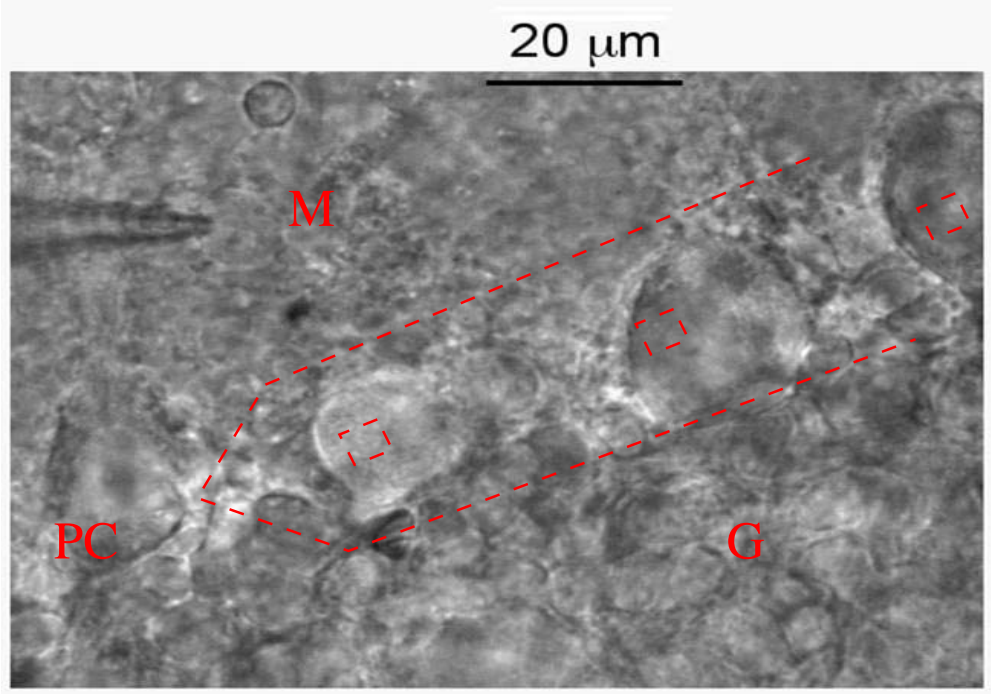

Figure 4.1: **Photographie d'une tranche de cervelet de rat telle qu'elle apparaît sous le microscope.** M: couche moléculaire. PC: couche des cellules de Purkinje. G: couche des cellules en grain. Le contour de l'électrode multisite, ainsi que ses sites d'enregistrement, sont tracés en lignes pointillées, à l'échelle, et tels qu'ils sont positionnés sur la tranche durant les expériences. Une pipette de *patch-clamp* est visible en haut à gauche pour comparaison.

sont bien plus grands et leurs PAs sont encore visibles à $50\,\mu m$ du soma. Dans la mesure où les sites d'enregistrement sont placés sur la couche des PCs, le signal enregistré est donc largement dominé par leur activité. Les PAs des cellules en panier proches de ces PCs sont de tailles bien plus faibles, indiscernables du bruit. Jamais aucun histogramme de corrélation croisée ou des temps du précédent (voir section 4.3) n'a mis en évidence l'inhibition d'un train de PAs par un autre, comme ce devrait être le cas si des cellules en panier étaient enregistrées en même temps que les PCs.

Dans les expériences de validation de la méthode de *spike-sorting* (deuxième article), la pipette de verre extracellulaire est avancée sous l'électrode multisite de façon à enregistrer les PAs de l'une des PCs vues par cette électrode. Le positionnement de la pipette sur le soma de cette PC est tel que seuls ses PAs sont présents sur la trace qu'elle enregistre. Sur l'oscilloscope (voir ci-dessous), les PAs de la pipette et certains PAs de l'un des sites de l'électrode multisite sont alors parfaitement simultanés. Le train de PAs de cette cellule est utilisé comme référence pour juger de la performance de nos méthodes de *spike-sorting* (sections 4.2.2 et 4.2.3).

Poste expérimental et appareils de mesure

La tranche est visualisée à l'aide d'un microscope Axioskop Zeiss (Allemagne), équipé d'un objectif 63X et d'un système de contraste interférentiel de Nomarski. On utilise un micromanipulateur piezzo-électrique ($70\,\mu m$ de course) monté sur un micromanipulateur mécanique ($12.5\,mm$ de course) pour déplacer les microélectrodes de verre. Pour l'électrode multisite, on utilise un micromanipulateur hydraulique ($10\,mm$ de course, Narishige International, Japon).

L 'électrode multisite est connectée à un pré-amplificateur 16 canaux situé à proximité de l'électrode et fabriqué au laboratoire. Ce pré-amplificateur amplifie deux fois le signal et est relié à deux amplificateurs différentiels de 4 canaux chacun (AM systems, modèle 1700, USA). Seuls les 8 premiers sites de l'électrode sont reliés à ces 8 canaux. L'amplification du signal y est de 1000, soit une amplification totale de 2000. Le signal est filtré entre 300 et $5000\,Hz$.

L'amplificateur utilisé pour les microélectrodes de *patch* extracellulaires est un Axoclamp 2B (Axon Instruments Inc., USA). Il amplifie 10 fois le signal et est relié à l'un des canaux des amplificateurs différentiels mentionnés ci-dessus. L'amplification de 100 de ce canal porte à 1000 l'amplification totale de ce signal extracellulaire. Son filtrage est identique à celui des canaux de l'électrode multisite.

Aucune stimulation n'est appliquée. Les activités enregistrées sont les activités spontanées des PCs, dans les différentes conditions pharmacologiques précisées plus haut.

Acquisition des données

Durant toute l'expérience, les données fournies par 4 des sites au contact des PCs sont continûment visualisées sur un oscilloscope à 4 canaux (Tektronix, TDS 224). Le rapport signal-sur-bruit est toujours excellent et peut dépasser 15. La valeur typique de l'amplitude au pic des PAs extracellulaires enregistrées est de $500\,\mu V$. L'acquisition des données est réalisée à $15\,kHz$, à l'aide d'une carte 16 bit A/D (PD2MF-64-500/16H,

United Electronics Industries, USA) et du logiciel Matlab. Les données sont stockées sur le disque dur pour l'analyse.

4.2 Le *spike-sorting*

Dans chaque expérience, les données enregistrées sur quatre sites de l'électrode sont analysées simultanément. En général, il s'agit des quatre premiers sites (Fig. 4.2). Cette analyse comprend essentiellement deux volets : la classification des trains de PAs (*spike-sorting*), décrite dans cette section, et l'analyse des corrélations temporelles entre ces trains, décrite dans la section suivante (section 4.3). Toutes les routines utilisées dans ces analyses ont été programmées par Christophe Pouzat et moi-même, en C et dans les environnements d'analyse Scilab[1] et R[2] (voir section 4.4).

Avant de procéder au *spike-sorting* à proprement parler, il faut extraire les PAs des données brutes enregistrées sur les sites de l'électrode. Cette étape préliminaire est décrite dans la sous-section 4.2.1. Deux méthodes de *spike-sorting* sont alors utilisées pour trier l'ensemble de PAs obtenus. Toutes deux reposent sur deux modèles de génération de données différents. Dans les sous-sections 4.2.2 et 4.2.3, je décris séparément les deux procédures complètes de *spike-sorting* telles qu'elles sont effectuées dans ce travail. L'exposé mathématique et algorithmique de la première méthode figure dans le chapitre 3 (section 3.3.4), celui de la seconde dans les deux premiers articles (chapitre 5). Dans ce qui suit, j'utilise également le terme d' "événement" pour désigner un PA.

4.2.1 Traitement des données brutes avant le *spike-sorting*

Détection

On commence par détecter un premier ensemble de grands événements : les maxima locaux dont les valeurs dépassent un certain seuil fixé *a priori* et égal à 5 fois la déviation standard (*standard deviation*, SD) de toute la trace de ce site. Ces événements sont moyennés et normalisés (amplitude au pic égale à 1), de façon à obtenir un prototype d'événement (*template*) avec lequel la trace de chaque site est filtrée (convolution de la trace avec l'événement prototypique). Les événements sont alors détectés sur la trace filtrée de chaque site : ce sont les maxima locaux dont les valeurs dépassent un certain seuil, défini comme multiple de la SD de la trace filtrée. Le seuil utilisé dépend des données, notamment du rapport signal sur bruit. Il est ajusté en fonction de la qualité de la détection qu'il permet d'obtenir. En général, il est compris entre 3 et 5. La longueur de l'événement prototypique utilisé pour filtrer les traces est également un paramètre d'ajustement pour la qualité de la détection. Le plus souvent cette longueur est de 60 points d'échantillonnage, soit $4\,ms$. Enfin, pour éviter de détecter un même événement sur deux sites différents, on impose une distance minimale de 5 points d'échantillonnage ($1/3\,ms$) entre deux événements. Lorsque deux événements sont détectés dans cet intervalle de temps sur deux sites différents, seul le temps de celui de plus grande amplitude est conservé.

[1] `http ://scilabsoft.inria.fr`
[2] `http ://www.r-project.org`

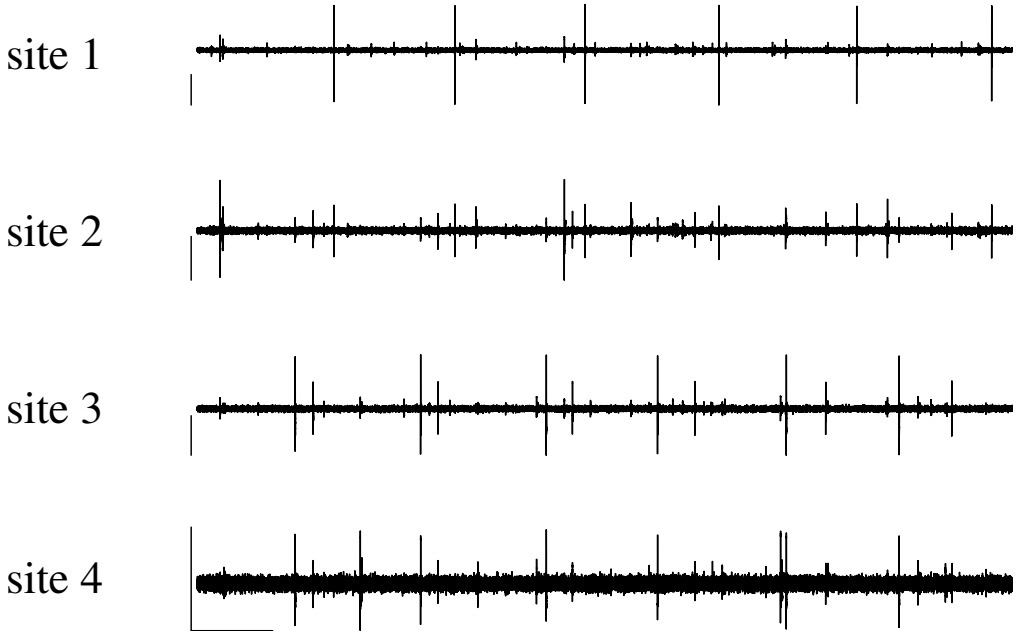

Figure 4.2: **Exemple de données brutes enregistrées sur les 4 premiers sites de l'électrode multisite** (durée totale: 5 secondes). Les données brutes extracellulaires sont un mélange d'activités neuronales individuelles. Barre d'échelle horizontale: $500\,ms$. Barres d'échelles verticales: $0.25\,mV$.

Représentation des événements

Pour chaque événement détecté, on sélectionne un segment de données autour du maximum d'amplitude sur le site où il a été détecté. Ce segment est choisi de façon à ce que son début et son terme soient à 0. En général, 45 à 75 points d'échantillonnage suffisent (soit 3 à 5 ms pour une fréquence d'échantillonnage de 15000 Hz). La position du pic d'amplitude dans ce vecteur est fixée et demeure la même pour tous les événements. Le segment de données débutant au même instant, et d'une durée identique, est sélectionné sur chacun des trois autres sites considérés. L'événement est alors représenté par un vecteur de dimension d constitué de la concaténation de ces quatre segments : pour des segments de 45 points d'échantillonnage, d vaut donc 180 (Fig. 4.3).

Reduction de la dimensionalité des événements

Afin de réduire les temps de calcul qui seront effectués ultérieurement pour la classification des événements, il est nécessaire de réduire la dimension de leur espace de représentation (chapitre 3, section 3.2). On a utilisé l'analyse en composante principale (voir section 3.2.4), ainsi que la sélection des composantes du vecteur de représentation dont la variance est la plus grande sur l'ensemble des événements détectés (voir section 3.2.3). Dans le cas de la PCA, 3 à 9 directions principales suffisaient à rendre compte de plus de 90% de la variance des événements. C'est le deuxième type de réduction de la dimensionalité, plus immédiat et également performant, qui a été le plus souvent employé. Les 3 composantes de plus grandes variances sur le vecteur d'échantillonnage de chaque site sont sélectionnées (Fig. 4.3) (rappelons qu'un tel vecteur possède 45 à 75 composantes, voir le paragraphe précédent). Les événements sont donc finalement représentés par des vecteurs de dimension 12 : on parle d' "événements réduits". Lorsque le *spike-sorting* est effectué avec l'algorithme *Monte Carlo par Chaînes de Markov* (MCMC) (section 4.2.3), les événements sont représentés par l'amplitude au pic sur chaque site d'enregistrement, soit par un vecteur de dimension 4.

Matrice de covariance du bruit et blanchiment

La matrice de covariance du bruit est calculée à partir d'un ensemble d'événements dits "de bruits" (typiquement 1000), pris entre deux événements détectés. Ces événements de bruit sont de même longueur que les événements réels initiaux décrits ci-dessus (et correspondant à des PAs). Leur dimension est réduite de la même manière que l'on a réduit les événements réels : on ne garde que les composantes de ces vecteurs de bruit qui ont été conservées sur les vecteurs des événements réels. La matrice de covariance Γ calculée sur ces événements de bruit est ensuite utilisée pour "blanchir" les événements réels (Pouzat et al., 2002).

Ce blanchiment est une transformation linéaire des événements. La matrice triangulaire supérieure T de cette transformation est obtenue à partir de Γ par une décomposition de Cholesky :

$$\Gamma^{-1} = T^t T$$

où l'exposant t désigne la transposition. La matrice T est donc appliquée à tous les événements détectés. Après cette transformation, si le bruit est bien décrit par sa

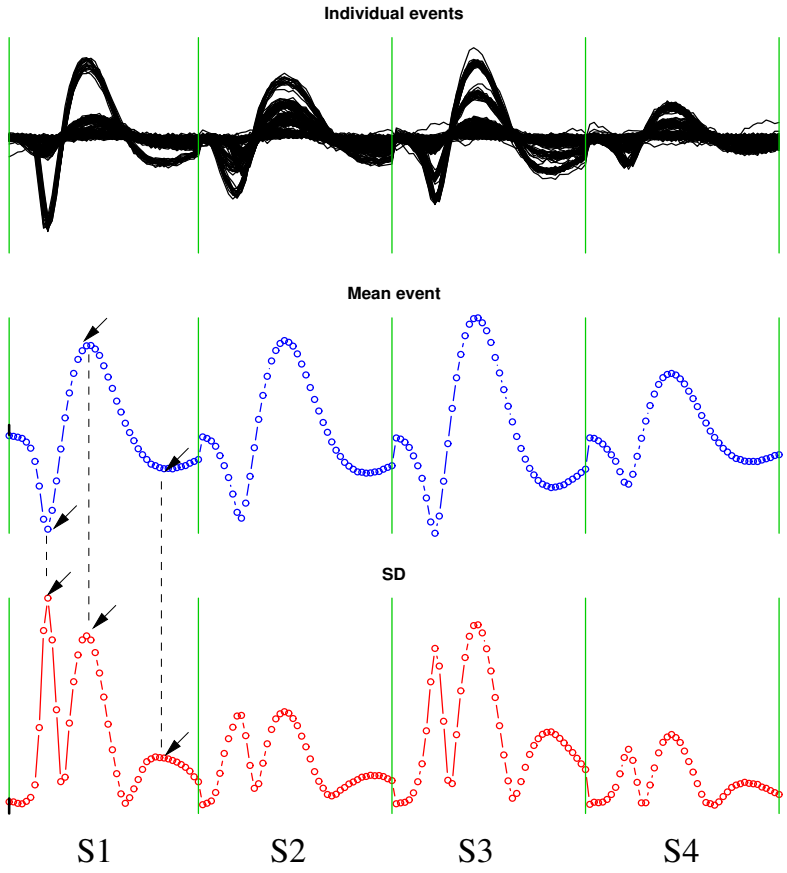

Figure 4.3: **Détection des potentiels d'action (événements).** *Haut*: les événements détectés sur les 4 premiers sites de l'électrode sont superposés. Chaque événement est constitué par la mise bout à bout des formes qu'il prend sur les 4 sites (S1 à S4, délimités par les 4 lignes verticales vertes). *Milieu*: événement moyen. *Bas*: déviation standard. Il y a 45 points d'échantillonnage par site, soit 180 au total. On réduit la dimensionalité de cette représentation en ne conservant que 3 points par site, à savoir ceux dont la déviation standard est la plus grande. Sont marqués d'une flèche les 3 points retenus sur le site 1. Ces mêmes points sont également retenus sur les 3 autres sites dans la représentation finale d'un événement, qui est dès lors de dimension 12. On peut également se contenter d'un seul point par site, le second des trois ci-dessus (représentation de dimension 4). Les index des points retenus constituent le système de coordonnées de l'espace réduit de représentation.

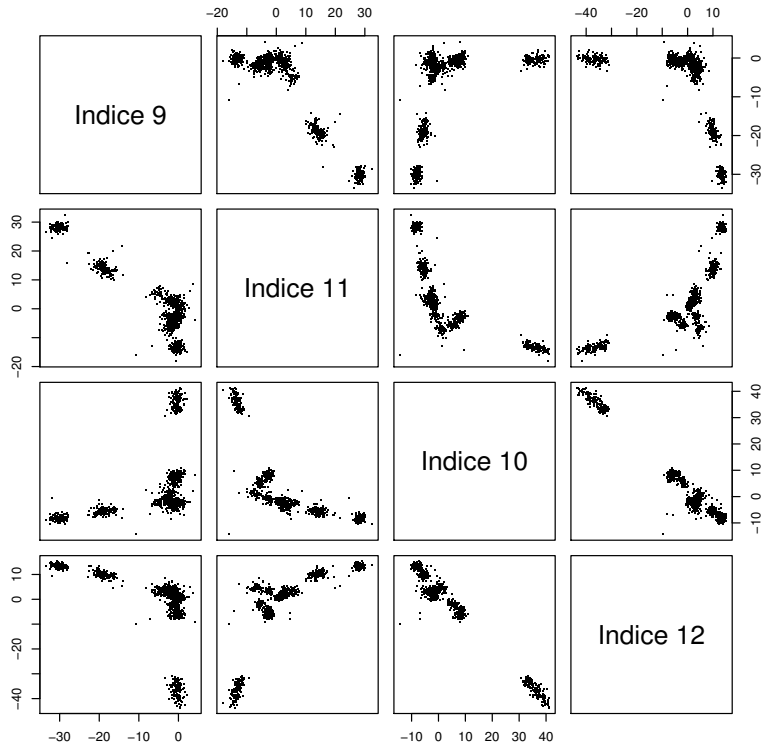

Figure 4.4: **Graphiques de Wilson.** Les événements détectés (Fig. 4.3) sont visualisés dans l'espace réduit de représentation (dimension 12 ou 4), par paires de coordonnées (l'index de la coordonnée est précisé sur la diagonale). On choisit de représenter les paires des 4 coordonnées les plus discriminantes.

matrice de covariance (*i.e* ses propriétés statistiques de second ordre), les variances des événements dues au bruit sont indépendantes et identiques selon toutes les directions. La matrice de covariance du bruit est égale à l'identité et celui-ci est qualifié de "blanc". Cette transformation linéaire des événements permet de simplifier les calculs et de réduire la complexité de nos algorithmes de classification : dans le cas du modèle de mélange (chapitre 3, section 3.3.4), la matrice de covariance $\Sigma_k$de la densité k du mélange devient l'identité ; dans le cas de notre algorithme MCMC, c'est tout notre modèle de génération de données qui se trouve simplifié.

Les événements réduits et blanchis sont visualisés dans leur espace de représentation, à l'aide d'un ensemble de graphiques à 2 dimensions que l'on dénomme "graphiques de Wilson" dans ce travail (Fig. 4.4). Cette représentation fait apparaître les nuages d'événements, plus ou moins séparés.

4.2.2 Le modèle de mélange gaussien multivarié et l'algorithme EM

Ajustement des paramètres du modèle

Le premier modèle utilisé pour réaliser la classification des événements - le *clustering* des événements dans leur espace réduit et blanchi - est un modèle de mélange gaussien multivarié classique, dont on ajuste les paramètres à l'aide d'un algorithme dit d'*Expectation-Maximization* (EM), qui permet la maximisation de la vraisemblance de ce modèle (plus précisément de son logarithme). Ce modèle repose sur les deux hypothèses générales suivantes :

1. Les formes de PAs émis par un neurone donné sont constantes.
2. Les PAs sont corrompus par un bruit blanc gaussien qui s'y ajoute linéairement et qui en est statistiquement indépendant.

La seconde hypothèse nécessite un blanchiment préalable du bruit, comme décrit dans la section 4.2.1. Ce modèle, ainsi que l'algorithme EM lui-même, sont décrits formellement dans le chapitre 3, section 3.3.4. Brièvement, il est constitué d'un mélange de densités de probabilité gaussiennes de SD égale à 1 ; *dans l'espace de représentation réduit et blanchi,* chaque événement est supposé être issu de l'une de ces densités (on parle aussi de composantes du mélange), avec une certaine probabilité. Chaque densité correspond à un neurone. L'algorithme EM est un algorithme itératif qui, à chaque itération, détermine tour à tour la répartition des événements dans ces densités (la partition) et les paramètres de celles-ci : à chaque itération, l'algorithme calcule les probabilités que chaque événement soit issu des différentes densités gaussiennes munies des paramètres calculés sur la partition de l'itération précédente. Les paramètres des densités gaussiennes (leurs centres, puisque leurs matrices de covariance sont égales à l'identité) ainsi que les proportions de chaque densité (le nombre d'événements qui leur sont attribués) sont ensuite mis à jour sur la base de cette nouvelle partition. L'algorithme converge vers un maximum local de la vraisemblance. On le lance donc à partir de plusieurs partitions aléatoires initiales et on conserve le résultat obtenu à partir de celle qui a mené à la plus grande valeur de la vraisemblance. C'est la partition finale des événements qui nous intéresse : à chaque événement est associé la densité gaussienne de laquelle il est le plus probablement issu.

Détermination du nombre de neurones

Cet ajustement des paramètres du modèle de mélange gaussien multivarié est réalisé pour différents nombres de composantes (*i.e* neurones) dans le mélange. La comparaison de ces différents modèles ajustés est effectuée avec le critère d'information bayésienne BIC (voir chapitre 3, section 3.5). Ce critère est la log-vraisemblance du modèle ajusté, pénalisée par sa dimension (le nombre de paramètres libres qu'il comporte). C'est le modèle qui minimise ce critère qui est conservé. En pratique, ce critère s'est révélé d'une utilité limitée sur la plupart des données analysées, en raison notamment de la présence de nuages de points allongés provenant des *bursts* de PC et favorisant, à tort, les modèles ayant de nombreuses composantes (voir ci-dessous, le paragraphe

"Reconstitution des *bursts*"). La détermination du nombre de neurones est donc souvent le fruit d'un choix personnel *a posteriori* : les classifications fournies par l'algorithme EM avec différents nombres de neurones sont comparées à l'aide des tests détaillés dans le paragraphe ci-dessous "Estimation de la qualité de la classification". C'est la meilleure classification qui est retenue.

Classification des événements par *template matching*

Comme évoqué dans le chapitre 3, il n'est pas nécessaire de réaliser l'ajustement des paramètres du modèle sur l'ensemble des événements détectés dans un enregistrement long. On établit donc les valeurs des paramètres du mélange sur un segment d'enregistrement, d'une minute en général. Compte tenu de la fréquence des PCs (de l'ordre de $10\,Hz$ à température ambiante), cette durée est suffisante pour obtenir plusieurs centaines d'événements par neurone. Tous les événements qui ont été assignés à l'une des composantes du modèle de mélange ajusté sont moyennés *dans l'espace initial de leur représentation* pour obtenir l'événement prototypique, ou standard, de cette composante, c'est-à-dire de ce neurone (Fig. 4.5). L'ensemble de ces formes standard est utilisé pour classer les événements détectés dans le reste de l'enregistrement par *template matching* (voir chapitre 3, section 3.1.3) : chaque événement de l'enregistrement est associé à la forme standard la plus proche, au sens de la distance euclidienne.

Cette classification des événements par *template matching* est peu coûteuse en calculs et peut être effectuée sur de grandes quantités d'événements, en particulier sur la totalité d'un enregistrement de plusieurs minutes. Cependant, une lente dérive de l'électrode est inévitable au cours de l'expérience. Parce qu'elle modifie les distances entre les sites d'enregistrement et les cellules enregistrées, cette dérive est la cause d'un changement léger et progressif des formes des événements. Il est possible de suivre cette dérive lente en effectuant la classification par *template matching* sur des segments de données courts et successifs (une ou deux minute(s) par segment). Tous les événements associés à un neurone donné dans ce segment sont moyennés, de façon à mettre à jour la forme standard de ce neurone. Ce jeu de formes standard mises à jour est utilisé pour effectuer la classification sur le segment suivant de données.

Une fois la classification des PAs effectuée sur les segments successifs de données, les trains de PAs des neurones identifiés sur ces segments sont concaténés. On dispose alors du train de PAs de chaque neurone sur la totalité de l'enregistrement.

Estimation de la qualité de la classification

Les tests exposés au chapitre 3, section 3.7, permettent d'évaluer la pertinence et la qualité du *spike-sorting* effectué. La visualisation directe des événements labélisés fournie une estimation qualitative précieuse. Cette visualisation directe est faite de trois manières différentes : (i) tous les événements associés à un neurone sont superposés (Fig. 4.5), (ii) la trace brute des données est visualisée avec le numéro de neurone assigné à chaque événement sous forme de code de couleurs ou de symboles associés (Fig. 4.6), (iii) dans les graphiques de Wilson, les événements forment des nuages et sont colorés suivant le label qui leur a été attribué (Fig. 4.7).

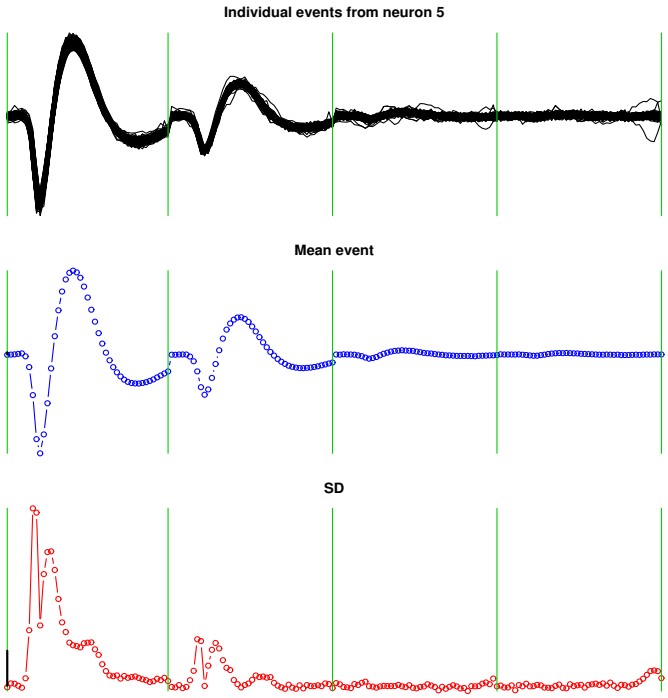

Figure 4.5: **Evénements du neurone 5** (neurone bleu ciel dans les Fig. 4.6 et 4.7)**.** *Haut*: tous les événements attribués au neurone 5 sont superposés. *Milieu*: événement moyen du neurone 5. Cet événement moyen est utilisé comme événement standard de ce neurone pour classer les événements d'enregistrements ultérieurs par *template matching*. Dans cette dernière procédure, tous les neurones sont représentés par leur événement moyen. *Bas*: déviation standard des événements du neurone 5.

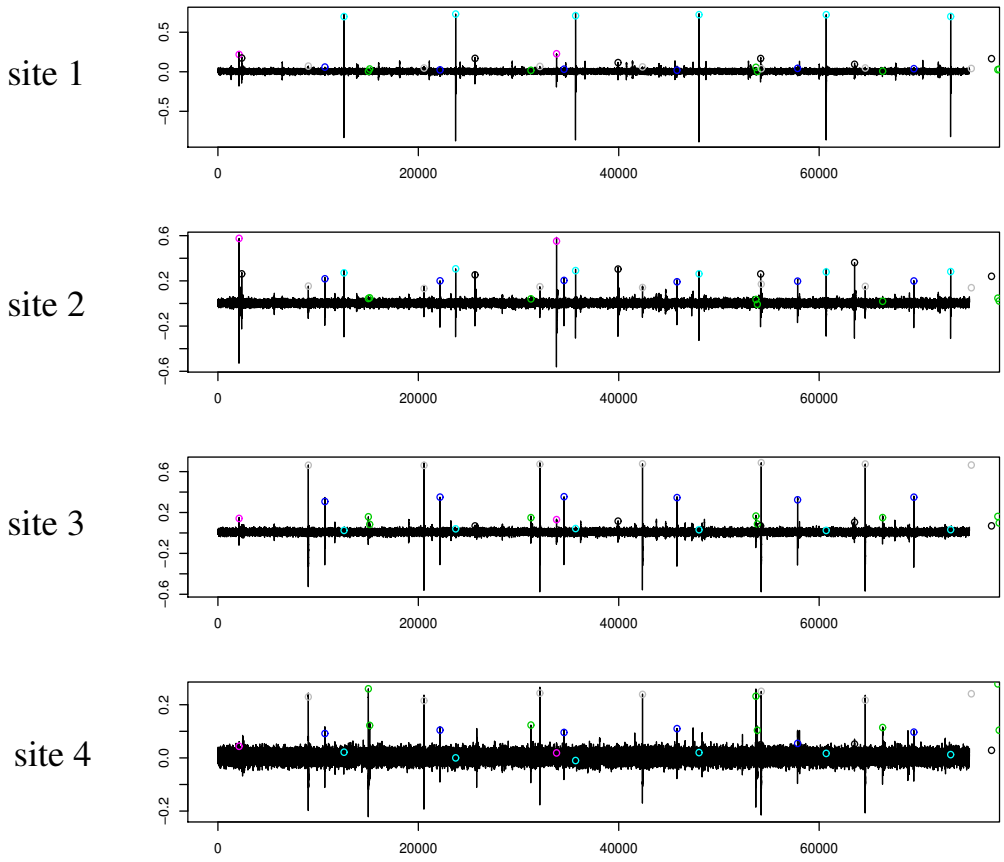

Figure 4.6: **Données brutes labélisées.** Un cercle de couleur est tracé au-dessus de chaque événement pour indiquer son neurone d'origine. Les 5 secondes montrées sont celles de la Fig. 4.2. Echelle des abscisses en période d'échantillonnage: $15000 \equiv 1\,s$. Echelle des ordonnées: $0.5 \equiv 0.25\,mV$.

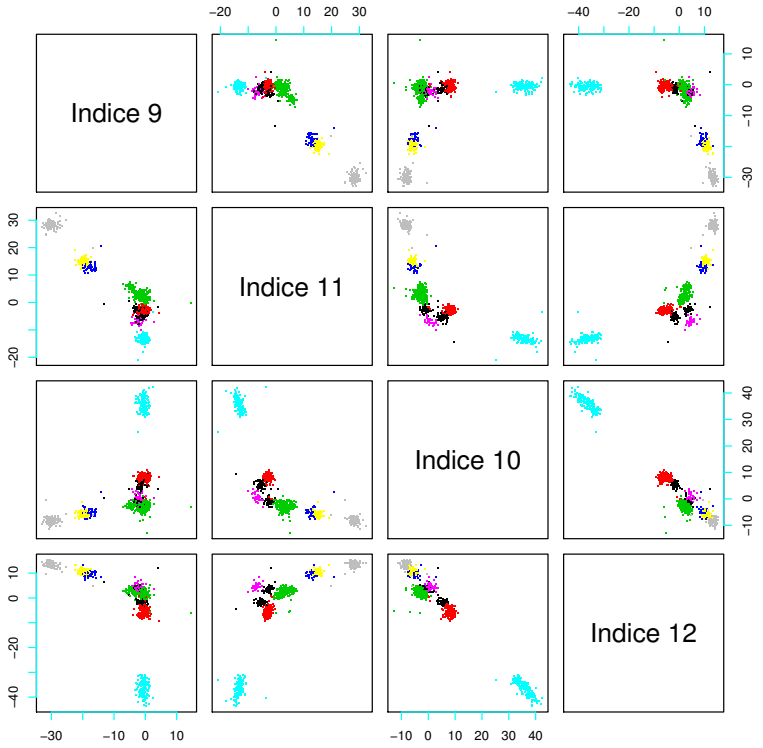

Figure 4.7: **Graphiques de Wilson après classification des événements.** Les événements sont visualisés dans l'espace réduit de représentation comme dans la Fig. 4.4, mais cette fois avec un code de couleur relatif à leur labélisation une fois le *spike-sorting* (EM ou MCMC) effectué. La couleur de chaque événement indique le neurone qui lui a été attribué (une couleur par neurone).

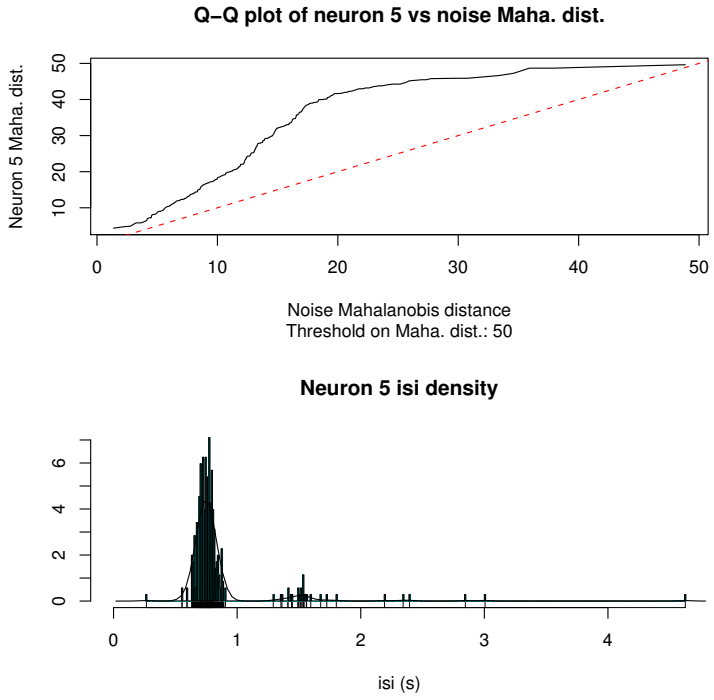

Figure 4.8: **Isolation du neurone.** *Haut*: graphique quantile-quantile (*Q-Q plot*) opposant, dans l'espace blanchi, la distribution des carrés des distances (euclidiennes) des événements du neurone 5 à son événement moyen, à la distribution des carrés des distances à l'origine d'événements de bruit. Le carré de la distance euclidienne dans l'espace blanchi est également appelée *distance de Mahalanobis*. La première bissectrice (pointillés) représente l'égalité stricte des deux distributions. *Bas*: histogramme des ISIs du neurone 5.

Par ailleurs, les histogrammes d'intervalles entre événements (*inter-spike intervals*, ISIs) de tous les neurones sont tracés (Fig. 4.8 bas) et les fréquences moyennes de décharge calculées. L'absence d'ISIs inférieurs à $5\,ms$ est le signe d'une bonne isolation de neurone (période réfractaire). Plus encore, on peut comparer la forme générale de l'histogramme à ceux obtenus à partir d'enregistrements de PCs uniques effectués séparément dans les mêmes conditions.

Enfin, la SD des événements de chaque neurone reconstitué est tracée (Fig. 4.5 bas). Pour un neurone dont les événements ont bien été isolés, la SD doit idéalement rester constante sur toute la durée d'un événement. Cependant, on constate que la SD de neurones bien isolés présente tout de même des pics aux endroits de plus fortes pentes du potentiel d'action (Fig. 4.5 bas) : ceci est une conséquence inévitable de l'échantillonnage (*sampling jitter*) : à ces temps-là du potentiel d'action, la variation de potentiel est très importante sur une seule période d'échantillonnage. Par conséquent, les valeurs prises aux points d'échantillonnage des pentes ascendantes et descendantes du potentiel d'action s'étendent sur une gamme bien plus importante que partout ailleurs, augmentant ainsi artificiellement la SD en ces points. En revanche, la présence d'événements

étrangers dans le train reconstruit d'un neurone est à l'origine d'importantes valeurs de la SD localisées en d'autres points d'échantillonnages.

La distribution des carrés des distances euclidiennes entre les événements d'un neurone et leur moyenne dans l'espace blanchi (distance dite de Mahalanobis) est comparée à celle des carrés des distances à l'origine d'événements de bruit sur un graphique quantile-quantile (*Q-Q plot*, Fig. 4.8 haut). Ce test (chapitre 3, section 3.7.2) met en évidence l'écart qui existe entre le bruit réel associés aux événements d'un neurone donné et le bruit réel situé entre les événements. Si cet écart est plus important pour certains neurones, leurs trains reconstitués doivent être considérés comme un mélange de plusieurs décharges différentes.

Ces différents tests permettent d'éliminer les neurones les moins bien séparés. En pratique, il s'agit systématiquement de neurones dont les PAs sont de faibles amplitudes, à la limite du seuil de détection imposé. Ces tests permettent également d'identifier les neurones dont les décharges ont été les mieux isolées et ceux dont les décharges sont légèrement "polluées" par la présence de quelques événements qui ne leur appartiennent pas.

Reconstitution des *bursts* par groupement de nuages

Comme il a été dit dans le chapitre 2, les PCs sont capables d'émettre spontanément des *bursts* de PAs d'amplitudes décroissantes, en particulier chez le rat jeune. Ces événements donnent naissance à des nuages de points très allongés, et même souvent séparés, dans l'espace de leur représentation, visualisé par les graphiques de Wilson. La méthode de délimitation de nuages gaussiens utilisée est donc condamnée à les séparer. Les PAs d'un *burst* sont nécessairement considérés comme venant de neurones différents.

Les tests ci-dessus, ainsi que les histogrammes de corrélation croisée entre neurones (voir section 4.3), permettent de reconstituer ces *bursts* en regroupant les nuages correspondants sur la base de quatre constatations : (i) les événements de ces nuages sont enregistrés sur le même site ; (ii) les histogrammes de corrélation croisée (voir section 4.3) entre les décharges de ces nuages pris par paires présentent un pic significatif important situé entre 5 et $10\,ms$, ou encore plusieurs pics successifs à des multiples de ce délai ; cette durée est typique des ISIs au sein des *bursts* de PCs ; (iii) les événements de ces nuages se succèdent toujours par ordre d'amplitude décroissante ; (iv) les rapports d'amplitudes entre événements de ces nuages sont conservés sur les sites voisins et sont comparables au rapport d'amplitude des PAs de *bursts* enregistrés sur des PCs individuelles (de l'ordre de 0.5). Enfin, on constate également que ces corrélations étroites à $5-10\,ms$ demeurent entre ces nuages en présence de bicuculline. Elles ne sont donc pas induites par les interneurones.

Dans ces conditions, les événements de ces nuages sont considérés comme issus de la même PC et sont donc regroupés *a posteriori* : les vecteurs des temps des événements de ces nuages sont concaténés et réordonnés par temps croissant. Cependant, cette opération de regroupement introduit potentiellement des erreurs. Prenons le cas simple de doublets (*bursts* consitutés de deux PAs uniquement). Il est fréquent qu'existe dans l'enregistrement une PC émettant des PAs d'amplitudes similaires au second (petit) PA de ces doublets. Dans ce cas, le nuage auquel les petits PAs des doublets sont assignés

contient également les PAs de cette autre PC. En groupant tous les PAs de ce nuage aux PAs du nuage correspondant aux premiers (grands) PAs des doublets, on attribue donc à la PC émettant des doublets des PAs d'une autre origine. La Fig. 4B2 du deuxième article illustre parfaitement ce problème : ce mêlent aux *bursts* de la cellule référence, les PAs d'une autre cellule en arrière plan. Ceux-ci sont indiscernables des PAs de la queue des *bursts* de notre cellule d'intérêt.

Un deuxième problème lié à la présence de ces *bursts* est celui de la détermination du nombre de neurones. En effet, l'expérience montre qu'un modèle multipliant le nombre de composantes pour rendre compte d'un nuage allongé est systématiquement privilégié par le critère BIC. De ce fait, la détermination du nombre de neurones à l'aide de ce critère est impossible sur de telles données : le BIC favorise une sur-partition des données (*overclustering*). C'est la raison pour laquelle le choix du nombre de neurones repose quasiment systématiquement sur mon évaluation propre : je compare la qualité des classifications obtenues avec différents nombres de neurones à l'aide des tests décrits plus haut et choisit la meilleure sur la base de ceux-ci.

4.2.3 Le modèle de Markov caché dynamique et l'algorithme MCMC

Le second modèle de génération de données est plus compliqué et plus réaliste. Il repose sur les trois hypothèses générales suivantes :

1. La séquence des temps des PAs émis par un neurone donné est la réalisation d'un processus ponctuel de Markov caché (Camproux et al., 1996 ; Güçlü et Bolanowski, 2004).
2. L'amplitude d'un PA émis par un neurone dépend du temps qui le sépare du précédent PA de ce neurone.
3. L'amplitude mesurée d'un PA est corrompue par un bruit blanc gaussien qui s'ajoute linéairement au PA et en est indépendant.

La troisième hypothèse nécessite un blanchiment préalable du bruit, comme décrit dans la section 4.2.1. Ce modèle de génération d'un train de PAs comprend deux volets. Le premier concerne la statistique de décharge de ce train et le second la dynamique d'amplitude des PAs, c'est-à-dire la dépendance de leur amplitude vis-à-vis des ISIs.

La densité des ISIs

La densité d'ISIs empirique des neurones enregistrés est modélisée par un modèle de Markov caché (*Hidden Markov Model*, HMM) à 3 états. Dans cette perspective, une séquence d'ISIs (*i.e* un train de PAs) est considérée comme la séquence *observable* d'une séquence d'*états cachés* du neurone. Cette dénomination particulière vient du fait que ces états ne sont pas directement observables dans les données ; la seule quantité qui le soit est la valeur de l'ISI que cet état produit. En d'autres termes, la densité de probabilité de laquelle est tirée un ISI dépend de l'état sous-jacent. Dans notre cas, la densité d'ISIs de chaque état est une densité log-normale, caractérisée par deux

paramètres : un paramètre d'échelle s (en secondes) et un paramètre de forme σ (sans dimension). Cette densité est donnée par :

$$f(isi) = \frac{1}{isi \cdot \sigma \cdot \sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2} \cdot (\frac{\log(\frac{isi}{s})}{\sigma})^2\right) \tag{4.1}$$

La densité log-normale est une alternative pertinente à la densité exponentielle qui est habituellement utilisée pour modéliser les trains de PAs (c'est le cas, en particulier, du modèle de mélange gaussien de la section 4.2.2 qui fait l'hypothèse implicite d'une décharge Poisson). Elle est unimodale, possède une période réfractaire, croît vite et décroît lentement. Le modèle de Markov caché permet de rendre compte de distributions d'ISIs multimodales.

Toutes les transitions entre états du modèle de Markov caché sont possibles. En plus des 3 paires de paramètres d'échelle et de forme des 3 densités log-normales, il y a donc 9 paramètres de transitions entre états, dont 6 indépendants (la somme des transitions d'un état donné vers lui-même et vers les deux autres est égale à 1). 12 paramètres libres sont donc requis pour modéliser la statistique de décharge d'un neurone.

La Fig. 1A du deuxième article propose un schéma résumant ce modèle. Dans un train de PAs, chaque événement (ou ISI) est émis par l'une des 3 densités de probabilité, en fonction de l'état dans lequel se trouve le neurone : s'il est dans l'état 1 (resp. 2, 3), il donne naissance à un ISI court (resp. intermédiaire, long) à partir de la densité rouge (resp. bleue, verte). La transition vers n'importe quel état, y compris le même, est possible, comme indiqué par les flèches entre états. Une présentation plus formelle du modèle de Markov caché se trouve dans l'appendice du deuxième article (section A.1.1 de cet article).

Nous avons choisi un modèle à 3 états sur la base des enregistrements en régime spontané de PCs effectués en configuration cellule-attachée lâche. Les PCs émettant des *bursts* présentent en effet 3 types d'ISIs : les ISIs au sein des *bursts* (de l'ordre de $5 - 10\,ms$), les ISIs entre *bursts* (de l'ordre de $60 - 90\,ms$) et les pauses plus longues, supérieures à $100\,ms$. Les sections 3.1 et 3.2 du deuxième article montrent respectivement ce type de décharge en *bursts* et la qualité de l'ajustement du modèle de Markov caché à ces données (Fig. 1, 2 et 7 dudit article). Ce modèle à 3 états peut également être ajusté à des densités d'ISIs bimodales ou unimodales. Dans ce cas, l'un (ou deux) des états est vide et ne contient pas, ou très peu, d'événements. Il n'a jamais était nécessaire de considérer un plus grand nombre d'états.

Enfin, il est important de préciser que nous ne cherchons aucunement à établir un lien entre ces états de Markov cachés, formels, et des états physiologiques réels du neurone. Ce modèle est un outil statistique descriptif qui permet de rendre compte des principales caractéristiques de la statistique de décharge des cellules enregistrées. Il ne s'agit pas pour nous de justifier ces états formels, ni de les fonder sur des propriétés biophysiques particulières de ces cellules.

La dynamique d'amplitude des PAs

Les événements sont décrits par leur amplitude au pic sur chacun des 4 sites considérés. On modélise la dépendance de ces amplitudes vis-à-vis de l'ISI par une relaxation

exponentielle qui s'écrit (Fee et al., 1996) :

$$\mathbf{A}(isi) = \mathbf{P} \cdot (1 - \delta \cdot \exp(-\lambda\, isi)) \tag{4.2}$$

où λ est l'inverse d'une constante de temps de relaxation (en $1/s$), $\mathbf{P}$ est le vecteur (de dimension 4) de l'amplitude maximale de l'événement sur chaque site d'enregistrement (c'est donc l'amplitude observée lorque $isi \gg \lambda^{-1}$) et δ est la modulation maximale de cette amplitude (pour $i \ll \lambda^{-1}$, on a $\mathbf{A}(isi) \approx \mathbf{P} \cdot (1 - \delta)$). Ce modèle suppose donc que la modulation d'amplitude d'un événement est la même sur les 4 sites d'enregistrement. C'est ce qui est effectivement observé dans nos expériences et dans la littérature (Gray et al., 1995). Ce volet de notre modèle de génération de données comprend donc 6 paramètres pour 4 sites d'enregistrement (3 paramètres dans le cas d'un seul site). Ajoutés aux 12 paramètres du volet concernant la statistique de décharge, notre modèle complet contient 18 paramètres par neurone pour 4 sites d'enregistrement (15 paramètres dans le cas d'un seul site).

La section 3.3 du deuxième article montre la qualité de l'ajustement de ce modèle aux données de PCs individuelles (Fig. 3 de cet article). On y montre que ce modèle est une approximation valable de la dynamique d'amplitude des PCs, mais qu'il ne rend pas compte de tous les aspects de cette dynamique. La SD du bruit (2.64) est plus grande qu'attendu après blanchiment (1) et notre troisième hypothèse générale sur le bruit blanc gaussien n'est donc qu'une approximation. En particulier, il existe une décroissance faible de l'amplitude maximale pour les longs ISIs, alors que celle du modèle est constante.

L'algorithme MCMC

L'inférence statistique sur ce modèle est réalisée par un algorithme de *Monte Carlo par Chaînes de Markov* (MCMC). L'exposé détaillé des fondements mathématiques et de l'algorithme lui-même fait l'objet du premier article. Dans les paragraphes qui suivent, je m'attache plutôt à décrire le déroulement complet de la procédure de *spike-sorting* effectué sur la base de cet algorithme. Pour comprendre cette procédure, il est nécessaire de rappeler brièvement les grandes lignes de l'approche MCMC.

Son principe est de construire une chaîne de Markov (qui ne doit pas être confondue avec la séquence d'états d'un neurone de notre modèle de Markov caché décrit précédemment : ces états du neurone sont dits HMM dans ce qui suit) dont l'espace $\mathcal{S}$ d'évolution est le produit de deux espaces : l'espace des paramètres du modèle de génération de données présenté ci-dessus et l'espace des "configurations" des trains de PAs. Une configuration est ici définie par l'attribution à chaque PA des données d'un neurone d'origine (ou encore "label") *et* d'un état HMM de ce neurone. Ainsi, dans cet espace $\mathcal{S}$, un état de notre chaîne de Markov est défini par deux vecteurs : un vecteur θ des paramètres du modèle et un vecteur C de la configuration de l'ensemble des PAs analysés. Le vecteur θ est de dimension $18 \cdot K$ pour 4 sites d'enregistrement, où K est le nombre de neurones dans le modèle (voir la présentation du modèle ci-dessus) ; la dimension du vecteur C est égale au double du nombre de PAs présents dans les données soumises au *spike-sorting,* puisqu'à chaque PA sont attribués un label et un état HMM. L'algorithme construit cette chaîne de Markov de manière à échantillonner

l'espace $\mathcal{S}$ selon la densité postérieure $\pi_{post}(\theta, C \mid Y)$ des paramètres du modèle et des configurations étant données les données Y : à chaque pas t de l'algorithme, on tire un nouvel état $\left[\theta^{(t)}, C^{(t)}\right]$ de la chaîne de Markov à partir de l'état de la chaîne au pas $t-1$, $\left[\theta^{(t-1)}, C^{(t-1)}\right]$. Ce tirage est réalisé suivant la procédure décrite ci-dessous. L'appendice du premier article montre que cette procédure assure la convergence de la chaîne de Markov vers une distribution unique donnée par $\pi_{post}(\theta, C \mid Y)$. L'échantillonnage de l'espace $\mathcal{S}$ selon cette postérieure nous donne accès à la distribution postérieure de chaque paramètre du modèle ainsi qu'à celle des labels de chaque PA (voir le paragraphe ci-dessous "Analyse de la sortie de l'algorithme"). Le nombre K de neurones est choisi et fixé d'emblée.

Tirage d'un nouvel état de la chaîne de Markov

Etat initial de la chaîne de Markov Les amplitudes maximales au pic des K neurones sur les 4 sites sont initialisées à l'aide des valeurs des 4 amplitudes au pic de K événement tirés aléatoirement parmi les N événements détectés. Le paramètre δ est initialisé à sa valeur minimale et la valeur initiale du paramètre λ est fixée à l'inverse du temps de l'enregistrement. Les valeurs initiales des paramètres d'échelle et de forme des 3 log-normales de chaque neurone sont aléatoirement tirés selon leur densité *a priori* (plate sur leur intervalle autorisé). Les paramètres de transition entre états HMM de chaque neurone sont eux initialisés à 1/3. Enfin, la configuration initiale est obtenue en tirant aléatoirement, pour chaque événement, un label parmi les K possibles, avec probabilité $1/K$.

Au pas t de l'algorithme, on obtient l'état $\left[\theta^{(t)}, C^{(t)}\right]$ à partir de l'état $\left[\theta^{(t-1)}, C^{(t-1)}\right]$ en tirant successivement chaque paramètre du modèle et pour chaque PA, un label et un état HMM, selon les procédures décrites dans ce paragraphe. Pour éviter d'alourdir les notations, l'exposant (t) indexant le pas de l'algorithme est omis dans ce qui suit. On note C_{-i} la configuration spécifiant les labels et les états HMM de tous les PAs excepté le PA d'index i. De même, on note θ_{-a} le vecteur spécifiant tous les paramètres du modèle, excepté le paramètre a. Chaque paramètre a du modèle a une densité *a priori* uniforme sur un segment $[a_{min}, a_{max}]$ pertinent pour ce paramètre. Un pas de l'algorithme est terminé une fois que tous les labels et états HMM, ainsi que tous les paramètres du modèle, ont été tirés comme décrit ci-dessous. Ceci définit le nouvel état de la chaîne de Markov.

Labels et états HMM Pour chaque PA i du train de PAs, on tire un label $l_i \in \{1, ..., K\}$ et un état HMM $m_i \in \{1, 2, 3\}$ selon leur densité postérieure conditionnelle :

$$l_i, m_i \mid Y, \theta, C_{-i} \sim \frac{\pi_{post}(l_i, m_i \mid Y, \theta, C_{-i})}{\sum_{l_j, m_j} \pi_{post}(l_j, m_j \mid Y, \theta, C_{-i})} \tag{4.3}$$

Paramètres d'amplitude Pour chaque neurone q, on tire les paramètres d'amplitude $(P_{q,1}, P_{q,2}, P_{q,3}, P_{q,4}, \delta_q, \lambda_q)$ avec un pas de Metropolis-Hastings ; la nouvelle valeur

proposée pour chaque paramètre est tirée selon une approximation linéaire par morceaux de leurs densités conditionnelles postérieures respectives. Prenons le cas de λ_q pour illustrer la procédure. Soient $\pi_{post}\left(\lambda_q \mid Y, \theta_{-\lambda_q}, C\right)$ sa densité conditionnelle postérieure et $\tilde{\pi}_{approx}\left(\lambda_q \mid Y, \theta_{-\lambda_q}, C\right)$ son approximation linéaire par morceaux. Soit λ la valeur actuelle de λ_q.

Tout d'abord, la nouvelle valeur proposée pour λ, notée $\tilde{\lambda}$, est tirée selon la densité :

$$\tilde{\lambda} \mid Y, \theta_{-\lambda_q}, C \sim \tilde{\pi}_{approx}\left(\lambda_q \mid Y, \theta_{-\lambda_q}, C\right) \tag{4.4}$$

Puis, cette valeur $\tilde{\lambda}$ est acceptée avec la probabilité A donnée par :

$$A = \min\left(1, \frac{\pi_{post}\left(\tilde{\lambda} \mid Y, \theta_{-\lambda_q}, C\right) \cdot \tilde{\pi}_{approx}\left(\lambda \mid Y, \theta_{-\lambda_q}, C\right)}{\pi_{post}\left(\lambda \mid Y, \theta_{-\lambda_q}, C\right) \cdot \tilde{\pi}_{approx}\left(\tilde{\lambda} \mid Y, \theta_{-\lambda_q}, C\right)}\right) \tag{4.5}$$

Si $\tilde{\lambda}$ est accepté, alors $\lambda \rightarrow \tilde{\lambda}$.

Sinon $\lambda \rightarrow \lambda$.

Les densités conditionnelles postérieures de $\mathbf{P}$ et δ étant gaussiennes, il n'est pas nécessaire d'avoir recours à leur approximation linéaire par morceaux evoquée ci-dessus pour ces paramètres. En revanche, cela reste nécessaire pour le paramètre λ.

Paramètres des densités log-normales

paramètres d'échelle Pour chaque neurone q et chaque état HMM r ($r \in \{1, 2, 3\}$) du neurone q, le paramètre d'échelle de la densité log-normale est notée s_q^r.

En premier lieu, on tire u selon :

$$u \mid Y, \theta_{-s_q^r}, C \sim Norm\left(\overline{\log i_q}, \frac{\left(\sigma_q^r\right)^2}{n_q}\right) \tag{4.6}$$

où n_q est le nombre d'ISIs du neurone q, σ_q^r est le paramètre de forme du neurone q dans l'état HMM r, et $\overline{\log i_q} = \frac{1}{n_q}\sum_{j=1}^{n_q} \log i_{q,j}$, $i_{q,j}$ étant l'ISI d'index j du neurone q.

Alors, si $s = \exp(u) \in [s_{min}, s_{max}]$, on fixe $s_q^r = s$.

Sinon, on tire un autre u.

paramètres de forme Pour chaque neurone q et chaque état HMM r ($r \in \{1, 2, 3\}$) du neurone q, le paramètre de forme de la densité log-normale est notée σ_q^r.

En premier lieu, on tire u selon :

$$u \mid Y, \theta_{-\sigma_q^r}, C \sim Gamma\left(\frac{n_q}{2} - 1, \frac{n_q}{2}\left(\overline{\log i_q} - \log s_q^r\right)^2\right) \tag{4.7}$$

avec les mêmes notations que pour les paramètres d'échelle.

Alors, si $\sigma_{min} \leq \sqrt{1/u} \leq \sigma_{max}$, on fixe $\sigma_q^r = \sqrt{1/u}$.

Sinon on tire un autre u.

Paramètres de transition Pour chaque q, les paramètres de transition entre les 3 états HMM de chaque neurone forment une matrice carrée 3 par 3 dont les 3 lignes sont tirées successivement.

Soit $m = (m_1, ..., m_N)$ la configuration du train de PAs du neurone q, où m_k est l'état HMM du PA k de ce neurone. Soit $\mathrm{n_{ij}}$ le nombre de PAs de ce neurone qui sont dans l'état m_j après un PA de ce neurone dans l'état m_i. On tire la ligne i de cette matrice de transition selon la distribution de Dirichlet $\mathcal{D}_3(1+n_{i1}, 1+n_{i2}, 1+n_{i3})$ (Robert and Casella, 1999).

Analyse de la sortie de l'algorithme

Avant que la chaîne de Markov ne parvienne à l'équilibre, *i.e* n'échantillonne son espace $\mathcal{S}$ d'évolution selon sa distribution stationnaire, en l'occurrence la densité postérieure $\pi_{post}(\theta, C \mid Y)$, les états successifs qu'elle visite dépendent de l'état initial dans lequel elle a débuté. L'ensemble de ces pas successifs effectués par l'algorithme avant de parvenir à l'équilibre constituent une période dite de *burn-in*, qu'on ne considère pas dans l'exploitation de ses résultats. Les *diagnostics de convergence* ont pour objet de s'assurer de façon empirique que la chaîne a atteint l'équilibre. En pratique, on se contente de surveiller l'évolution, au fil des pas de l'algorithme, des paramètres du modèle ou de tout autre fonction de ces paramètres, telle l'énergie[3]. Cette dernière décroît de pas en pas, à mesure que l'ajustement des paramètres du modèle aux données s'améliore (Fig. 4.9). Lorsque cette décroissance cesse, ou semble cesser (voir le paragraphe ci-dessous "La méthode d'échange de réplique"), l'énergie reste à peu près constante si on laisse l'algorithme progresser. On considère alors que l'équilibre a été atteint, l'algorithme a convergé : la chaîne de Markov échantillonne alors sa densité cible $\pi_{post}(\theta, C \mid Y)$. C'est pourquoi, avant toute analyse ultérieure, on commence par tracer l'évolution de l'énergie en fonction des pas de l'algorithme, afin de s'assurer qu'elle ne décroît plus.

Une fois que la chaîne de Markov a évolué un certain nombre de pas à l'équilibre, il est possible d'estimer les valeurs des paramètres du modèle, celles des labels des PAs, ainsi que les SDs de ces estimations. Pour cela, on calcule la moyenne d'un paramètre donné θ_i à partir des valeurs qu'il prend sur N_T pas de l'algorithme qui suivent N_D pas de *burn-in*, nécessaires pour atteindre l'équilibre :

$$\overline{\theta_i} = \frac{1}{N_T - N_D} \sum_{t=N_D}^{N_T} \theta_i^{(t)} \tag{4.8}$$

Cependant, les états successifs de la chaîne de Markov construits par notre algorithme à l'équilibre sont corrélés : les valeurs prises par un paramètre θ_i au cours des pas successifs de l'algorithme ne sont pas indépendantes. Ceci vient du fait que les états successifs visités par notre chaîne ne sont pas directement et indépendamment tirés de

[3]L'énergie E est définie par : $E(\theta, C) = -log\,[L(Y, C \mid \theta)\pi_{prior}(\theta)]$, où L est la vraisemblance et π_{prior} la densité *a priori* des paramètres, plate sur l'intervalles pertinent pour chaque paramètre. Une fois la configuration (les labels) établie lors d'un pas de l'algorithme et une fois tirées les nouvelles valeurs des paramètres, on peut calculer la valeur de L, et en déduire celle de E.

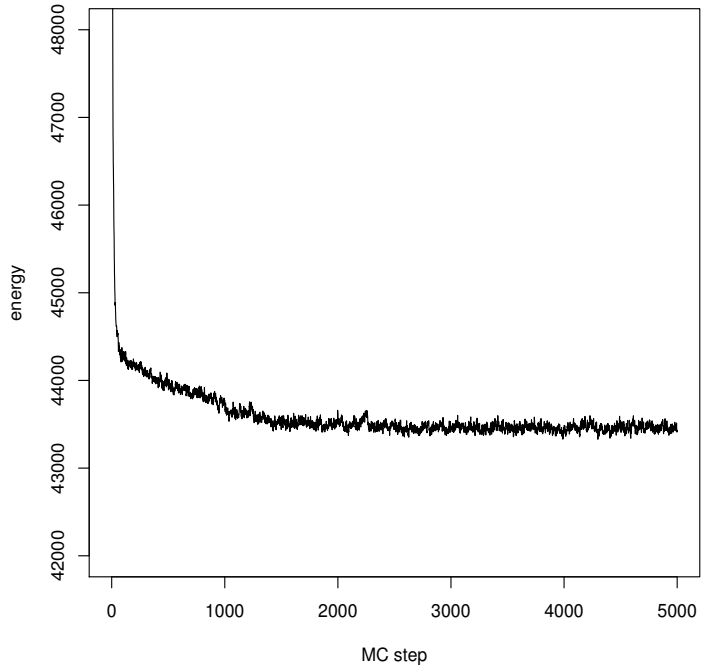

Figure 4.9: **Evolution de l'énergie:** valeurs de l'énergie des états successifs de la réalisation de la chaîne de Markov (un état à chaque pas de l'algorithme, *MC step*).

la densité d'intérêt $\pi_{post}(\theta, C \mid Y)$. Par conséquent, la SD de ce paramètre doit être corrigée pour cette autocorrélation (Janke, 2002 ; Sokal, 1989). La correction apportée est obtenue en multipliant la variance empirique $\sigma^2(\theta_i)$ de ce paramètre par le temps d'autocorrélation intégré $\tau_{autoco}(\theta_i)$:

$$\tau_{autoco}(\theta_i) = \frac{1}{2} + \sum_{l=1}^{L} \frac{\rho(l;\theta)}{\rho(0;\theta)} \tag{4.9}$$

où L est le nombre de pas nécessaires pour que la fonction d'autocorrélation ρ de θ_i oscille autour de 0, ρ étant donnée par :

$$\rho(l;\theta_i) = \frac{1}{N_T - N_D - 1} \sum_{t=N_D}^{N_T} \left(\theta_i^{(t)} - \overline{\theta_i}\right) \left(\theta_i^{(t+l)} - \overline{\theta_i}\right) \tag{4.10}$$

La variance de θ_i , $Var(\theta_i)$, corrigée pour l'autocorrélation est alors égale à :

$$Var(\theta_i) = 2 \cdot \tau_{autoco} \cdot \sigma^2(\theta_i) \tag{4.11}$$

où la variance empirique $\sigma^2(\theta_i)$ est donnée par :

$$\sigma^2(\theta_i) = \frac{1}{N_T - N_D} \sum_{t=N_D}^{N_T} \left(\theta_i^{(t)} - \overline{\theta_i}\right)^2 \tag{4.12}$$

De même, on peut calculer pour chaque PA des données les probabilités relatives d'assignation aux différents neurones sur la base des labels que ce PA reçoit au cours des pas successifs de l'algorithme qui sont considérés. La probabilité qu'un PA j des données, de label l_j, ait le neurone i pour origine est donnée par :

$$p(l_j = i) = \frac{\sum_{k=N_D}^{N_T} \delta_{l_j^{(k)},i}}{N_T - N_D} \tag{4.13}$$

où $\delta_{l_j^{(k)},i}$ est le symbole de Kronecker (= 1 si $l_j^{(k)} = i$, 0 sinon).

La prise en compte de la statistique de décharge et de la dynamique d'amplitude des PAs dans la classification de ceux-ci constituait le premier point fort de la méthode. On voit ici les autres avantages de la méthode : celle-ci permet d'obtenir une estimation des paramètres du modèle accompagnée de la variance de cet estimateur (égale à $Var(\theta_i)/(N_T - N_D)$). De plus, on dispose d'une classification "douce" : pour chaque événement, l'algorithme fournit la probabilité qu'il ait été émis par chacun des K neurones du modèle. On obtient une classification "dure" en attribuant un seul label à chaque événement, en l'occurrence, le label dont la probabilité est la plus grande.

La méthode d'échange de répliques

Il peut arriver que la fonction énergie évoquée dans le précédent paragraphe possède des minima locaux dans lesquels l'algorithme peut rester bloqué longtemps. Comme on a pu le voir, la dynamique de l'algorithme est essentiellement locale, en ce sens

que les transitions d'un état à un autre se font paramètre par paramètre et PA par PA. Les transitions possibles ne se font donc que dans un certain voisinage de l'état présent. Cette caractéristique de l'algorithme explique pourquoi il est particulièrement sensible aux minima locaux : pour sortir d'un minimum local et en atteindre un autre, il faut passer par une succession d'états peu probables et la probabilité d'une succession d'états peu probables est très faible. Le temps, en terme de pas, que l'algorithme passe à explorer un minimum local peut donc être très long. L'énergie ne décroît plus et l'on peut croire que l'algorithme a convergé.

L'appendice du premier article présente une méthode très utilisée pour remédier à ce problème. Il s'agit de la méthode d'échange de répliques (*Replica Exchange Method*, REM, également connue sous le nom de *Parallel Tempering*). Brièvement, le principe de cette méthode est de simuler différentes chaînes de Markov (appelées "répliques") à différentes températures β_i : chaque chaîne effectue une transition de son état présent vers un nouvel état suivant la procédure détaillée plus haut pour $\beta = 1$. L'échange des états entre les chaînes de deux températures voisines intervient selon une règle qui préserve l'échantillonnage de la chaîne $\beta = 1$ selon $\pi_{post}(\theta, C \mid Y)$. L'idée fondamentale de cette méthode est de faire bénéficier la réplique $\beta = 1$ de l'exploration rapide de l'espace $\mathcal{S}$ par les répliques à plus haute température. En effet, ces dernières franchissent plus facilement les barrières d'énergie entre minima locaux et sont susceptibles, grâce à l'échange des états des répliques, de faire passer la réplique $\beta = 1$ d'un minimum à un autre, beaucoup plus rapidement qu'elle ne pourrait le faire seule. Il est évident que seule cette réplique échantillonne l'espace $\mathcal{S}$ selon la densité $\pi_{post}(\theta, C \mid Y)$; c'est donc la seule à faire l'objet de l'analyse détaillée dans le paragraphe précédent. Les autres répliques sont simplement un moyen de la faire parvenir plus rapidement au minimum de la fonction énergie.

Estimation de la qualité de la classification

Les ressorts de l'estimation de la qualité du *spike-sorting* effectué par cette métode sont inchangés par rapport à ceux décrits dans la section 4.2.2. En particulier, la classification forcée des événements (voir le paragraphe "Analyse de la sortie de l'algorithme") est indiquée sur les données brutes par un code de couleurs et/ou de symboles (Fig. 4A du deuxième article) et on trace les graphiques de Wilson avec un code de couleurs relatif à la labélisation de chaque événement (Fig. 5A du deuxième article).

On construit également l'histogramme des ISIs de chaque neurone, ainsi que son modèle muni des paramètres ajustés trouvés par l'algorithme (voir le paragraphe "Analyse de la sortie de l'algorithme"). Cette superposition du modèle ajusté et de l'histogramme empirique du neurone (premier volet de notre modèle de génération de données) permet d'apprécier visuellement la qualité de cet ajustement ; l'histogramme empirique permet de juger de la qualité de l'isolation des PAs du neurone (Fig. 5B du deuxième article). La simple fréquence moyenne de décharge des neurones reconstitués est un élément pour décider de leur pertinence ou non. On peut également visualiser, pour chaque neurone, la qualité de l'ajustement du deuxième volet de notre modèle de génération de donnée, à savoir la dynamique d'amplitude, en superposant la relaxation exponentielle, munie de ses paramètres ajustés, aux amplitudes au pic de ce neurone sur son site principal

d'enregistrement en fonction des ISIs (Fig. 3 du deuxième article).

Ces méthodes sont précieuses et systématiquement utilisées à l'issue de cette procédure de *spike-sorting*, mais elles n'apportent pas la preuve formelle que les trains de PAs reconstitués sont effectivement ceux des neurones enregistrés. L'objet principal du deuxième article est de tester le *spike-sorting* effectué par la présente méthode en apportant une preuve externe de la qualité de l'isolation d'un neurone particulier. Pour cela, l'activité de l'une des PCs vues par l'électrode multisite est indépendamment enregistrée par une micropipette de *patch* en configuration cellule attachée lâche (voir section 4.1). Les temps des PAs des différents trains reconstitués par notre algorithme sont comparés aux temps des PAs du train, dit "de référence", enregistré par cette micropipette : l'un de ces trains de PAs reconstitués reprend 98% des PAs du train de référence en n'y ajoutant que 1% de PAs qui lui sont étrangers (section 3.4 du deuxième article). Il s'agit d'un test externe incontestable de la qualité de la reconstitution de l'activité de la PC de référence par notre algorithme, et ce, en dépit d'une décroissance notable de l'amplitude de ses PAs au sein des *bursts* qu'elle émet.

Détermination du nombre de neurones

Comme précisé au tout début de l'exposé de cette méthode, le nombre K de neurones dans le modèle est fixé *a priori*. Afin de choisir le nombre de neurones le plus pertinent, l'algorithme est lancé successivement avec différentes valeurs de K. Les classifications obtenues avec ces différents nombres de neurones sont comparées avec les outils décrits dans le paragraphe précédent ("Estimation de la qualité de la classification") et la meilleure classification est retenue.

Classification des événements sur de longues durées d'enregistrement

Comme dans la procédure de *spike-sorting* utilisant l'algorithme EM et décrite dans la section 4.2.2, il n'est pas nécessaire de traiter la totalité d'un enregistrement de plusieurs minutes en une seule analyse. Le temps de calcul requis par l'algorithme MCMC étant proportionnel au nombre d'événements soumis, il est préférable de le lancer sur des segments de données relativement courts. On effectue donc le *spike-sorting* sur un premier segment d'une durée d'une minute. Une fois le nombre de neurones déterminé sur ce segment, on lance l'algorithme sur le segment de données suivant, à partir d'un état initial composé d'une labélisation aléatoire des événements de ce segment et des valeurs de paramètres du modèle telles qu'elles ont été ajustées sur le premier segment de données. Ceci permet de conserver les mêmes numéros de neurones sur les différents segments de données et de converger plus rapidement, dans la mesure où les paramètres du modèle sont déjà ajustés dès le démarrage de la chaîne de Markov. Une fois la classification des PAs réalisée sur les segments de données successifs, on concatène les vecteurs des temps des PAs de chaque neurone de façon à disposer des trains de PAs sur la totalité de l'enregistrement.

4.3 L'analyse des trains de PAs multiples

Les deux méthodes de *spike-sorting* exposées dans les sections 4.2.2 et 4.2.3 permettent d'obtenir un ensemble de trains de PAs dont on suppose qu'ils sont ceux des différentes cellules enregistrées par l'électrode multisite. La durée des PAs étant très courte devant les intervalles de temps qui les séparent, la séquence des temps des PAs d'un neurone est vue comme une série d'événements ponctuels. Comme, de plus, les intervalles entre PAs ne sont pas déterministes mais sont l'objet de fluctuations, un train de PAs peut être considéré comme la réalisation d'un *processus ponctuel stochastique*. Il s'agit alors d'analyser les corrélations entre les séries temporelles correspondant aux différents neurones. Dans cette analyse, on a recours à trois méthodes différentes : (i) la construction des histogrammes de *corrélations croisées* entre toutes les paires possibles de trains de PAs (sous-section 4.3.1) ; (ii) la construction des histogrammes dits des *temps du précédent*, détaillée dans la sous-section 4.3.2 ; (iii) l'examen des corrélations dans les *changements de fréquences de décharge* (sous-section 4.3.3). Ces routines ont été écrites par moi-même dans les environnements Scilab et R (voir section 4.4).

4.3.1 Histogrammes de corrélation croisée

Considérons deux processus ponctuels que nous désignons par leurs fonctions aléatoires de comptage N_1 et N_2 (une fonction aléatoire de comptage $N(t)$ compte le nombre d'événements du processus ponctuel dans l'intervalle $(0, t]$). Soit T leur durée, exprimée en périodes élémentaires d'échantillonnage (par exemple, $T = 15000$ pour une durée de $1s$ échantillonnée à $15\,kHz$). On note respectivement $\{r_i;\, i = 1, \ldots, N_1(T)\}$ et $\{s_i;\, i = 1, \ldots, N_2(T)\}$ les temps des événements des processus N_1 et N_2. On peut construire la variable de comptage $J_{12}^T(u)$:

$$J_{12}^T(u) = \sharp \left\{ (r_i, s_j)\ tel\ que\ \left(u - \frac{b}{2}\right) < (s_j - r_i) < \left(u + \frac{b}{2}\right) \right\} \tag{4.14}$$

où $\sharp\{A\}$ désigne le nombre d'événements dans l'ensemble A. La variable $J_{12}^T(u)$ compte donc le nombre d'événements de N_2 qui ont lieu dans une fenêtre de largeur b située à un intervalle de temps u après un événement de N_1. Elle est connue sous le nom d'*histogramme de corrélation croisée*.

Brillinger (1976) montre que dans le cas de deux processus ponctuels stationnaires indépendants, la racine carrée de la variable aléatoire $J_{12}^T(u)/bT$ est approximativement distribuée selon une densité gaussienne centrée sur la racine carrée produit des intensités moyennes de ces processus f_1 et f_2 :

$$\left(J_{12}^T(u)/bT\right)^{1/2} \sim \mathcal{N}\left(\left(f_1 f_2\right)^{1/2}, (4bT)^{-1}\right) \tag{4.15}$$

On estime f_i $(i = 1, 2)$ par :

$$\hat{f}_i = \frac{N_i(T)}{T} \tag{4.16}$$

On peut donc tracer la valeur attendue de l'histogramme J_{12}^T sous l'hypothèse nulle d'indépendance des deux processus, ainsi que les intervalles de confiance à 95%, égaux à

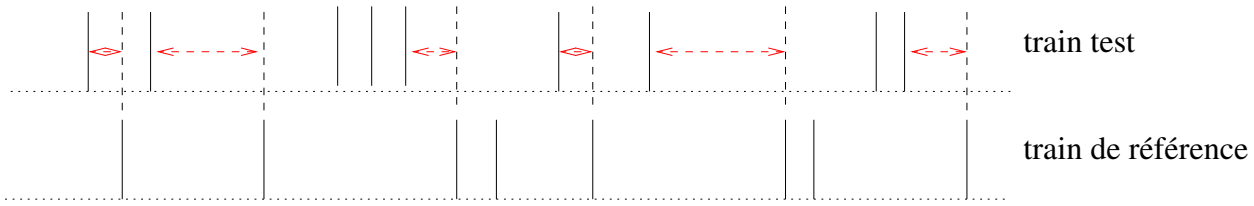

Figure 4.10: **Définition des temps du précédent.** Les intervalles de temps considérés pour construire l'histogramme des temps du précédent d'un train test par rapport à un train de référence sont indiqués par des doubles flèches pointillées horizontales: il s'agit des intervalles de temps séparant les PAs du train de référence des PAs du train test les précédant immédiatement.

$\pm 1.96\,(4bT)^{-1/2}$. Les écarts de l'histogramme empirique à la valeur attendue en cas d'indépendance, s'ils sont supérieurs à l'intervalle de confiance mentionné, sont la preuve de l'existence d'une corrélation temporelle significative entre les deux processus ponctuels.

En pratique, les histogrammes de corrélation croisée sont calculés pour toutes les paires de trains de PAs reconstitués par nos méthodes de *spike-sorting*. Les intervalles de temps u auxquels sont calculés les valeurs de J_{12}^T vont généralement de -100 à $+100\,ms$, avec une largeur de fenêtre b de $1\,ms$. Les corrélations étroites PA à PA doivent apparaître sur cette échelle de temps. Afin de voir des corrélations plus lâches, sur des durées plus longues, on trace également les histogrammes J_{12}^T sur $[-1, +1s]$, avec une largeur de fenêtre de $10\,ms$. Enfin, la durée des enregistrements analysées est en général de 5 minutes, échantillonnées à $15\,kHz$, soit une valeur de T égale à $4.5 \cdot 10^6$.

4.3.2 Histogrammes des temps du précédent

L'histogramme des temps du précédent de deux trains de PAs est l'histogramme des intervalles de temps entre les PAs de l'un des trains (dit "train de référence") et les PAs de l'autre train (dit "train de test") qui les précèdent immédiatement (Fig. 4.10). Sous l'hypothèse nulle que les deux trains de PAs sont deux processus de renouvellement indépendants, la valeur attendue de l'histogramme est donnée par (Johnson et Kiang, 1976) :

$$E[\mathcal{H}(t)] = \upsilon_{test} \cdot \int_t^\infty p_{test}(\tau) d\tau \tag{4.17}$$

où υ_{test} est la fréquence moyenne du train test et p_{test} représente sa densité d'ISIs. La variance de cet histogramme peut être approximée par (Johnson, 1996) :

$$VAR\,(\mathcal{H}(t)) \approx \frac{E[\mathcal{H}(t)]}{N\delta} \tag{4.18}$$

où N est le nombre total d'intervalles dans l'histogramme ($i.e$ le nombre de PAs du train de référence) et δ est la largeur de fenêtre de l'histogramme.

On calcule la valeur attendue de l'histogramme sous l'hypothèse nulle à partir de l'histogramme d'ISIs du train test, utilisé comme estimateur de sa densité d'ISIs p_{test}. On compare cette valeur attendue, assortie de l'intervalle de confiance $\pm 2SD$ donné

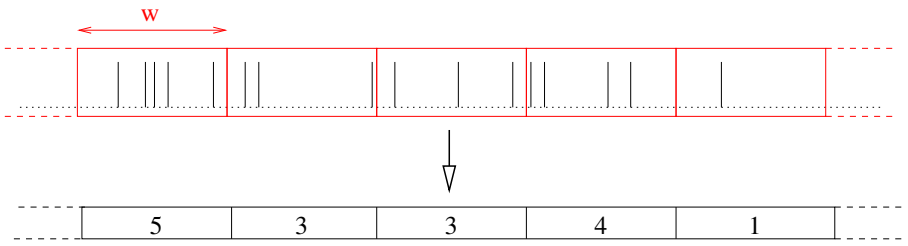

Figure 4.11: **Evolution de la fréquence de décharge au cours du temps:** on compte le nombre de potentiels d'action dans des fenêtres de temps successives, de largeur w, contiguës, sans recouvrement, sur toute la longueur du train. Par la suite, on soustrait au vecteur obtenu sa moyenne et on le normalise par sa déviation standard.

par 4.18, à la valeur réelle de l'histogramme $\mathcal{H}$ calculés à partir des trains test et de référence.

En pratique, on calcule ces histogrammes des temps du précédent pour toutes les paires de trains de PAs reconstitués par nos méthodes de *spike-sorting*, avec des largeurs de fenêtre allant de 2 à 5 ms.

4.3.3 Evolution des fréquences instantanées

Les deux types d'histogramme décrits dans les sections 4.3.1 et 4.3.2 permettent de détecter des corrélations temporelles étroites, PA à PA, entre deux trains. En particulier, ces méthodes sont efficaces pour mettre en évidence une synchronisation des PAs de deux trains.

Cependant, les corrélations temporelles entre les décharges neuronales peuvent être plus diffuses et ne pas se manifester au niveau des temps relatifs des PAs individuels. Il peut exister, par exemple, des corrélations entre les évolutions des fréquences de décharge des cellules au cours du temps. Ces corrélations sont mises en évidence, pour une paire de train de PAs, de la façon suivante.

La fréquence de chaque train est calculée sur des fenêtres de temps successives de largeur w ne se chevauchant pas et couvrant leurs totalités (Fig. 4.11). Aux deux vecteurs obtenus, on soustrait les valeurs moyennes respectives et on les normalise par leurs déviations standard. On définit un *coefficient de corrélation* des deux trains de PAs comme le produit scalaire de ces deux vecteurs. Ce coefficient de corrélation peut être calculé pour différentes valeurs de w.

Pour chaque valeur de w, il est possible de calculer la valeur du coefficient de corrélation défini plus haut lorsque les deux trains de la paire sont indépendants. Pour cela, il suffit d'effectuer une permutation aléatoire de l'un des deux vecteurs avant d'effectuer le produit scalaire. En répétant 200 fois une telle permutation aléatoire de l'un des vecteurs, on obtient 200 valeurs du coefficient de corrélation pour des paires de trains indépendants : ils couvrent la gamme de valeurs prises par le coefficient de corrélation sous l'hypothèse nulle d'indépendance des deux trains. On les utilise pour calculer l'intervalle de confiance correspondant à l'hypothèse nulle d'indépendance des trains : la valeur du coefficient de corrélation des deux vecteurs originaux, correspondant aux

trains de PAs effectifs, doit être comparée à la moyenne de cette gamme de 200 valeurs, augmentée et diminuée de la SD de cette gamme. Si le coefficient de corrélation sort de cet intervalle, il y a corrélation significative dans les changements de fréquence de décharge, pour cette taille de fenêtre w. On répète ces calculs pour des tailles de fenêtre différentes, allant de 0.1 à 10 secondes.

L'interprétation des résultats est moins évidente qu'il n'y paraît. La méthode décrite ci-dessus permet certes de déterminer si un coefficient de corrélation, pour une certaine valeur de w, est significativement supérieur ou inférieur à ce qu'il doit être pour des trains indépendants. Mais l'expérience montre qu'un seul changement de fréquence, même faible, plus ou moins concomitant dans les deux trains, peut à lui seul faire sortir le coefficient de corrélation de son intervalle de confiance, du moins pour les durées d'enregistrement analysées (5 minutes). Or, ce type d'événement peut arriver par hasard, dans le cas de PCs très éloignées, voire dans le cas de PCs enregistrées successivement et non simultanément. Il est donc difficile de décider en général, si un changement de fréquence approximativement simultané est dû au hasard ou non. Il semble plus raisonnable de penser que ces changements doivent se répéter plusieurs fois sur quelques minutes s'il existe un lien réel entre les évolutions des fréquences de décharge de la paire de cellules. Mais à ce stade, il n'existe pas de moyen quantitatif de décider de ce qui peut être dû au hasard de ce qui ne l'est pas.

4.4 Disponibilité des routines d'analyse : les logiciels *SpikeOMatic* et *STAR*

Toutes les routines nécessaires à l'analyse des données extracellulaires multiunitaires, depuis la détection des PAs jusqu'à la classification par les algorithmes EM et MCMC et l'exploitation des résultats qu'ils fournissent, sont disponibles gratuitement, sous licence GPL (GNU Public Licence), sur le site *web* du laboratoire[4]. Ce logiciel, *SpikeOMatic*, est un paquetage R[5], dont l'utilisation est décrite et guidée par deux tutoriels qui font l'objet du troisième article. Ces tutoriels sont évidemment disponibles à la même adresse que le logiciel lui-même. Une interface graphique facilite son utilisation. Comme pour tout paquetage R, toutes les fonctions du logiciel sont par ailleurs accompagnées d'un descriptif détaillé qui peut être sollicité dans R. L'algorithme MCMC de classification des PAs a été programmé sous la forme d'une librairie de fonctions C (*SpikeOMatic Library*) appelée par l'une des fonctions du paquetage R. Ces fonctions C ont recours à la GNU Scientific Library (GSL) pour la manipulation des vecteurs et des matrices, ainsi que pour la génération de nombres aléatoires. L'élaboration de ce logiciel a été supervisée par Christophe Pouzat.

Les routines d'analyse des trains de PAs ont été écrites en C, dans les plateformes d'analyse Scilab[6] et R. La version R de ces programmes est incluse dans un deuxième paquetage R qui regroupe tout un ensemble de fonctions d'analyse de trains de PAs : *SpikeTrainAnalysiswithR*, ou *STAR*. Ce paquetage comprend également des routines de

[4] `http ://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/SOM.html`

[5] R est une plateforme d'analyse disponible à l'adresse : `http ://www.r-project.org`

[6] `http ://scilabsoft.inria.fr`

simulation de trains de PAs interagissant. Il a été réalisé par Christophe Pouzat et moi-même, avec la participation d'Arthur Leblois. Il est également disponible gratuitement, sous licence GPL, sur le site *web* du laboratoire[7]. Toutes les méthodes des routines de ce paquetage ne sont pas exposées dans ce manuscrit. Leur utilisation est illustrée dans l'analyse du quatrième article.

La réalisation et la mise à disposition de ces deux logiciels constitue une part importante de ce travail.

4.5 Reproductibilité des analyses

Le deuxième article et l'appendice du quatrième sont également disponibles sur le site *web* du laboratoire[8] sous forme de *compendia* : chaque compendium est un paquetage R qui contient les données analysées dans l'article auquel il est associé, les codes spécifiques développés pour réaliser les calculs et les figures dudit article, ainsi qu'un "meta-fichier" (ou "vignette" dans le jargon R) qui contient le texte descriptif de l'analyse et les commandes utilisées pour la produire. Ces compendia permettent au lecteur de reproduire toute l'analyse effectuée dans l'article sur son propre ordinateur (voir le quatrième article pour la description, la fabrication et l'utilisation d'un compendium).

Implémentation

Toutes les figures de cette thèse et des articles ont été réalisées dans les deux environnements d'analyse Scilab et R. Ce manuscrit a été tapé avec LyX. Les articles l'ont été sous Emacs et LyX.

Tous les codes (C, Scilab et R) ont été développés sous Emacs. Les codes C ont été compilés avec le compilateur *intel* icc et exécutés sur un PC (Pentium IV 2 GHz) sous Linux.

[7] `http ://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/STAR.html`
[8] `http ://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Compendium.html`

Chapitre 5

Résultats

5.1 Premier article : l'approche MCMC du *spike-sorting*

Dans le chapitre 3 ont été présentées les différentes méthodes de *spike-sorting* utilisées jusqu'à présent. Ont également été présentées les difficultés que ces méthodes ne peuvent pas résoudre. En se fondant exclusivement sur la forme des potentiels d'action (PAs), elles ignorent l'information temporelle contenue dans les temps des PAs. Elles ne peuvent pas, de ce fait, prendre en compte automatiquement une quelconque dépendance de la forme des PAs vis-à-vis des intervalles de temps qui les séparent. Cette non-stationnarité des formes de PAs a pourtant été constatée expérimentalement dans de multiples préparations.

Cet article est l'article fondateur de la nouvelle approche que nous proposons dans le domaine. Il décrit une application de la méthodologie de *Monte Carlo par Chaînes de Markov* (MCMC) au problème du *spike-sorting*. Cette classe générale de méthodes est utilisée depuis plus de 50 ans par les physiciens. Et depuis 20 ans, elles connaissent un développement spectaculaire au sein de la communauté des statisticiens. Elles sont maintenant utilisées dans de multiples domaines scientifiques. Cet article s'articule autour de trois axes centraux :

Le modèle de génération de données. Notre méthode MCMC permet de réaliser une inférence statistique sur un modèle de génération de données plus réaliste que ceux considérés par les méthodes de *spike-sorting* existantes. Ce modèle est constitué de deux volets :

1. *la statistique de décharge des neurones enregistrés* : la densité empirique des intervalles de temps ente PAs (ISIs) d'un neurone est modélisée par une densité de probabilité log-normale.
2. *la dépendance de l'amplitude des PAs vis-à-vis des ISIs* : elle est modélisé par une relaxation exponentielle de l'amplitude avec les ISIs croissants.

L'algorithme MCMC (Metropolis-Hastings). Cet algorithme et ses fondements mathématiques sont présentés formellement. Son implémentation effective, dans

le cadre de notre modèle particulier est exposée dans le détail. Nous montrons également comment cette méthode permet d'obtenir une densité *a posteriori* des paramètres du modèle, ainsi qu'une distribution de probabilités discrètes sur les labels (neurones d'origine) de chaque PA.

Le test sur des données simulées. Afin d'illustrer les performances de la méthode, celle-ci est mise en oeuvre sur des données simulées. Parmi les décharges de neurones simulées, plusieurs l'ont été sur la base de modèles différents de celui utilisé par notre algorithme, afin de montrer la robustesse de la méthode. D'autre part, les recouvrements entre nuages de points tels qu'ils peuvent apparaître sur les graphiques de Wilson (voir chapitre 4, section 4.2) sont importants, ce qui rend inopportune toute méthode classique de délimitation de nuages (*clustering*). Cette analyse de données simulées fait apparaître le problème de relaxation lente de l'algorithme en raison de l'existence d'états métastables qui le piègent dans des minima locaux d'énergie. Nous présentons une solution performante à ce problème, introduite en Statistiques et en Physique : la méthode d'échange de répliques (*Replica Exchange Method*, REM).

Cet article pose ainsi les fondements d'une nouvelle approche du problème du *spike-sorting*. Ce nouveau cadre de travail permet de lever certaines difficultés fondamentales dans le domaine. La méthode est générale et flexible ; elle ne se cantonne évidemment pas au modèle de génération de données envisagé dans cet article. Il est possible de remplacer celui-ci par un modèle différent si les données l'exigent (voir le second article).

A ce stade de l'exposé de la méthode, il reste à montrer sa flexibilité, son efficacité et ses performances sur des données réelles. C'est l'objet du second article. Par ailleurs, deux axes importants sont en cours de développement. La méthode ne fournit actuellement pas d'estimation du *nombre de neurones* ; celui-ci doit être choisi par l'expérimentateur au vu des résultats obtenus avec différentes valeurs de ce nombre. Dans le cadre de notre approche, la comparaison de modèles peut se faire avec la méthode dite d'intégration thermodynamique (Ogata, 1989). Les premières mises en oeuvre préliminaires sont encourageantes. L'autre axe en développement est la *parallélisation* de l'algorithme. Le temps de calcul peut être divisé par le nombre de processeurs travaillant en parallèle.

J Neurophysiol 91: 2910–2928, 2004.
First published January 28, 2004; 10.1152/jn.00227.2003.

Improved Spike-Sorting By Modeling Firing Statistics and Burst-Dependent Spike Amplitude Attenuation: A Markov Chain Monte Carlo Approach

Christophe Pouzat,[1] Matthieu Delescluse,[1] Pascal Viot,[2] and Jean Diebolt[3]

[1]*Laboratoire de Physiologie Cérébrale, Centre National de la Recherche Scientifique (CNRS) Unité Mixte de Recherche (UMR) 8118, Université René Descartes, 75006, Paris;* [2]*Laboratoire de Physique Théorique des Liquides, CNRS UMR 7600, Université Pierre et Marie Curie, 75252 Paris Cedex 05; and* [3]*Laboratoire d'Analyse et de Mathématiques Appliquées, CNRS UMR 8050, Université de Marne la Vallée, Batiment Copernic, Cité Descartes, 77454 Marne la Vallée Cedex, France*

Submitted 10 March 2003; accepted in final form 14 January 2004

Pouzat, Christophe, Matthieu Delescluse, Pascal Viot, and Jean Diebolt. Improved spike-sorting by modeling firing statistics and burst-dependent spike amplitude attenuation: a Markov chain Monte Carlo approach. *J Neurophysiol* 91: 2910–2928, 2004. First published January 28, 2004; 10.1152/jn.00227.2003. Spike-sorting techniques attempt to classify a series of noisy electrical waveforms according to the identity of the neurons that generated them. Existing techniques perform this classification ignoring several properties of actual neurons that can ultimately improve classification performance. In this study, we propose a more realistic spike train generation model. It incorporates both a description of "nontrivial" (i.e., non-Poisson) neuronal discharge statistics and a description of spike waveform dynamics (e.g., the events amplitude decays for short interspike intervals). We show that this spike train generation model is analogous to a one-dimensional Potts spin-glass model. We can therefore tailor to our particular case the computational methods that have been developed in fields where Potts models are extensively used, including statistical physics and image restoration. These methods are based on the construction of a Markov chain in the space of model parameters and spike train configurations, where a configuration is defined by specifying a neuron of origin for each spike. This Markov chain is built such that its unique stationary density is the posterior density of model parameters and configurations given the observed data. A Monte Carlo simulation of the Markov chain is then used to estimate the posterior density. We illustrate the way to build the transition matrix of the Markov chain with a simple, but realistic, model for data generation. We use simulated data to illustrate the performance of the method and to show that this approach can easily cope with neurons firing doublets of spikes and/or generating spikes with highly dynamic waveforms. The method cannot automatically find the "correct" number of neurons in the data. User input is required for this important problem and we illustrate how this can be done. We finally discuss further developments of the method.

INTRODUCTION

The study of neuronal populations activity is one of the main experimental challenges of contemporary neuroscience. Among the techniques that have been developed and used to monitor populations of neurons, the multielectrodes extracellular recordings, albeit one of the oldest, still stands as one of the methods of choice. It is relatively simple and inexpensive to implement (especially when compared to imaging techniques) and it can potentially give access to the activity of many neurons with a fine time resolution (below the ms). The full exploitation of this method nevertheless requires some basic problems to be solved, among which stands prominently the spike-sorting problem. The raw data collected by extracellular recordings are of course corrupted by some recording noise but, more important, are a mixture of activities from different neurons. This means that even in good situations where action potentials or spikes can be unambiguously distinguished from background noise, the experimentalist is left with the intricate problem of finding out how many neurons are in fact contributing to the recorded data and, for each spike, which is the neuron of origin. This in essence is the spike-sorting problem. Extracellular recordings being an old method, the problem has been recognized as such for a long time and there is already a fairly long history of proposed solutions (Lewicki 1998). Nevertheless it seems to us that none of the available methods makes full use of the information present in the data. They consider mainly or exclusively the information provided by the waveform of the individual spikes and they neglect that provided by their occurrence time. Yet, the importance of the spikes occurrence times shows up in two ways.

1) First, the sequence of spike occurrence times emitted by a neuron has well-known features like the presence of a refractory period (e.g., after a spike has been emitted no other spike will be emitted by the same neuron for a period whose duration varies from 2 to 10 ms). Other features include an interspike interval (ISI) probability density, which is often unimodal and skewed (it rises "fast" and decays "slowly"). Although some methods make use of the presence of a refractory period (Fee et al. 1996a; Harris et al. 2000), none makes use of the full ISI density. This amounts to throwing useful information away, for if event 101 has been attributed to neuron 3 we not only know that no other spike can arise from neuron 3 for, say, the next 5 ms, but we also know that the next spike of this neuron is likely to occur with an ISI between 10 and 20 ms. As pointed out by Lewicki (1994), any spike-sorting method that would include an estimate of the ISI density of the different neurons should produce better classification performances.

2) Second, the spike waveform generated by a given neuron often depends on the time elapsed since its last spike (Quirk et al. 1999). This nonstationarity of spike waveform will worsen the classification reliability of methods that assume stationary waveform (Lewicki 1994; Pouzat et al. 2002). Other methods

Address for reprint requests and other correspondence: C. Pouzat, Laboratoire de Physiologie Cérébrale, CNRS UMR 8118, Université René Descartes, 45 rue des Saints Pères, 75006, Paris, France (E-mail: christophe.pouzat@univ-paris5.fr).

The costs of publication of this article were defrayed in part by the payment of page charges. The article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. Section 1734 solely to indicate this fact.

(e.g., Gray et al. 1995 for a "manual" method and Fee et al. 1996a for an automatic one) assume a continuity of the cluster of events originating from a given neuron when represented in a "feature space" [the peak amplitude, valley(s) amplitude(s), half-width are examples of such features] and make use of this property to classify nonstationary spikes. Although it is clear that the latter methods will in general outperform the former, we think that some cases could arise where none of them would work. In particular, doublet-firing cells could generate well-separated clusters in feature space and these clusters would not be put together by presently available methods.

These considerations motivated us to look for a new method that would have a built-in capability to take into account both the ISI density of the neurons and their spike waveform dynamics. However, there are two additional issues with spike-sorting that, we think, are worth considering.

1) *"Hard" versus "soft" spike-sorting.* Most spike-sorting methods, including the ones where users actually perform the clustering and some automatic methods like the one of Fee et al. (1996a), generate "hard" classification. That is, each recorded event is attributed "fully" to one of the K neurons contributing to the data. Methods based on probabilistic models (Lewicki 1994; Nguyen et al. 2003; Pouzat et al. 2002; Sahani 1999) deal with this issue differently: each event is given a probability to have been generated by each of the K neurons (e.g., one gets results like: event 100 was generated by neuron 1 with probability 0.75 and by neuron 2 with probability 0.25). We will refer to this kind of classification as "soft" classification. Users of the latter methods are often not aware of their soft classification aspect because what one does for subsequent processing like estimating the ISI density of a given neuron or computing the cross-correlogram of spike trains from 2 different neurons, is to *force* the soft classification into the most likely one (in the example above we would force the classification of event 100 into neuron 1). By doing so, however, the analyst introduces a bias in the estimates. It seems therefore interesting to find a way to keep the soft classification aspect of the model-based methods when computing estimates.

2) *Confidence intervals on model parameters.* Methods based on an explicit model for data generation could in principle generate confidence intervals for their model parameters, although this was never done. However, the values of model parameters do influence the classification and a source of bias could be reduced by including information about uncertainty of model parameters values.

Based on these considerations we have developed a semi-automatic method that takes spike waveform dynamics and ISI densities into account, which produces soft classification and allows the user to make full use of it, and which generates a full posterior density for the model parameters and confidence intervals. Our method uses a probabilistic model for data generation where the label (i.e., neuron of origin) of a given spike at a given time *depends* on the label of other spikes occurring just before or just after. We will call *configuration* the specification of a label for each spike in the train. It will become clear that with this model the spike-sorting problem is analogous to an image restoration problem where a picture (i.e., a set of pixel values) that has been generated by a real object and corrupted by some noise is given. The problem is to find out what the actual pixel value was, given the noise properties and the known correlation properties of noise free images. The case of spike sorting is equivalent to a one-dimensional "pixel sequence" where the pixel value is replaced by the label of the spike. More generally it is a special case of a Potts spin-glass model encountered in statistical physics (Newman and Barkema 1999; Wu 1982). Thanks to this analogy solutions developed to study Potts models (Landau and Binder 2000; Newman and Barkema 1999) and to solve the image restoration problem (Geman and Geman 1984) can be tailored to the spike-sorting problem. These solutions rely on a Markov chain that has the posterior density of model parameters and configurations given the observed data as its unique stationary density. This Markov Chain is stochastically simulated on the computer. This general class of methods has been used for 50 years by physicists (Metropolis et al. 1953) who call it *Dynamic Monte Carlo* and for 20 years by statisticians (Fishman 1996; Geman and Geman 1984; Liu 2001; Robert and Cassela 1999) who call it *Markov Chain Monte Carlo* (MCMC).

Our purpose in this paper is 2-fold: First to demonstrate that a particular MCMC method allows incorporation of more realistic data generation models and by doing so to perform reliable spike-sorting even in difficult cases (e.g., in the presence of neurons generating 2 well-separated clusters in feature space). Second, to explain how and why this method works, thereby allowing its users to judge the quality of the produced classification, to improve the algorithm implementing it, and to adapt it to their specific situations. Our method does not yet provide an automatic estimate of the number of neurons present in the data, although we illustrate how this critical problem can be addressed by the user.

METHODS

Data generation model

In the present manuscript we will make the following assumptions about data generation.

1) The firing statistics of each neuron are fully described by a time-independent interspike interval density. That is, the sequence of spike times from a given neuron is a realization of a homogeneous renewal point process (Johnson 1996).

2) The spike amplitudes generated by each neuron depend on the elapsed time since the previous spike of this neuron.

3) The measured spike amplitudes are corrupted by a Gaussian white noise, which sums linearly with the spikes and is statistically independent of them.

These assumptions have been chosen to show as simply as possible the implementation of our method and should not be taken as intrinsic limitations of this method. They constitute, moreover, a fairly good first approximation of real data in our experience. More sophisticated models where the next ISI of a neuron depends on the value of the former one (Johnson et al. 1986) could easily be included. In the same vein, models where the amplitude of a spike depends on several of the former ISIs could be considered. The Gaussian white noise assumption means that the analyst has "whitened" the noise before starting the spike-sorting procedure (Pouzat et al. 2002).

INTERSPIKE INTERVAL DENSITY. We will use in this paper a log-Normal density for the ISIs. This density looks like a good next guess, after the exponential density, when one tries to fit empirical ISI densities. It is unimodal, exhibits a refractory period, rises "fast," and decays "slowly." The version of the log-Normal density we are using depends on 2 parameters: a dimensionless shape parameter σ and a scale parameter s measured in seconds. The probability density for a

realization of a log-Normal random variable I with parameters σ and s to have the value i is

$$\pi_{isi}(I = i|\sigma, s) = \frac{1}{i\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{\log\left(\frac{i}{s}\right)}{\sigma}\right)^2\right] \quad (1)$$

In what follows we will use a shorter notation

$$i \sim \log - \text{Norm}\,(\sigma, s)$$

meaning that i is a realization of a log-Normal random variable with parameters σ and s. A discussion of the properties of the log-Normal distribution can be found in the *e-Handbook of Statistical Methods.*

SPIKE AMPLITUDE DYNAMICS. We will consider events described by their occurrence time and their peak amplitude measured on one or several recording sites. This makes the equations more compact, but full waveforms can be accommodated if necessary. Following Fee et al. (1996b) we will describe the dependence of the amplitude on the ISI by an exponential relaxation

$$\mathbf{A}(i) = \mathbf{P}[1 - \delta \exp(-\lambda i)] \quad (2)$$

where i is the ISI, λ is the inverse of the relaxation time constant (measured in 1/s), $\mathbf{P}$ is the vector of the maximal amplitude of the event on each recording site (i.e., this is the amplitude observed when $i \gg \lambda^{-1}$), and δ is the maximal modulation [i.e., for $i \ll \lambda^{-1}$ we have $\mathbf{A}(i) \approx \mathbf{P}(1 - \delta)$]. It is clear from this equation that the amplitudes of events generated by a given neuron are modulated by the same relative amount *on each recording site.* This is one of the key features of modulation observed experimentally (Gray et al. 1995).

To keep equations as compact as possible we will assume that the amplitudes of the recorded events on each recording site have been normalized by the SD of the noise on the corresponding site. **A** and **P** in *Eq. 2* are therefore dimensionless and the noise SD equals 1. Combining *Eq. 2* with our third model hypothesis (independent Gaussian noise corruption) we get for the density of the amplitude vector **a** of a given neuron conditioned on the ISI value i

$$\pi_{amp}(\mathbf{a}|i, \mathbf{P}, \delta, \lambda) = (2\pi)^{-(n_s/2)} \exp[-\tfrac{1}{2} \|\mathbf{a} - \mathbf{P}[1 - \delta \exp(-\lambda i)]\|^2] \quad (3)$$

where n_s is the number of recording sites and $\|\mathbf{v}\|$ stands for the Euclidean norm of $\mathbf{v}$. We will sometimes express *Eq. 3* in a more compact form

$$\mathbf{a} \sim \text{Norm}\,[\mathbf{P}(1 - \delta \exp(-\lambda i)], 1] \quad (4)$$

where Norm ($\mathbf{m}$, v) stands for a normal distribution with mean $\mathbf{m}$ and whose covariance matrix is diagonal with diagonal values equal to v.

We get the density of the couple (i, **a**) by combining *Eq. 1* with *Eq. 3*

$$\pi(i, \mathbf{a}|\sigma, s, \mathbf{P}, \delta, \lambda) = \pi_{amp}(\mathbf{a}|i, \mathbf{P}, \delta, \lambda)\pi_{isi}(i|\sigma, s) \quad (5)$$

NOTATIONS FOR THE COMPLETE MODEL AND THE DATA. We now have, for each neuron in the model, 2 parameters (σ and s) to specify the ISI density and *2 + number of recording sites* parameters (e.g., for tetrode recordings: P_1, P_2, P_3, P_4, δ, λ) to specify the amplitude density conditioned on the ISI value (*Eq. 3*). That is, in the case of tetrode recording: 8 parameters per neuron. In the sequel we will use the symbol θ to refer to the complete set of parameters specifying our model. That is for a model with K neurons applied to tetrode recordings

$$\theta = (\sigma_1, s_1, P_{1,1}, P_{1,2}, P_{1,3}, P_{1,4}, \delta_1, \lambda_1, \ldots, \sigma_K, s_K, P_{K,1}, P_{K,2}, P_{K,3}, P_{K,4}, \delta_K, \lambda_K) \quad (6)$$

We will assume that our data sample is of size N and that each element j of the sample is fully specified by its occurrence time t_j and its peak amplitude(s), $a_{j,1}$, $a_{j,2}$, $a_{j,3}$, $a_{j,4}$ (for tetrode recording). We will use Y to refer to the full data sample, that is

$$Y = (t_1, a_{1,1}, a_{1,2}, a_{1,3}, a_{1,4}, \ldots, t_N, a_{N,1}, a_{N,2}, a_{N,3}, a_{N,4}) \quad (7)$$

Data augmentation, configurations, and likelihood function

DATA AUGMENTATION AND CONFIGURATIONS. Until now we have specified our model and our data representation but what we are really interested in is to find the neuron of origin of each individual spike, or more precisely, for a model with K neurons, the probability for each spike to have been generated by each of the K neurons (remember that we are doing soft classification). To do that we will associate with each spike j a *latent* variable, $c_j \in \{1, 2, \ldots, K\}$. When c_j has value 2, that means that spike j has been "attributed" to neuron 2 of the model. We will use C to refer to the set of N variables c

$$C = (c_1, \ldots, c_N) \quad (8)$$

For a data sample of size N and a model with K neurons we have K^N different C values. We will call *configuration* a specific C value. That is, a configuration is a N-dimensional vector whose components take value in $\{1, 2, \ldots, K\}$ [e.g., $(1, \ldots, 1)$, $(K, \ldots, K)$, are 2 configurations]. In the statistical literature, this procedure of "making the data more informative" by adding a latent variable is called *data augmentation* (Liu 2001; Robert and Casella 1999).

LIKELIHOOD FUNCTION. Because C has been introduced it is pretty straightforward to compute the likelihood function of the "augmented" data, given specific values of θ: $L(Y, C|\theta)$. We remind the reader here that the likelihood function is *proportional* to the probability density of the (augmented) data given the parameters. Figure 1 illustrates how this is done on a simple case with 2 units in the model ($K = 2$) and a single recording site.

We first use the configuration specified by C to split the sample into K spike trains, one train from each neuron. (This is done in Fig. 1 when one goes from *A* to *B* and *C*.) Then because our model in its present form does not include interactions between neurons we just need to compute K distinct likelihood functions, one for each neuron specific spike train, and multiply these K terms to obtain the full likelihood

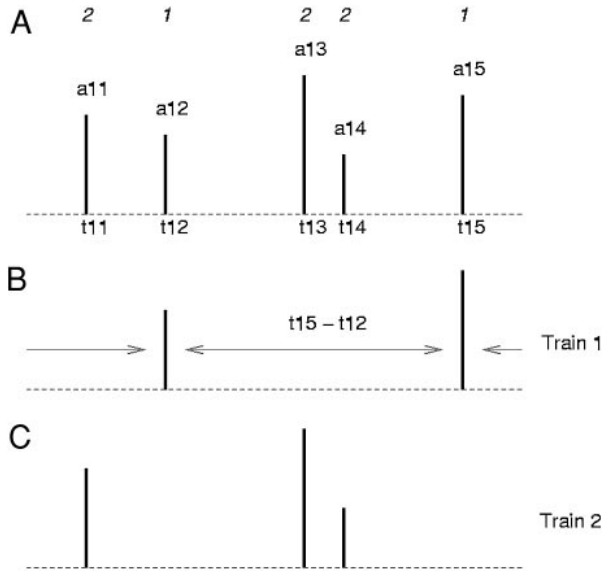

FIG. 1. Likelihood computation. *A*: snapshot of spikes 11 to 15 of a train. c_j values are shown at the *top*. Spike amplitudes are given (a_j) as well as the occurrence times (t_j). *B*: spikes from units 1 are shown alone. *C*: spikes from units 2 are shown alone.

$$L(Y, C|\theta) = \prod_{l=1}^{K} L(Y_l, C|\theta) \quad (9)$$

where Y_l is the subsample of Y for which the components c_j of C are equal to l. The $L(Y_l, C|\theta)$ themselves are product of terms like *Eq. 5*. In the case illustrated on Fig. 1 we will have for the first train

$$L(Y_1, C|\theta) \propto \pi(t_{12} - t_{1,former}, a_{12}|\sigma_1, s_1, P_1, \delta_1, \lambda_1)\pi(t_{15} - t_{12}, a_{15}|\sigma_1, s_1, P_1, \delta_1, \lambda_1) \times \pi(t_{1,next} - t_{15}, a_{1,next}|\sigma_1, s_1, P_1, \delta_1, \lambda_1) \quad (10)$$

where $t_{1,former}$ stands for the time of the former spike attributed to neuron 1 in configuration C and $t_{1,next}$ stands for the next spike attributed to neuron 1 in configuration C. If there are N_1 spikes attributed to neuron 1 then, using periodic boundary conditions (see *Parameter-specific transition kernels*), there will be N_1 terms in $L(Y_1, C|\theta)$. For the second neuron of Fig. 1 we obtain

$$L(Y_2, C|\theta) \propto \pi(t_{11} - t_{2,former}, a_{11}|\sigma_2, s_2, P_2, \delta_2, \lambda_2)\pi(t_{13} - t_{11}, a_{13}|\sigma_2, s_2, P_2, \delta_2, \lambda_2) \times \pi(t_{14} - t_{13}, a_{14}|\sigma_2, s_2, P_2, \delta_2, \lambda_2)\pi(t_{2,next} - t_{14}, a_{2,next}|\sigma_2, s_2, P_2, \delta_2, \lambda_2) \quad (11)$$

The reader can see that for any configuration *Eq. 9* involves the computation of N terms.

Posterior and prior densities, a combinatorial explosion

POSTERIOR DENSITY. Now that we know how to compute the likelihood function we can write the expression of the posterior density of model parameters and configuration given the data using Bayes formula

$$\pi_{post}(\theta, C|Y) = \frac{L(Y, C|\theta)\pi_{prior}(\theta)}{Z} \quad (12)$$

where $\pi_{prior}(\theta)$ is the prior density of the parameters and

$$Z = \sum_C \int_\theta L(Y, C|\theta)\pi_{prior}(\theta)\mathrm{d}\theta \quad (13)$$

where Z is the probability of the data. It is clear that if we manage to compute or estimate $\pi_{post}(\theta, C|Y)$ we will have done our job, for then the soft classification is given by the marginal density of C

$$\pi_{post}(C|Y) = \int_\theta \pi_{post}(\theta, C|Y)\mathrm{d}\theta \quad (14)$$

The reader will note that this posterior marginal density includes *all the information available about* θ, that is, the uncertainty on the parameters is automatically taken into account.

PRIOR DENSITY. We will assume here that several thousand spikes have been collected and that we perform the analysis by taking chunks of data, say the first 5,000 spikes, then spikes 5,001 to 1,000 and so on. For the first chunk we will assume we know "little" a priori and that the joint prior density $\pi_{prior}(\theta)$ can be written as a product of the densities for each component of θ, that is

$$\pi_{prior}(\theta) = \prod_{q=1}^{K} \pi(\sigma_q)\pi(s_q)\pi(\delta_q)\pi(\lambda_q)\pi(P_{q,1})\pi(P_{q,2})\pi(P_{q,3})\pi(P_{q,4})$$

where we are assuming that 4 recording sites have been used. We will further assume that our signal to noise ratio is not better 20 (a rather optimistic value), that our spikes are positive, and therefore the $\pi(P_{q,1\ldots4})$ are null below 0 and above +20 (remember we are working with normalized amplitudes). We will reflect our absence of prior knowledge about the amplitudes by taking a uniform distribution between 0 and +20. The λ value reported by Fee et al. (1996b) is 45.5 s^{-1}. λ must, moreover, be smaller than ∞, so we took a prior density uniform between 10 and 200 s^{-1}. δ must be ≤ 1 (the amplitude of a spike from a given neuron on a given recording site *does not* change sign) and ≥ 0 (spikes do not become larger upon short ISI), so we used a uniform density between 0.1 and 0.9 for δ. An inspection of the effect of the shape parameter σ on the ISI density is enough to convince an experienced neurophysiologist that empirical unimodal ISI densities from well-isolated neurons will have $\sigma \in [0.1, 2]$. We therefore took a prior density uniform between 0.1 and 2 for σ. The same considerations led us to take a uniform prior density between 0.005 and 0.5 for s.

When one analyzes the following data chunks, say spikes 5,001 to 1,000, the best strategy is to take as a prior density for θ the posterior density from the previous chunk

$$\pi_{post}(\theta|Y) = \sum_C \pi_{post}(\theta, C|Y)$$

One has therefore a direct way to track parameters drifts during an experiment.

A COMBINATORIAL EXPLOSION. We have so far avoided considering the normalizing constant Z of *Eq. 12*, whose expression is given by *Eq. 13*. A close examination of this equation shows that it involves a multidimensional integration in a rather highly dimensional space (e.g., for a model with 10 neurons and data from tetrode recordings, we have 80 components in θ) and a summation over every possible configuration, that is a sum of K^N terms. That is really a lot, to say the least, in any realistic case (say, $N = 1{,}000$ and $K = 10$)! One can wonder how we will manage to compute Z and therefore implement our approach. Fortunately, we do not really need to compute it, as will now be explained.

The Markov Chain Monte Carlo method

AN ANALOGY WITH THE POTTS MODEL IN STATISTICAL PHYSICS. Let us define

$$e(\theta, C|Y) = L(Y, C|\theta)\pi_{prior}(\theta) \quad (15)$$

that is, $e(\theta, C|Y)$ is an *unnormalized* version of $\pi_{post}(\theta, C|Y)$, the posterior density, and

$$E(\theta, C) = -\log[e(\theta, C|Y)] \quad (16)$$

Then *Eq. 12* can be rewritten as

$$\pi_{post}(\theta, C|Y) = \frac{\exp[-\beta E(\theta, C)]}{Z} \quad (17)$$

with $\beta = (1/kT) = 1$. If one interprets E as an energy, β as an "inverse temperature," Z as a *partition function*, one sees that the posterior density, $\pi_{post}(\theta, C|Y)$, is the *canonical density* (or canonical distribution) used in statistical physics (Landau and Binder 2000; Newman and Barkema 1999). A closer examination of *Eqs. 10* and *11* shows that, from a statistical physics viewpoint, the (log of the) likelihood function describes "interactions" between spikes attributed to the same neuron. If one goes a step further and replaces "spikes" by "atoms on crystal nodes" and "neuron of origin" by "spin orientation," ones sees that our problem is analogous to what physicists call a one-dimensional Potts model. The typical Potts model is a system of interacting spins where each spin interacts only with its nearest neighbors and only if they have the same spin value (Wu 1982). In our case, spins (that is, spikes) interact with the former and next spins with the same value regardless of the distance (time interval) at which they are located. Moreover the "interaction" strength is a random variable that makes our system analogous to a spin glass (Binder and Young 1986; Landau and Binder 2000; Newman and Barkema 1999) and the amplitude contribution (*Eq. 3*) gives rise to an energy term analogous to an interaction between a spin and a random magnetic field (Newman and Barkema 1999). To be complete we should say that physi-

cists consider that θ is given, then *Eq. 17* gives the probability to find the "crystal" in a specific configuration C.

The point of this analogy will now become clear. Physicists are interested in estimating *expected values* (which are what they can experimentally measure) of functions that can be easily computed for each configuration. For instance, they want to compute the expected energy of the "crystal"

$$\langle E(\theta)\rangle = \sum_C E(\theta, C)\pi_{post}(\theta, C|Y) \tag{18}$$

To do that they obviously have to deal with the combinatorial explosion we alluded to. The idea of their solution is to draw configurations: $C^{(1)}, C^{(2)}, \ldots, C^{(m)}$, from the "target" distribution: $\pi_{post}(\theta, C|Y)$. Then an estimate of $\langle E(\theta)\rangle$ is the empirical average

$$\bar{E}(\theta) = \frac{1}{m}\sum_{t=1}^{m} E[\theta, C^{(t)}] \tag{19}$$

In other words, they use a Monte Carlo (MC) method (Landau and Binder 2000; Newman and Barkema 1999). The problem becomes therefore to generate the draws $[C^{(1)}, C^{(2)}, \ldots, C^{(m)}]$.

A SOLUTION BASED ON A MARKOV CHAIN. For reasons discussed in detail by Newman and Barkema (1999) and Liu (2001) the draws cannot be generated in practice by "direct" methods like rejection sampling (Fishman 1996; Liu 2001; Robert and Casella 1999). By "direct" we mean here that the $C^{(t)}$ are independent and identically distributed draws from $\pi_{post}(\theta, C|Y)$ for *a given* θ value. Instead we must have recourse to an artificial dynamics generating a sequence of *correlated* draws. This sequence of draws is in practice the realization of a Markov chain. That means that in the general case, where we are interested in getting both θ and C, we will use a *transition kernel:* $T[\theta^{(t)}, C^{(t)}|\theta^{(t-1)}, C^{(t-1)}]$, and simulate the chain starting from $[\theta^{(0)}, C^{(0)}]$, chosen such that $\pi_{post}[\theta^{(0)}, C^{(0)}|Y] > 0$. Moreover, we will build T such that the chain converges to a unique stationary distribution given by $\pi_{post}(\theta, C|Y)$. The estimator *Eq. 19* of the expected value *Eq. 18 is then still correct* even though the successive $[\theta^{(t)}, C^{(t)}]$ are correlated; we just need to be cautious when we compute the variance of the estimator (Janke 2002; Sokal 1989; *Empirical averages and SDs*). By still correct we formally mean that the following equality holds

$$\lim_{m\to\infty}\frac{1}{m}\sum_{t=1}^{m} E[\theta, C^{(t)}] = \langle E(\theta)\rangle \tag{20}$$

We give in *The Metropolis–Hastings algorithm* of the APPENDIX a general presentation and justification of the procedure used to build the transition kernel T: the *Metropolis–Hastings (MH) algorithm.* We then continue in *Parameter-specific transition kernels* with an account of the specific kernel we used for the analysis presented in this paper.

Empirical averages and SDs

As explained at the beginning of *Slow relaxation and REM,* once our Markov chain has reached equilibrium, or at least once its behavior is compatible with the equilibrium regime, we can estimate values of parameters of interest as well as errors on these estimates. The estimator of the probability for a given spike, say the 100th, to originate from a given neuron, say the second, is straightforward to obtain. If we assume that we discard the first $N_D = 15{,}000$ on a total of $N_T = 20{,}000$ iterations we have

$$\Pr(c_{100} = 2|Y) \approx \frac{1}{5{,}000}\sum_{t=15{,}000}^{20{,}000} I_2[c_{100}^{(t)}] \tag{21}$$

where I_q is the *indicator* function defined by

$$I_q[c_j^{(t)}] = \begin{cases} 1, & \text{if } c_j^{(t)} = \text{q} \\ 0, & \text{if } c_j^{(t)} \neq \text{q} \end{cases} \tag{22}$$

In a similar way, the estimate of the expected value of the maximal peak amplitude of the first neuron on the first recording site is

$$\bar{P}_{1,1} = \frac{1}{5{,}000}\sum_{t=15{,}000}^{20{,}000} P_{1,1}^{(t)}$$

To obtain the error on such estimators we have to keep in mind that the successive states $[\theta^{(t)}, C^{(t)}]$ generated by the algorithm are correlated. Thus, we cannot use the empirical variance divided by the number of generated states to estimate the error (Janke 2002; Sokal 1989). As explained in detail by Janke (2002) we have to compute for each parameter θ_i of the model the normalized autocorrelation function (ACF), $\rho_{norm}(l;\ \theta_i)$, defined by

$$\rho(l;\ \theta_i) = \frac{1}{N_T - N_D - l}\sum_{t=N_D}^{N_T-1}[\theta_i^{(t)} - \bar{\theta}_i][\theta_i^{(t+l)} - \bar{\theta}_i]$$

$$\rho_{norm}(l;\ \theta_i) = \frac{\rho(l;\ \theta_i)}{\rho(0;\ \theta_i)} \tag{23}$$

Then we compute the *integrated autocorrelation time,* $\tau_{autoco}(\theta_i)$

$$\tau_{autoco}(\theta_i) = \tfrac{1}{2} + \sum_{l=1}^{L}\rho(l;\ \theta_i) \tag{24}$$

where L is the lag at which ρ starts oscillating around 0. Using an empirical variance, $\sigma^2(\theta_i)$ of parameter θ_i, defined in the usual way

$$\sigma^2(\theta_i) = \frac{1}{N_T - N_D - 1}\sum_{t=N_D}^{N_T}[\theta_i^{(t)} - \bar{\theta}_i]^2 \tag{25}$$

Our estimate of the variance, $\mathrm{Var}\,[\bar{\theta}_i]$ of $\bar{\theta}_i$ becomes

$$\mathrm{Var}\,[\bar{\theta}_i] = \frac{2\tau_{autoco}(\theta_i)}{N_T - N_D - 1}\sigma^2(\theta_i) \tag{26}$$

The first consequence of the autocorrelation of the states of the chain is therefore to reduce the effective sample size by a factor of $2\tau_{autoco}(\theta_i)$. This gives us a first quantitative element on which different algorithms can be compared (as explained in *The Metropolis–Hastings algorithm* of the APPENDIX, the MH algorithm does in fact give us a lot of freedom on the choice of proposal transition kernels). It is clear that the faster the autocorrelation functions of the parameters fall to zero, the greater the statistical efficiency of the algorithm. The other quantitative element we want to consider is the computational time, τ_{cpu}, required to perform one MC step of the algorithm. One could for instance imagine that a new sophisticated proposal transition kernel allows us to reduce the largest τ_{autoco} of our standard algorithm by a factor of 10, but at the expense of an increase of τ_{cpu} by a factor of 100. Globally the new algorithm would be 10 times less efficient than the original one. What we want to keep as small as possible is therefore the product $\tau_{autoco}\tau_{cpu}$. With this efficiency criterion in mind, the replica exchange method described in *Slow relaxation and REM* becomes even more attractive.

Initial guess

MCMC methods are iterative methods that must be started from an initial guess. We chose randomly with a uniform probability $1/N$ as many actual events as neurons in the model (K). That gave us our initial guesses for the $P_{q,i}$. δ was set to δ_{min} for each neuron. All the other parameters were randomly drawn from their prior distribution. The initial configuration was generated by labeling each individual

spike with one of the K possible labels with a probability $1/K$ for each label (this is the $\beta = 0$ initial condition used in statistical physics).

Data simulation

We have tried to make the simulated tetrode data set used in this paper a challenging one. First the noise corruption was generated from a Student's t density with 4 degrees of freedom instead of a Gaussian density as assumed by our model (*Eq. 3*). Second, among the 6 simulated neurons only 3 had an actual log-Normal ISI density. Among the 3 others, one had a gamma density

$$\pi_{gamma}(i|s, \sigma) = \frac{s^{-\sigma}}{\Gamma(\sigma)} i^{\sigma-1} \exp\left(-\frac{i}{s}\right) \quad (27)$$

where s is a scale parameter and σ is a shape parameter. One neuron was a doublet-generating neuron and the third one had a truncated log-Normal ISI density. The truncation of the latter came from the fact that 18% of the events it generated were below the "detection threshold," which would have been at 3 noise SD. The doublet-generating neuron was obtained with 2 log-Normal densities, one giving rise to short ISIs (10 ms) the other one to "long" ones (60 ms); the neuron was moreover switching from the short, respectively the long, ISI generating state to the long, respectively the short, ISI generating state after each spike, giving rise to a strongly bimodal ISI density (Fig. 9*C*). The mean ISI of these 6 neurons ranged from 10 to 210 ms (Table 1). A "noise" neuron was moreover added to mimic the collective effect of many neurons located far away from the recording sites, which would therefore not be detected most of the time. This "noise" neuron was obtained by generating first a sequence of time points exponentially distributed. A random amplitude given by the background noise density (t density with 4 degrees of freedom; see above) was associated to each point and the point was kept in the "data" sample only if its amplitude exceeded the detection threshold (3 noise SD) on at least one of the 4 recording sites. The initial exponential density for this "noise" neuron was set such that a preset fraction of events in the final data sample would arise from it (in that case 15%, or 758 events on a total of 5,058 events). Details on the pseudorandom number generators required to simulate the data can be found in *Random number generators for the simulated data* of the APPENDIX.

Implementation details

Codes were written in C and are freely available under the Gnu Public License at our web site: http://www.biomedicale.univ-paris5.fr/physcerv/Spike-0-Matic.html. The free software Scilab (http://www-rocq.inria.fr/scilab/) was used to generate output plots as well as the graphical user interface, which comes with our release of the routines. The GNU Scientific Library (GSL: http://sources.redhat.com/gs1/) was used for vector and matrix manipulation routines and (pseudo)random number generators (Uniform, Normal, log-Normal, Gamma). The GSL implementation of the MT19937 generator of Matsumoto and Nishimura (1998) was used. This generator has a period of $2^{19,937} - 1$. Because the code requires a lot of exponential and logarithm evaluations, the exponential and logarithm functions were tabulated and stored in memory, which reduced the computation time by 30%. Codes were compiled and run on a PC laptop computer (Pentium IV with CPU speed of 1.6 GHz, 256 MB RAM) running Linux. The *gcc* compiler (http://www.gnu.org/software/gcc/gcc.htm) version 3.2 was used.

RESULTS

Data properties

The performances of our algorithm will be illustrated in this paper with a simulated data set. The data are supposed to come from tetrode recordings. Six neurons are present in the data together with a "noise" neuron supposed to mimic events rarely detected from neurons far away from the tetrode's recording sites. The data set has been made challenging for the algorithm by producing systematic deviations with respect to our model assumptions (*Data simulation*). This data set is supposed to come from 15 s of recording during which 5,058 events were detected. Wilson plots as they would appear to the user are shown on Fig. 2*A*. Figure 2*B* shows the same data with colors corresponding to the neuron of origin. The parameters used to simulate each individual neuron are given in Table 1. The reader can easily see that, whereas one cluster has very few points (purple cluster on Fig. 2*B*), others have a lot (brown, green, and blue clusters). Indeed, the smallest cluster contains 73 events, whereas the biggest contains 1,474 events. Some clusters are very elongated (e.g., green and red clusters) and one neuron, the doublet-generating neuron, even gives rise to 2 *well-separated clusters* (red). The "noise" neuron cluster (black) is much more spread out than the others. Two cluster pairs always exhibit an overlap (the yellow-black and the green-red pairs).

The deviation between the recording noise properties assumed by our model and the simulated noise is illustrated in Fig. 3. Figure 3*A1* shows the peak amplitude versus the *isi* for the second (green) neuron on the second recording site together with the theoretical relation between these 2 parameters. The reader can therefore get an idea of what *Eq. 2* means. Figure

TABLE 1. *Parameters used to simulate the neurons*

	Neuron						
Parameter	1	2	3	4	5	6	7
P_1, P_2, P_3, P_4	15, 10, 5, 0	10, 15, 5, 0	8, 12, 8, 3	9, 9, 9, 9	3, 8, 14, 19	3, 4, 5, 4	0, 0, 0, 0
δ	0.7	0.8	0.6	0.5	0.8	0.8	
λ	100	100	50	67	20	50	
s	20	5	10 and 50	10	200	20	
σ	0.3	3	0.2 and 0.2	0.2	0.3	0.5	
$\langle isi \rangle$	21	15	31	10	210	28	20
n	732	981	486	1,474	73	554	758

The maximal peak amplitude values (P_i) are given in units of noise SD. The scale parameters (s) and mean isi($\langle isi \rangle$) are given in milliseconds. The neuron with a gamma ISI density is in column 2. For the doublet-generating neuron (column 3) the parameter values of the 2 underlying log-normal densities are given. The neuron with a truncated log-normal density is located in column 6. The bottom row indicates the number of events from each neuron. The correspondence between neuron number and color on Figs. 2*B* and 5*A* is: 1, blue; 2, green; 3, red; 4, brown; 5, purple; 6, yellow; 7, black.

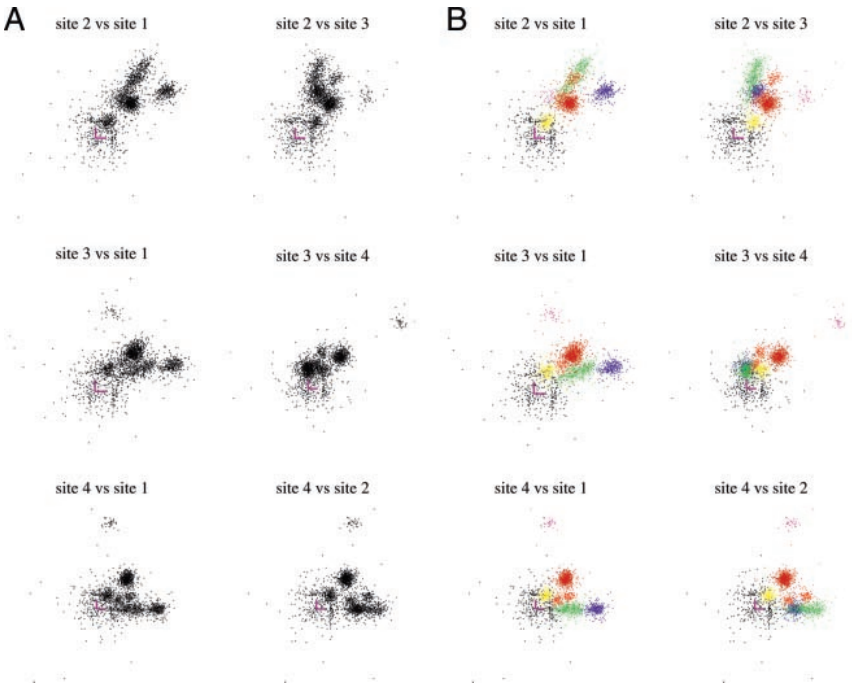

FIG. 2. Wilson plots showing about 25% of the sample (1,215 events). On each plot the scale bars meet at amplitude (0, 0) and are of size 2 (in units of noise SD). *A*: data as the analyst would see them before starting the spike-sorting procedure. *B*: same data sorted with the known neuron of origin encoded by the color: neuron 1 (blue), neuron 2 (green), neuron 3 (red), neuron 4 (brown), neuron 5 (purple), neuron 6 (yellow), noise events (black).

3*A2* shows the corresponding residuals (actual value − theoretical one). By looking at the distribution of these points the reader can see that the cluster shape of this neuron, in the 4-dimensional space whose projections are shown on the Wilson plots of Fig. 2, will be significantly skewed. Algorithms assuming a multivariate Gaussian shape of the clusters would therefore not perform well on these data. Figure 3*B1* shows the histogram of these residuals together with the Student's *t* density that was used to generate them (*Data simulation & Random number generators for the simulated data*) and the Gaussian density assumed by our model. The simulated recording noise generates at the same time more events very close to the ideal event and more events very far from it than a Gaussian noise would do.

Early exploration and model choice

We want to illustrate here some basic features of our algorithm dynamics as well as how model comparison can be done. In its present form, our approach requires the user to decide

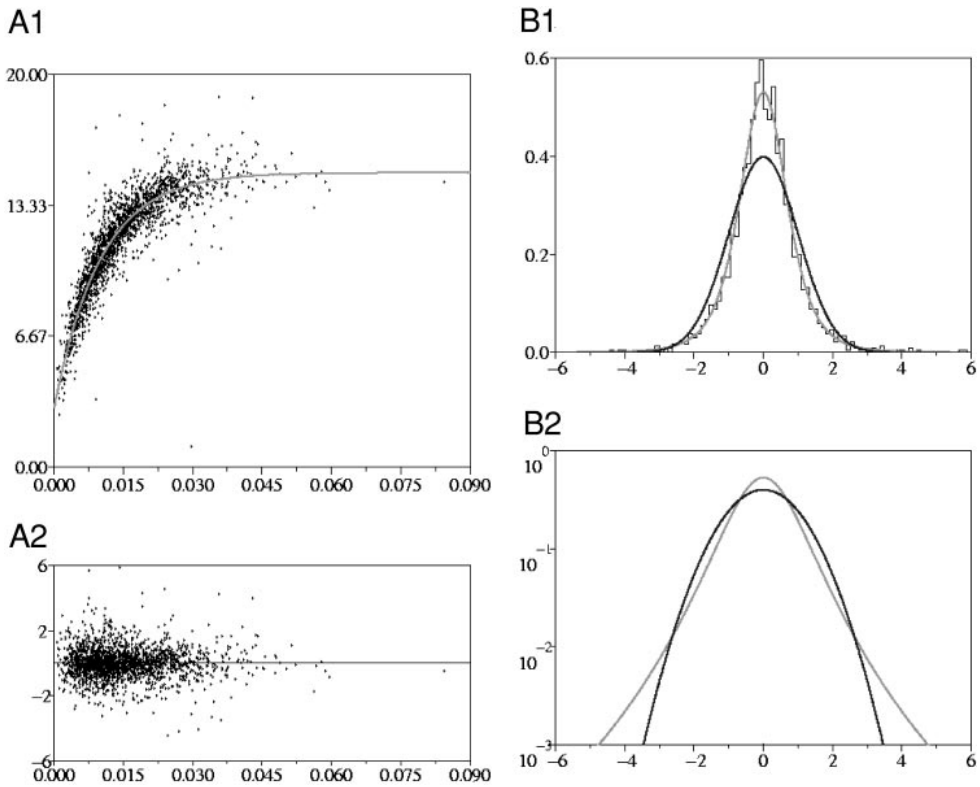

FIG. 3. Noise properties. *A1*: amplitudes of the events generated by neuron 2 (green on Fig. 2*B*) on the second recording site are plotted against the corresponding interspike interval (ISI). Ideal amplitude relaxation curve (*Eq. 2*, with $P_2 = 15$, $\delta = 0.8$, $\lambda = 100$) is shown in gray. Abscissa scale in *s*. *A2*, residuals: actual amplitude − ideal amplitude. *B1*: histogram of the residual values (broken line), ideal density from which these residuals have been generated (gray, *t* density with an SD = 1 and 4 degrees of freedom) and the density assumed by our model (black, Gaussian density with an SD = 1). *B2*: *t* (gray) and Gaussian (black) densities on a log scale showing the heavier tails of the *t* density.

what is the proper model (i.e., the proper number of neurons). This decision is based on the examination of "empirical" features associated with the different models. To obtain these empirical features, we have to perform the same computation on the different models, although the computation time increases with the number of events considered. It is therefore a good idea to start with a reduced sample that contains enough events for the model comparison to be done and little enough for the computation to run quickly. In this section we will work with the first 3 s of data (1,017 events), which represent approximately 20% of the total sample (5,058 events). We will focus here on a model with 7 neurons, but a similar study was performed with models containing from 4 to 10 neurons.

EVIDENCE FOR META-STABLE STATES. To explore a model with a given number of neurons we start by simulating 10 different and independent realizations of a Markov chain. The way random initial guesses for these different realizations are obtained is explained in *Initial guess.* For each realization, one Monte Carlo (MC) step consists in an update of the label of each of the 1,017 spikes (SPIKE LABEL TRANSITION MATRIX), an update of the amplitude parameters ($\mathbf{P}$, δ, λ) (AMPLITUDE PARAMETERS TRANSITION KERNELS) and of the 2 parameters of the ISI density, s and σ (SCALE PARAMETER TRANSITION KERNEL & SHAPE PARAMETER TRANSITION KERNEL) for each neuron. As explained in AMPLITUDE PARAMETERS TRANSITION KERNELS, the amplitude parameters are generated with a piecewise linear approximation of the corresponding posterior conditional. This piecewise linear approximation requires a number of discrete sampling points to be specified. Because when we start our model exploration, we know little about the location of the corresponding posterior densities, we used during these first 2,000 MC steps, 100 sampling points regularly spaced on the corresponding parameter domain defined by the prior (PRIOR DENSITY). Figure 4*A* illustrates the energy evolution of the 10 different trials (realizations). The reader not familiar with our notion of energy (*Eq. 16*) can think of it as an indication of the quality of fit. The lower it is, the better the fit. Indeed, if we were working with a template-matching paradigm, the sum of the squared errors would be proportional to our energy. The striking feature on Fig. 4*A* is the presence of "meta-stable" states (a state is defined by a configuration *and* a value for each model parameter). The trials can spend many steps at an almost constant energy level before making discrete downward steps (like the one falling from 5,800 to 4,300 before the 500th step). If we look at the parameters values and configurations generated by the different trials (not shown), the ones that end above 5,000 "miss" the purple cluster on Fig. 2*B*. The time required to perform 2,000 MC steps with 7 neurons in the model was roughly 9 min, meaning that 1.5 h was required to obtain the data of Fig. 4*A*. This time (for 2,000 MC steps) grew from 8′ for a model with 4 neurons to 10′ for a model with 10 neurons.

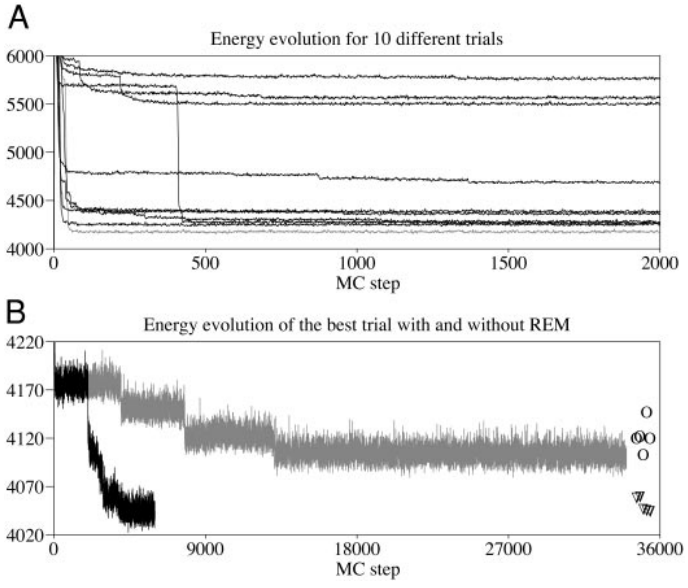

FIG. 4. *A*: energy evolution during 2,000 steps with 10 different initial random guesses. Best trial is shown in gray. *B*: follow-up of the energy evolution of the best trial without (gray) and with (black) the replica exchange method (REM). Circles: mean energy computed from the last 5,000 Monte Carlo (MC) steps of 5 runs without REM. Triangles: mean energy computed from the last 1,000 MC steps of 5 runs with REM.

THE REPLICA EXCHANGE METHOD SPEEDS UP CONVERGENCE IN THE PRESENCE OF META-STABLE STATES. The presence of meta-stable states is indeed a severe problem of our Markov chain dynamics, as further illustrated on the *gray trace* of Fig. 4*B*. Here the Markov chain was restarted from the state it had at the end of the 2,000 steps of the best among the 10 initial trials (*gray trace* on Fig. 4*A*). The last 1,000 steps of this trial were moreover used to locate 13 posterior conditional sampling points for each amplitude parameter of each neuron (AMPLITUDE PARAMETERS TRANSITION KERNELS). Then 5 different runs with 32,000 steps were performed. The best one is shown on the *gray trace* of Fig. 4*B* and the final mean energy of each of the 5 is shown on the *right end* of the graph (circles). What the reader sees here is a (very) slow convergence of a Markov chain to its stationary density. It is problematic because, if we stop the chain too early, our results could reflect more the properties of our initial guess than the properties of the target density we want to sample from. To circumvent this slow relaxation problem we have implemented a procedure commonly used in spin-glass (Hukushima and Nemoto 1996) and biopolymer structure (Hansmann 1997; Mitsutake et al. 2001) simulations: the replica exchange method (REM). The details of the methods are given in *Slow relaxation and REM* of the APPENDIX. The REM consists in simulations of "noninteracting" replicas of the same system (model) at different "inverse temperatures" (β) combined with exchanges between replicas states. These inverse temperatures are purely artificial and are defined by analogy with systems studied in statistical physics. The idea is that the system with a small β value will experience little difficulty in crossing the energy barriers separating local minima (meta-stable states) and will therefore not suffer from the slow relaxation problem. The replica exchange dynamics will allow "cold" replicas (replicas with a larger β value) to take profit of the "fast" state space exploration performed by the "hot" replica (small β). The efficiency of the method is demonstrated by the *black trace* on Fig. 4*B*. Here we again took the last state of the best trial of Fig. 4*A* to restart the Markov chain. We used 8 different β values: 1, 0.8, 0.6, 0.5, 0.45, 0.4, 0.35, and 0.3. Five different runs with 4,000 steps were performed and the energy trace at $\beta = 1$ is shown in its integrity; this was the best of the 5. The final mean energy of the 5 trials is shown on the *right side* of the figure (triangles). The Markov chains simulated with the REM relax clearly faster than the ones simulated with the single replica method.

The number of MC steps has been chosen so that the same computational time (32′) was required for both runs, the long one (32,000 steps) with a single replica at $\beta = 1$ (*gray trace*) and the short one (4,000 steps) with 8 replicas. The reader can remark that for these trials the time required to perform 2,000 MC steps for a single replica is 2′, whereas it was previously much larger (9′). This is attributed to the reduction in the number of sampling points used for the posterior conditional, 13 instead of 100 (AMPLITUDE PARAMETERS TRANSITION KERNELS). A significant amount of the computational time is therefore spent generating new amplitude parameters values.

MODEL CHOICE. Figure 5 illustrates what we meant by "empirical features associated with different models" at the beginning of this section. We have now performed the same series of runs with 7 different models having between 4 and 10 neurons. One feature we can look at in the output of these runs is the most likely classification produced (*Empirical averages and SDs, Eq. 21*). By "most likely" we mean that the label of each spike has been forced to its most likely value. Figure 5*A* shows one of the Wilson plots with different neuron numbers, where the events have been colored according to their most likely label. The good behavior of the algorithm appears clearly. When asked to account for the data with 6 neurons, it mainly lumps together the "small" neuron (yellow on Fig. 2*B*, neuron 6 in Table 1) with the "noise" neuron (black on Fig. 2*B*, neuron 7 in Table 1). With 7 neurons, the actual number of neurons, it splits the previously lumped clusters into 2 (although it attributes too many spikes to the small neuron and too few to the noise one; see *Long runs with full data set*). With 8 neurons, it further splits the noise cluster. In fact it creates a "noise" cluster (black on the figure) whose peak amplitude on the fourth recording site is roughly 3 times the recording noise SD and another one (clear blue) whose peak amplitude on this same site is smaller than 3. We remind the reader that our detection threshold is here set at 3 noise SD. The algorithm does not split the doublet-firing neuron (red) into 2 neurons, although this neuron gives rise to two well-separated clusters. With 9 neurons, it splits even further the previous clear blue noise "neuron" into 2 neurons, one with a peak amplitude roughly equal to the detection threshold on the first recording site (pink cluster) and one whose peak amplitudes on sites 1 and 4 are smaller than 3. With 10 neurons (not shown) we end up with the actual noise neuron, resulting in 4 neurons with an average peak amplitude slightly above the detection threshold on one of the recording sites and very close to zero on the 3 others.

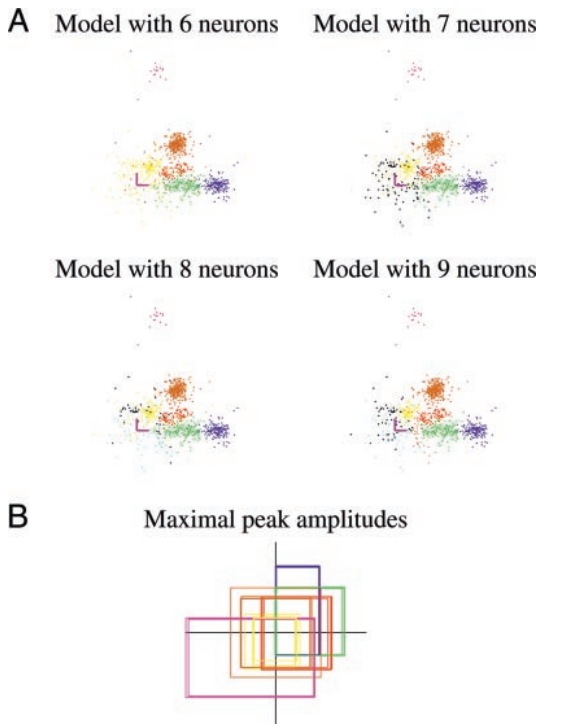

FIG. 5. Model choice. *A*: most likely configurations generated by 4 of the 7 models studied are displayed on a Wilson plot where the amplitude on site 4 is plotted against the amplitude on site 1. Scale bars as in Fig. 2. *B*: maximal peak amplitude plots showing that neuron identification from model to model is easy. Each semiaxis corresponds to the maximal peak amplitude (estimated from the algorithm output) for each of the 6 "important neurons." Vertical axis in the upper half-plan corresponds to the maximal peak amplitude on site 1. Horizontal axis in the right half-plan corresponds to the maximal peak amplitude on site 2. Vertical axis in the lower half-plan corresponds to the amplitude on site 3 and the horizontal axis in the left half-plan corresponds to the amplitude on site 4. For each semiaxis the amplitude value goes from 0 at the origin to 20 at the tip. Each neuron in each model corresponds to a frame. To make the figure more readable (by avoiding a too strong overlap of the frames) a random Gaussian noise ($\mu = 0$, $\sigma = 0.1$) has been added to the maximal peak amplitude values of each neuron.

Another feature we can look at is the average values of the model parameters. Figure 5*B* shows the maximal peak amplitudes for 6 of the neurons and for the 4 different models considered in Fig. 5*A*. These maximal peak amplitudes are constrained by our priors to take values between 0 and 20. Each of the 4 semiaxis represents therefore the maximal peak amplitude on each of the 4 recording sites (see legend) and each frame represents one neuron in one model. For instance the blue frames correspond to neuron 1 in Table 1. Its maximal peak amplitudes on sites 1, 2, 3, and 4 are 15, 10, 5, and 0, respectively. The frames corresponding to this neuron in the different models considered overlap perfectly. The same holds for each neuron, except the yellow and the brown ones. The frame change for the "yellow" neuron is normal and is attributed to the fact that when we consider a model with a total of 6 neurons, the small neuron and the noise neuron get lumped together (Fig. 5*A*). It is only when we consider models with 7 neurons or more that the small neuron becomes (roughly) properly identified. That explains the variability in the yellow frames on Fig. 5*B*. The case of the brown neuron is a bit more subtle. This neuron fires very fast with ISIs between 8 and 15 ms (Fig. 9*D*) and does not generate ISIs large enough to properly explore its potential amplitude dynamic range. Stated differently, the amplitude parameters of this neuron are not well defined by the data (see as well Fig. 7). A more relevant feature for this neuron is the "typical" peak amplitudes produced. By that we mean that if we have estimates of the scale s and shape σ parameters of this neuron (10 ms and 0.2), we can compute the most likely ISI: $s \exp(-\sigma^2) = 9.6$ ms. Then, the most likely peak amplitude is given by *Eq. 2*. It turns out that all the models considered gave the same and correct values for the 2 ISI density parameters (within error bars, not shown). The models with 6, 7, and 9 neurons gave for the amplitude parameters (P_1, δ, λ) of the brown neuron: 7.8, 0.45, 114 (the amplitudes on the different recording sites were the same within error bars and the parameters were the same, within error bars, across models, not shown), whereas the model with 8 neurons gave 9.9, 0.45, and 39. Then the most likely ampli-

tude for the models with 6, 7, or 9 neurons was 6.7, whereas it was 6.8 for the model with 8 neurons. That explains the larger frame of the brown neuron on Fig. 5*B*. In practice it turns out that the observation of the parameters alone is sufficient to find which neuron in a given model corresponds to which neuron in another model. Based on the considerations exposed in these last 2 paragraphs it seems reasonable to keep working with a model with 7 neurons, acknowledging the fact that the model cannot account very well for the noise neuron. In the sequel we will therefore proceed with the full data set (5,058) spikes and consider only the model with 7 neurons.

Long runs with full data set

The analysis of the full data set (5,058 spikes or 15 s of data) was done in 2 stages. During the first 4,000 MC steps we fixed the model parameters at their mean values computed from the last 1,000 MC steps of the REM run with the reduced data set (Fig. 4*B*, *black trace*) and we updated only the configuration. The initial configuration was, as usual, randomly set. We used the last 1,000 MC steps of this first stage to take "measurements" and compare the algorithm's output with the actual values of different data statistics (Figs. 8 and 9). The "complete" algorithm was used during the 21,000 MC steps of the second stage. By complete algorithm we mean that at each MC step a new configuration was drawn as well as new values for the model parameters. The 13 sampling points of the posterior conditionals for the amplitude parameters between steps 4,001 and 6,000 were set using the last 1,000 MC steps of the REM run with the reduced data set. The last 2,000 MC steps of the second stage were used to take measurements. We use here a very long run to illustrate the behavior of our algorithm. Such long, and time-consuming, runs are not required for a daily use of our method. The time required to run the first stage (4,000 MC steps) was 3 h, whereas 26 h were required for the second (21,000 MC steps).

THE REM REQUIRES MORE INVERSE TEMPERATURES WHEN THE SAMPLE SIZE INCREASES. Figure 6*A* shows the evolution of the energies at the 15 inverse temperatures used. The energy overlap at adjacent temperatures is clear and is confirmed by the *energy histograms* of Fig. 6*B*. One of the shortcomings of the REM is that it requires more β to be used (for a given range) as the number of spikes in the data set increases because the width of the energy histograms (Fig. 6*B*) is inversely proportional to the square root of the number N, of events in the sample (Hansmann 1997; Hukushima and Nemoto 1996; Iba 2001). The necessary number of β grows therefore as $\sqrt{N}$ (if we had used the same 8 inverse temperatures as during our model exploration with a reduced data set, we would have only every second histogram). The computation time per replica grows, moreover, linearly with N. We therefore end up with a computation time of the REM growing like $N^{1.5}$. This justifies keeping the sample size small during model exploration. Figure 6*C* shows that the autocorrelation function of the energy at each temperature falls to zero rapidly (within 10 MC steps) which is an indication of the good performance of the algorithm from a statistical view point (*Empirical averages and SDs*). A pictorial way to think of the REM is to imagine that several "boxes" at different preset inverse temperatures are used and that there is one and only one replica per box. After each MC step, the algorithm proposes to exchange the replicas

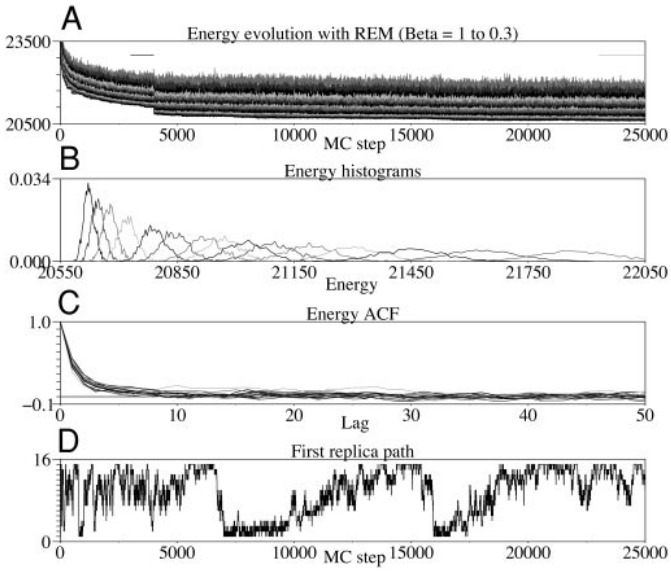

FIG. 6. *A*: energy evolution during a long run with a full data set (5,058 events) and a model with 7 neurons. Fifteen inverse temperatures used were: 1, 0.9, 0.8, 0.7, 0.6, 0.55, 0.5, 0.475, 0.45, 0.425, 0.4, 0.375, 0.35, 0.325, and 0.3. During the first 4,000 steps the algorithm updated only the configuration, not the parameters that were fixed at their average values obtained during the runs with the reduced data set. Black horizontal bar indicates the first measuring period (between steps 3,001 and 4,000), the gray one, the second measuring period (between steps 23,001 and 25,000). *B*: energy histograms obtained from the second measuring period. *Left histogram* corresponds to $\beta = 1$, the right one to $\beta = 0.3$. *C*: energy autocorrelation functions from the second measuring period. *D*: path of the first replica.

located in neighboring boxes (neighboring in the sense of their inverse temperatures) and this proposition can be accepted or rejected (*Eq. A28*). Then if the exchange dynamics works properly one should see each replica visit all the boxes during the MC run. More precisely, each replica should perform a random walk on the available boxes. Figure 6*D* shows the random walk performed by the replica which starts at $\beta = 1$. The ordinate corresponds to the box index (see legend). Between steps 1 and 25,000, the replica travels several times through the entire inverse temperature range.

MODEL PARAMETERS EVOLUTION AND POSTERIOR ESTIMATES. We have until now mainly shown parameters, like the energy of the first replica path, which are likely to be unfamiliar to our reader but our algorithm does generate as well outputs that should be more directly meaningful. We can for instance look at the evolution of the 8 parameters associated with each given neuron. The fluctuations of the parameters values around their means gives us the uncertainty of the corresponding estimates. As explained in *Empirical averages and SDs*, the accuracy of our estimates for a given number of MC steps will be better if the successive values of the corresponding parameters exhibit short autocorrelation times. We can as well use the algorithm output to build the posterior marginal density of each model parameter as shown on Fig. 7. It can be seen that most posterior densities are fairly close to a Gaussian except for 3 neurons whose amplitude parameters are not well defined by the data: neurons 4, 5, and 7. The case of neuron 4, the brown neuron on Fig. 2*B*, has already been discussed in *Early exploration and model choice.* The one of neuron 5 is a bit analogous, except that instead of not generating long enough ISIs it does not generate short enough ones (Fig. 9*E*) to fully explore the

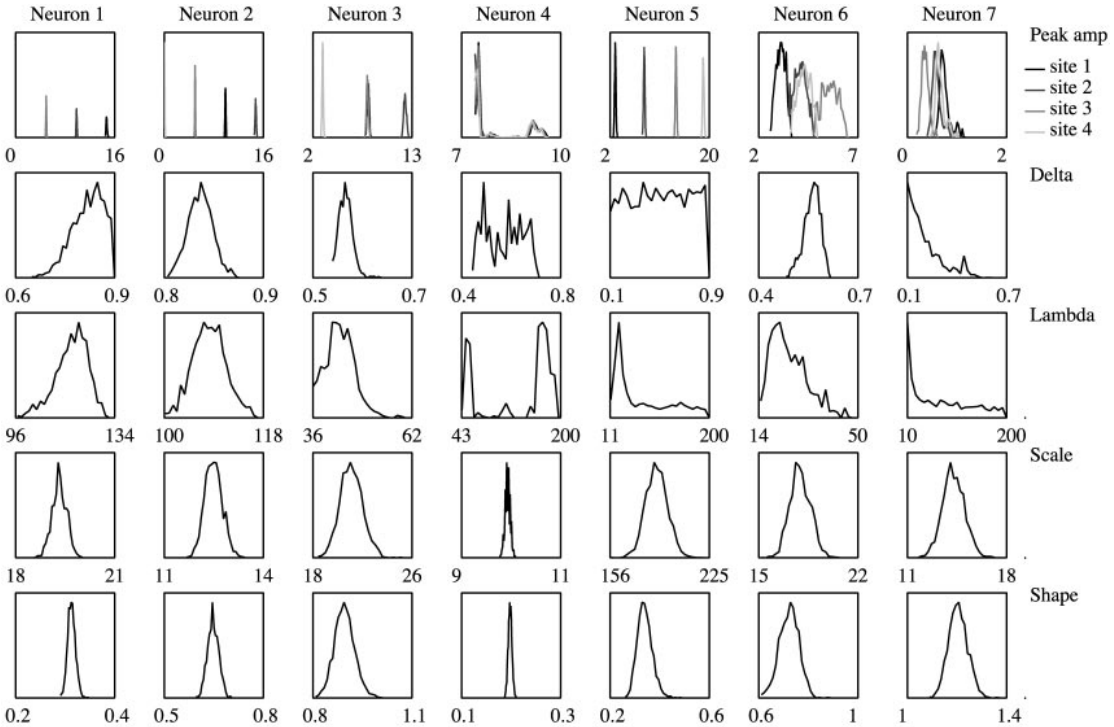

FIG. 7. Marginal posterior density estimate for each parameter of the model computed from the last 2,000 MC steps performed.

amplitude dynamic range. Neuron 7 is the noise neuron and suffers basically from the same problem as the 2 others in addition to the more fundamental one that it is not well described by our model.

A more compact way to present the information of Fig. 7 is to summarize the distributions that are approximately Gaussian by their means and autocorrelation corrected SDs. The other distributions can be summarized by a 95% confidence interval, the left boundary being such that 2.5% of the generated parameters values were below it and the right boundary such that 2.5% of the generated values were above it, as shown in Table 2 (compare with the actual values of Table 1).

SPIKE CLASSIFICATION PERFORMANCES. We can now fully exploit the fact that we are working with simulated data and compare the classification produced by our algorithm with the actual one. To do that easily we first forced the soft classification generated by the algorithm, which gives for each spike the posterior probability to originate from each of the neurons in the model (*Eq. 21*) into the "most likely" classification (see MODEL CHOICE). Then 2 kinds of errors can be made: false negatives and false positives. A false negative (Fig. 8*A*) is a spike actually generated by neuron j, which ends up with the label $i \neq j$. A false positive (Fig. 8*B*) is the reverse situation, a spike that was generated by neuron $i \neq j$ and which ends up with label j. Two false positive and false negative values corresponding to the 2 measurements periods defined in Fig. 6*A* are given for each neuron on Fig. 8. Three features appear already clearly. First, the errors are smaller for the neurons that correspond to the model (neurons 1, 4, and 5) than for the others. Second, the difference between the 2 measurements periods is surprisingly small (given the significant difference in computation times). Third, most of the errors are attributed to noise events (from neuron 7) wrongly attributed to the small neuron (neuron 6), as expected from the Wilson plots. Figure 8*C* shows for each neuron, the sum of false positive and negative divided by the number of spikes actually generated by

TABLE 2. *Values of estimated parameters from the second measurement period*

	Neuron						
Parameter	1	2	3	4	5	6	7
P_1	14.7 ± 0.1	9.9 ± 0.2	8.1 ± 0.1	[7.43, 9.53]	2.9 ± 0.1	3.1 ± 0.7	0.8 ± 0.2
P_2	9.88 ± 0.07	14.8 ± 0.3	12.2 ± 0.2	[7.42, 9.46]	8.2 ± 0.1	4.1 ± 0.4	0.6 ± 0.4
P_3	4.94 ± 0.05	4.98 ± 0.09	8.1 ± 0.1	[7.44, 9.51]	14.0 ± 0.3	5.6 ± 0.7	0.4 ± 0.4
P_4	0.02 ± 0.02	0.06 ± 0.04	3.09 ± 0.07	[7.42, 9.46]	18.9 ± 0.1	4.3 ± 0.7	0.7 ± 0.3
δ	[0.72, 0.9]	0.84 ± 0.03	0.57 ± 0.01	[0.46, 0.69]	[0.12, 0.88]	0.56 ± 0.02	[0.1, 0.46]
λ	[103, 128]	109 ± 10	[36.8, 51.5]	[52, 192]	[17.4, 190]	[16.3, 41]	[10.5, 192]
s	19.3 ± 0.2	12.5 ± 0.2	21 ± 2	9.94 ± 0.08	190 ± 8	18 ± 2	14.4 ± 0.9
σ	0.31 ± 0.01	0.65 ± 0.02	0.90 ± 0.05	0.198 ± 0.006	0.34 ± 0.03	0.7 ± 0.1	1.21 ± 0.06

Each estimated value is given with its SD (*Eq. 26*) when the corresponding posterior density (Fig. 7) is close to a Gaussian. When such is not the case, a 95% confidence interval is given. The maximal peak amplitude values (P_i) are given in units of noise SD. The scale parameters (s) are given in milliseconds. The ISI density parameters (s and σ) are those of the corresponding single log-normal density used by the model.

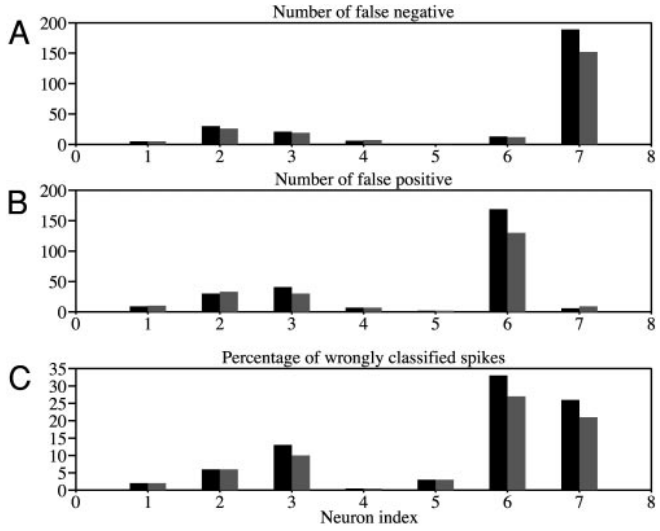

FIG. 8. Comparison between the actual configuration and the most likely configuration generated by the algorithm. Two measurements periods are considered, the first one between steps 3,001 and 4,000 (Fig. 6*A*) represented here in black and the second period (from step 23001 to step 25000) represented in gray. *A*: number of false negative for each neuron. *B*: number of false positive for each neuron. *C*: fraction of wrongly classified spikes for each neuron.

the neuron. The luxury simulation performed with the complete algorithm can be seen to decrease the fraction of misclassified spikes between the small neuron (6) and the noise neuron (7), as well as the fraction of misclassified spikes for the doublet-firing neuron (3). Overall, if we consider the full data set, we end up with 10.4% of the spikes misclassified from the first measurements period and 8.7% from the second one. If we (wisely) decide that neurons 6 and 7 cannot be safely distinguished and decide to keep only the first 5 neurons we misclassify 3.9% of the spikes during the first period and 3.5% during the second.

ISI HISTOGRAMS ESTIMATES. The last statistics shown are the *ISI histograms*. Our algorithm generates at each MC step a new configuration, that is, a new set of labels for the spikes. An ISI histogram can therefore be computed for each neuron after each algorithm step. Then the best estimate we can get for an actual histogram (i.e., the histogram obtained using the true labels) is the average of the histograms computed after each step. This is better than getting first the most likely configuration (as we did in the previous paragraph) and then the histogram for each neuron because this latter method introduces a bias in the estimate. We have done this computation for each neuron during the 2 periods. The results together with the actual histograms are shown on Fig. 9. Again, our algorithm generates very good estimates for the 5 good neurons. In particular it generates the proper strongly bimodal *ISI histogram* for the doublet-firing neuron (Fig. 9*C*), whereas the model does use a *unimodal* log-Normal ISI density to describe this neuron. Here again that gain in accuracy provided by the very long run with the complete algorithm is rather modest.

DISCUSSION

We have described an application of the MCMC methodology to the spike-sorting problem. This approach was recently applied to the same problem (Nguyen et al. 2003) but with a simpler data generation model, which led to a simpler algorithm because successive spikes were considered as independent. We have shown here that a MCMC algorithm allows the neurophysiologist to use much richer data generation models than he/she could previously afford, thereby allowing him/her to include a description of the ISI density as well as a description of the spike waveform dynamics. For the sake of illustration we have used in this paper simple but nontrivial models of ISI densities and waveform dynamics. It should nevertheless be clear that the MCMC approach is very flexible (perhaps too flexible). It is for instance very easy to include more sophisticated models for the neuronal discharge based on hidden Markov chains, like the one underlying the doublet-firing neuron of the present paper (Delescluse et al. unpublished observations). We have shown as well for the first time as far as we know, how to estimate both a probability density for the spike train configuration (i.e., the soft classification) *and* a probability density for the model parameters. The way to use the soft classification to compute statistics of interest (e.g., ISI histograms) was moreover illustrated.

We have shown the presence of meta-stable states, which are, we think, an intrinsic property of models with "interacting" spikes. We have illustrated the potential problems resulting from these meta-stable states, which could introduce a bias in our algorithm's output. However, using a powerful methodology recently introduced in statistics and computational physics, the replica exchange method, we were able to circumvent the slow relaxation resulting from the meta-stable states.

We have deliberately used simulated data that did not correspond to our model assumptions, but which were hopefully realistic enough. Our algorithm with its "simple" data generation model performed very well for the neurons that generated events larger than the detection threshold, even for a neuron which

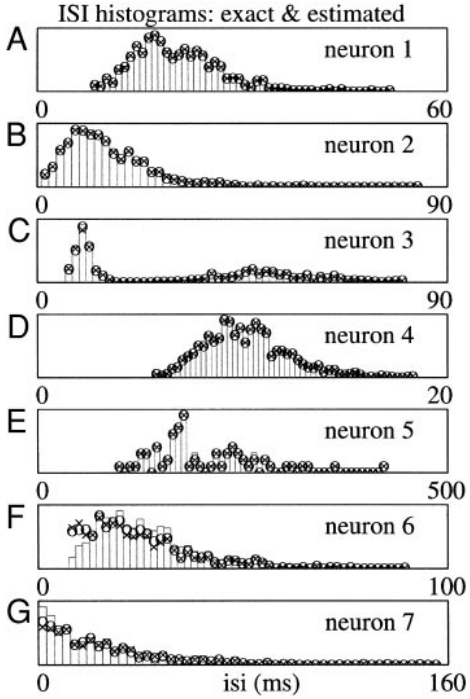

FIG. 9. Comparison between the exact *ISI histograms* and their estimates. *Exact histograms* are displayed in gray, their estimates from the first period with crosses and from the second period with circles. SD for the estimated values is always smaller than the size of the symbols.

generated two separated clusters on the Wilson plots. As far as we know, no other algorithm would automatically properly cluster the events of such a neuron. Clearly, such a performance is possible only if both the spike occurrence times and amplitude dynamics are taken into account. The robustness of the algorithm's output with respect to overclustering was moreover demonstrated. That being said, if a recording noise model based on a Student's *t* distribution turns out to be better than a Gaussian model, our algorithm can be trivially modified to work with the former. It can be easily modified as well to include an explicit "noise neuron" (i.e., a neuron with an exponential ISI density and a maximal amplitude below the detection threshold on all recording sites). We permit reader to download our implementation of the algorithm and check that an almost perfect classification (and model parameter estimation) can be obtained for data that correspond to the model, even when strong overlap between clusters are observed on the Wilson plots.

We have considered spikes described by their peak amplitude(s) instead of their waveforms, although it would not require a tremendous algorithmic change to deal with waveforms (see for instance Pouzat et al. 2002). Caution would nevertheless be required to deal with superpositions of spikes, but as long as spikes do not occur exactly at the same time, superpositions will show up as "interaction" terms. That is, *Eq. 2* would have to be modified to include a baseline change because of neighboring spikes.

The major drawback of our algorithm is the rather long computational time it requires. It should nevertheless be clear that the user can choose between different implementations of our approach. A fast one, where the model parameters are set during a prerun with a reduced data set, leads to an already very accurate classification of spikes from "good" neurons (a good neuron generates events clearly above the detection threshold). We think that this use of our method should satisfy most practitioners. For a high accuracy of the model parameters estimates the "full" version of our algorithm should be chosen. This version is now clearly too time consuming to be systematically used. This long computational time can be easily reduced, however, if one realizes that the REM can be trivially parallelized (Hansmann 1997; Hukushima and Nemoto 1996; Mitsutake et al. 2001). That is, if one has 15 PCs (which is not such a big investment anymore), one can simulate each replica of *Long runs with full data set* on a single CPU and when the replica exchange is accepted just "swap the inverse temperatures of the different CPUs" (rather than swapping the states of replicas simulated on different CPUs). We can therefore expect to reduce the computation time by an order of magnitude (even more if one considers that recent CPUs run at 3 GHz, whereas the one used in this paper ran at 1.6 GHz). Preliminary tests on a Beowulf cluster confirmed this expectation. Other improvements can be brought as well, in particular for the amplitude parameters generation. For the latter, a Langevin diffusion or a simple random walk can be used to generate the proposed moves (Besag 2001; Celeux et al. 2000; Liu 2001; Neal 1993). We have in fact already done it (Pouzat et al. unpublished observations) and found that the computation time required for the amplitude parameters was reduced by a factor of 10 (compared with a piecewise linear approximation of the posterior conditional based on 13 sample points).

Finally, as we mentioned in the INTRODUCTION, our method is semiautomatic because user input is still required to choose the "correct" number of active neurons in the data. To make the method fully automatic we will have to do better and that fundamentally means estimating the normalizing constant Z of *Eqs. 12* and *13*, which is the probability of the data. Indeed Bayesian model comparison requires the comparison (ratio) of data probability under different models (Gelman and Meng 1998; Green 1995) as explained, in a spike-sorting context, by Nguyen et al. (2003). This task of estimating Z could seem daunting but luckily is not. As we said, Dynamic MC and MCMC methods have been used for a long time, which means that other people in other fields already had that problem. Among the proposed solutions (Gelman and Meng 1998; Green 1995; Nguyen et al. 2003), what physicists call thermodynamic integration (Frenkel and Smit 2002; Landau and Binder 2000) seems very attractive because it is known to be robust and it requires simulations to be performed between $\beta = 1$ and $\beta = 0$. That is precisely what we are already (partially) doing with the REM. Of course the estimation of Z is only half of the problem. The prior distribution on the number of neurons in the model has to be set properly as well. Increasing the number of neurons will always lead to a decrease in energy which will give larger Z values (the derivative of the log of Z being proportional to the opposite of the energy; Frenkel and Smit 2002), we will therefore have to compensate for this systematic Z increase with the prior distribution (Pouzat et al. unpublished observations).

APPENDIX

The Metropolis–Hastings algorithm

The theory of Markov chains (Brémaud 1998) tells us that given a probability density like π_{post}, and a transition kernel T, an equation like *Eq. 20* will hold for any function E and for any initial "state" $[\theta^{(0)}, C^{(0)}]$, if T is irreducible and aperiodic and if π_{post} is *stationary* with respect to T, that is, if

$$\sum_{C_a} \int_{\theta_a} \mathrm{d}\theta_a \pi_{post}(\theta_a, C_a|Y) T(\theta_b, C_b|\theta_a, C_a) = \pi_{post}(\theta_b, C_b|Y) \qquad (A1)$$

In words, *irreducible* means that the Markov chain obtained by applying T repetitively can reach any "state" (θ, C) in a finite number of steps. *Aperiodic* means, loosely speaking, that there is no temporal order in the way the chain returns to any state it visited. These 2 properties are often lumped together in the notion of *ergodicity*. In practice, if a transition kernel is nonzero for any pair of arbitrary states (θ_a, C_a) and (θ_b, C_b), then it is both irreducible and aperiodic (ergodic). Our problem thus becomes to find an ergodic T for which *Eq. A1* holds.

It turns out that there is a very general prescription to build a transition kernel T with the desired properties and that, in fact, we do not even have to build it explicitly. We first need a "proposal" transition kernel $g[\theta, C|\theta^{(t-1)}, C^{(t-1)}]$, which is itself ergodic on the same space as π_{post}.

We then apply the following algorithm:

Given a state $[\theta^{(t-1)}, C^{(t-1)}]$

Generate:

$$(\theta_{proposed}, C_{proposed}) \sim g[\theta, C|\theta^{(t-1)}, C^{(t-1)}] \qquad (A2)$$

Take:

$$[\theta^{(t)}, C^{(t)}] = \begin{cases} (\theta_{proposed}, C_{proposed}) & \text{with probability} \\ [\theta^{(t-1)}, C^{(t-1)}] & \text{with probability} \end{cases} \begin{matrix} A[\theta_{proposed}, C_{proposed}|\theta^{(t-1)}, C^{(t-1)}] \\ \{1 - A[\theta_{proposed}, C_{proposed}|\theta^{(t-1)}, C^{(t-1)}]\} \end{matrix}$$

where the *acceptance probability* A is defined by

$$A[\theta_{proposed}, C_{proposed}|\theta^{(t-1)}, C^{(t-1)}] = \min\left\{1, \frac{\pi_{post}(\theta_{proposed}, C_{proposed}|Y)g[\theta^{(t-1)}, C^{(t-1)}|\theta_{proposed}, C_{proposed}]}{\pi_{post}[\theta^{(t-1)}, C^{(t-1)}|Y]g[\theta_{proposed}, C_{proposed}|\theta^{(t-1)}, C^{(t-1)}]}\right\} \quad (A3)$$

It is not hard to show that the transition kernel induced by this algorithm has the desired properties (Liu 2001; Robert and Casella 1999). The important feature of this procedure is that its implementation requires only a knowledge of π_{post} *up to a normalizing constant* because *Eq. A3* involves a ratio of π_{post} values in 2 states. In other words it requires only the capability to compute the energy of any state. An initial version of this algorithm was given by Metropolis et al. (1953) and the present version is attributed to Hastings (1970).

The reader can see that the Metropolis–Hastings (MH) algorithm is indeed a very large family of algorithms because there exists a priori an infinite number of transitions g from which a specific procedure can be built. We therefore have to find at least one g and if we find several we would like to have a way to compare the resulting MH transition kernels (T). A common way to proceed for complicated models, like the one we are presently dealing with, is to split the transition kernel T in a series of parameter and label specific transition kernels (Besag 2001; Fishman 1996). Let us define

$$\theta_{-i} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_{n_p}) \quad (A4)$$

where n_p is the number of parameters in the model and

$$C_{-i} = (c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_N) \quad (A5)$$

Then the parameter-specific transition kernels are objects like

$$T_{\theta_{-i}}(\theta_{-i}, \theta_i = b, C|\theta_{-i}, \theta_i = a, C) \quad (A6)$$

and the label specific transition kernels

$$T_{C_{-i}}(\theta, C_{-i}, c_i = b|\theta, C_{-i}, c_i = a) \quad (A7)$$

We will often use the following short notation for *Eqs. A6* and *A7*

$$T_{\theta_{-i}}(\theta_i = b|\theta_i = a) \equiv T_{\theta_{-i}}(\theta_{-i}, \theta_i = b, C|\theta_{-i}, \theta_i = a, C) \quad (A8)$$

and

$$T_{C_{-i}}(c_i = b|c_i = a) \equiv T_{C_{-i}}(\theta, C_{-i}, c_i = b|\theta, C_{-i}, c_i = a) \quad (A9)$$

A "complete" transition kernel can be constructed with, for instance, a sequential application of every label and parameter specific transition

$$T = T_{C_{-1}} \cdot \cdot T_{C_{-N}} T_{\theta_{-1}} \cdot \cdot T_{\theta_{-n_p}} \quad (A10)$$

In the sequel we will say that one *MC time step* has been performed every time a complete transition kernel has been applied to the "present" state of the Markov chain. Sufficient conditions the parameter and label specific transitions must exhibit for the complete transition T to be ergodic and for π_{post} to be stationary with respect to T are as follows:

1. Each parameter and label specific transition ($T_{\theta_{-i}}$, $T_{C_{-j}}$) is such that

$$T_{\theta_{-i}}(\theta_i = b|\theta_i = a) \neq 0 \quad (A11)$$

for all pairs of states (θ_{-i}, $\theta_i = a$, C) and (θ_{-i}, $\theta_i = b$, C) such that

$$\pi_{post}(\theta_{-i}, \theta_i = a, C|Y) > 0 \quad \text{and} \quad \pi_{post}(\theta_{-i}, \theta_i = b, C|Y) > 0 \quad (A12)$$

(It is straightforward to write the equivalent condition for $T_{C_{-j}}$)

2. Each parameter and label specific transition ($T_{\theta_{-i}}$, $T_{C_{-j}}$) is such that the *detailed balance* condition is met

$$\pi_{post}(\theta_{-i}, \theta_i = a, C|Y)T_{\theta_{-i}}(\theta_i = b|\theta_i = a) = \pi_{post}(\theta_{-i}, \theta_i = b, C|Y)T_{\theta_{-i}}(\theta_i = a|\theta_i = b) \quad (A13)$$

Moreover, we have

$$\pi_{post}(\theta_{-i}, \theta_i = a, C|Y) = \pi_{post}(\theta_i = a|\theta_{-i}, C, Y)\pi_{post}(\theta_{-i}, C|Y)$$

and *Eq. A12* implies that

$$\pi_{post}(\theta_i = a|\theta_{-i}, C, Y) > 0 \quad \text{and} \quad \pi_{post}(\theta_{-i}, C|Y) > 0$$

We can therefore rewrite the detailed balance condition as follows

$$\pi_{post}(\theta_i = a|\theta_{-i}, C, Y)T_{\theta_{-i}}(\theta_i = b|\theta_i = a) = \pi_{post}(\theta_i = b|\theta_{-i}, C, Y)T_{\theta_{-i}}(\theta_i = a|\theta_i = b) \quad (A14)$$

where $\pi_{post}(\theta_i|\theta_{-i}, C, Y)$ is the posterior conditional density of θ_i. There are, of course, equivalent equations for the label-specific transition kernels ($T_{C_{-i}}$).

A straightforward way to obtain $T_{\theta_{-i}}$ and $T_{C_{-i}}$ with the desired properties is to find parameter- and label-specific proposal transition kernels ($g_{\theta_{-i}}$ and $g_{C_{-i}}$) respecting *Eq. A11* in combination with an acceptance probability like *Eq. A3*, that is, to have an MH rule for each parameter of the model and for each label of the configuration. It is indeed easy to show that such a construct gives rise to $T_{\theta_{-i}}$ and $T_{C_{-i}}$ respecting the detailed balance condition because, assuming $a \neq b$

$$\begin{aligned}
&\pi_{post}(\theta_{-i}, \theta_i = a, C|Y)T_{\theta_{-i}}(\theta_i = b|\theta_i = a)\\
&= \pi_{post}(\theta_{-i}, \theta_i = a, C|Y)g_{\theta_{-i}}(\theta_i = b|\theta_i = a)A_{\theta_{-i}}(\theta_i = b|\theta_i = a)\\
&= \min\left[\pi_{post}(\theta_{-i}, \theta_i = a, C|Y)g_{\theta_{-i}}(\theta_i = b|\theta_i = a), \pi_{post}(\theta_{-i}, \theta_i = b, C|Y)g_{\theta_{-i}}(\theta_i = a|\theta_i = b)\right]\\
&= \pi_{post}(\theta_{-i}, \theta_i = b, C|Y)g_{\theta_{-i}}(\theta_i = a|\theta_i = b)A_{\theta_{-i}}(\theta_i = a|\theta_i = b)\\
&= \pi_{post}(\theta_{-i}, \theta_i = b, C|Y)T_{\theta_{-i}}(\theta_i = a|\theta_i = b)
\end{aligned}$$

To complete this justification of this parameter and label specific decomposition the reader needs to realize that the detailed balance condition is stronger than the stationarity condition (*Eq. A1*) and in fact implies it as shown in the following paragraph.

*The detailed balance condition imposed on the parameter specific transition kernels implies the stationarity condition on the resulting "complete" transition kernel.*We present here the justification for a simple model with 2 parameters (θ_1 and θ_2) and without latent variable, the extension to the full model considered in this paper being straightforward. So we have

$$T = T_{\theta_{-1}} T_{\theta_{-2}}$$

with the detailed balance conditions:

$$\pi_{post}(\theta_1 = a, \theta_2 = a')T_{\theta_{-1}}(\theta_1 = b|\theta_1 = a) = \pi_{post}(\theta_1 = b, \theta_2 = a')T_{\theta_{-1}}(\theta_1 = a|\theta_1 = b) \quad (A15)$$

and

$$\pi_{post}(\theta_1 = a, \theta_2 = a')T_{\theta_{-2}}(\theta_2 = b'|\theta_2 = a') = \pi_{post}(\theta_1 = a, \theta_2 = b')T_{\theta_{-2}}(\theta_2 = a'|\theta_2 = b') \quad (A16)$$

We then have

$$\int_{a'}\int_{a} da' da\, \pi_{post}(a, a')T_{\theta_{-1}}(b|a)T_{\theta_{-2}}(b'|a') \quad (A17)$$

$$= \int_{a'} da'\, T(b'|a')\left[\int_{a} da\, \pi_{post}(a, a')T_{\theta_{-1}}(b|a)\right]$$

$$= \int_{a'} da' T_{\theta_{-2}}(b'|a')[\int_{a} da \pi_{post}(b, a') T_{\theta_{-1}}(a|b)] \quad (A18)$$

$$= \int_{a'} da' T_{\theta_{-2}}(b'|a') \pi_{post}(b, a')[\int_{a} da T_{\theta_{-1}}(a|b)] \quad (A19)$$

$$= \int_{a'} da' T_{\theta_{-2}}(b'|a') \pi_{post}(b, a') \quad (A20)$$

$$= \int_{a'} da' T_{\theta_{-2}}(a'|b') \pi_{post}(b, b') \quad (A21)$$

$$= \pi_{post}(b, b') \int_{a'} da' T_{\theta_{-2}}(a'|b') \quad (A22)$$

$$= \pi_{post}(b, b') \quad (A23)$$

where the detailed balance conditions (*Eqs. A15* and *A16*) have been used to go from *Eq. A17* to *Eq. A18* and from *Eq. A20* to *Eq. A21*, respectively and the normalization condition for a proper transition kernel

$$\int_{\theta} d\theta T(\theta|\theta') = 1$$

has been used to go from *Eq. A19* to *Eq. A20* and from *Eq. A22* to *Eq. A23*.

It should become clear in the sequel that the appeal of the detailed balance comes from the (relative) ease with which it can be implemented (and checked). This is not the case of the stationarity condition which is much harder to check for an arbitrary transition kernel.

A close examination of the MH acceptance probability shows that a reasonable choice for the parameter specific proposal transition would be the corresponding conditional posterior density. For then we would have

$$A_{\theta_{-i}}(b|a) = min\left[1, \frac{\pi_{post}(\theta_{-i}, b, C|Y) \cdot \pi_{post}(a|\theta_{-i}, C, Y)}{\pi_{post}(\theta_{-i}, a, C|Y) \cdot \pi_{post}(b|\theta_{-i}, C, Y)}\right]$$

$$= min\left[1, \frac{\pi_{post}(b|\theta_{-i}, C, Y) \cdot \pi_{post}(\theta_{-i}C|Y) \cdot \pi_{post}(a|\theta_{-i}, C, Y)}{\pi_{post}(a|\theta_{-i}, C, Y) \cdot \pi_{post}(\theta_{-i}C|Y) \cdot \pi_{post}(b|\theta_{-i}, C, Y)}\right]$$

$$= 1$$

That is, with the posterior conditional as a proposal transition kernel the acceptance probability is always 1. This type of parameter specific transition is called *heat-bath algorithm* by physicists (Landau and Binder 2000; Newman and Barkema 1999; Sokal 1989) and *Gibbs sampler* by statisticians (Gelman et al. 1995; Liu 2001; Robert and Casella 1999). It has already been used in a neurophysiological context, for single-channel studies (Ball et al. 1999; Rosales et al. 2001) and even for spike-sorting (Nguyen et al. 2003). It is in general a good first trial when one wants to build a Markov chain on a parameter and configuration space like the one we are considering here. Its main drawback is that it requires the expression of the posterior conditional to be available in a closed form. For the present model, such a closed form posterior conditional is available for the labels and for the scale and shape parameters of the ISI density, but not for the "amplitude parameters" (**P**, δ, λ). The following section of the APPENDIX contains a detailed account of the algorithms used for the labels transition kernels (SPIKE LABEL TRANSITION MATRIX), the scale parameter transition kernel (SCALE PARAMETER TRANSITION KERNEL), the shape parameter transition kernel (SHAPE PARAMETER TRANSITION KERNEL), and the amplitude parameters transition kernels (AMPLITUDE PARAMETERS TRANSITION KERNELS). The algorithm of the later uses as proposal transition kernels, piecewise linear approximations of the posterior conditional densities. These piecewise linear approximations are computationally costly (they take a lot of time to compute) when one uses a lot of discrete posterior conditional evaluations, although they are then better approximations. To keep both a good precision and a reasonable computational cost we used a multiruns procedure where the sample generated during one run was used to optimize the few (13) locations at which the posterior conditional was evaluated (AMPLITUDE PARAMETERS TRANSITION KERNELS). This "position refinement" procedure is one of the reasons why we repetitively use successions of runs in the results section.

Parameter-specific transition kernels

SPIKE LABEL TRANSITION MATRIX. Here c_i takes integer values in $\{1, \ldots, K\}$, so we are therefore, strictly speaking, dealing with a transition matrix rather than a transition kernel. As explained in *The Metropolis–Hastings algorithm* we build $T_{C_{-i}}$ as a product of a proposal transition matrix $g_{C_{-i}}$ and an acceptance probability "matrix" $A_{C_{-i}}$. We always propose a label different from the present one [$g_{C_{-i}}(c_i = a|c_i = a) = 0$], and the "new" labels are proposed according to their posterior conditional density

$$g_{C_{-i}}(c_i = b|c_i = a) = \frac{\pi_{post}(c_i = b|\theta, C_{-i}, Y)}{\sum_{l \neq a} \pi_{post}(c_i = l|\theta, C_{-i}, Y)} \quad (A24)$$

The acceptance probability is defined by *Eq. A3* and gives here

$$A_{C_{-i}}(c_i = b|c_{i=a}) = \min\left[1, \frac{\sum_{l \neq a} \pi_{post}(c_i = l|\theta, C_{-i}, Y)}{\sum_{l \neq b} \pi_{post}(c_i = l|\theta, C_{-i}, Y)}\right] \quad (A25)$$

We will illustrate the computation on the simple case presented in LIKELIHOOD FUNCTION and Fig. 1, with one recording site and 2 neurons in the model ($K = 2$). Because c_i can only have 2 values, $c_i = 1$ and $c_i = 2$, the matrix $g_{C_{-i}}$ is straightforward to write

$$g_{C_{-i}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The acceptance probability requires the calculation of the ratio of the posterior conditionals: $\pi_{post}(c_i = 1|\theta, C_{-i}, Y)$ and $\pi_{post}(c_i = 2|\theta, C_{-i}, Y)$. From *Eq. 15* we have

$$\pi_{post}(c_i = 1|\theta, C_{-i}, Y) \propto L(Y, C_{-i}, c_i = 1|\theta)\pi_{prior}(\theta)$$

$$\pi_{post}(c_i = 2|\theta, C_{-i}, Y) \propto L(Y, C_{-i}, c_i = 2|\theta)\pi_{prior}(\theta)$$

Therefore

$$A_{C_{-i}}(c_i = 2|c_i = 1) = \min\left[1, \frac{L(Y, C_{-i}, c_i = 2|\theta)}{L(Y, C_{-i}, c_i = 1|\theta)}\right]$$

If we then assume for instance that

$$\frac{L(Y, C_{-i}, c_i = 2|\theta)}{L(Y, C_{-i}, c_i = 1|\theta)} = 0.75$$

we get the following transition matrix

$$T_{C_{-i}} = \begin{pmatrix} 0.25 & 0.75 \\ 1 & 0 \end{pmatrix}$$

In the case of Fig. 1, if we consider $i = 13$, we get, using *Eq. 5*

$$L(Y, C_{-13}, c_{13} = 1|\theta)$$
$$= M\pi(t_{13} - t_{12}, a_{13}|\sigma_1, s_1, P_1, \delta_1, \lambda_1) \times \pi(t_{15} - t_{13}, a_{15}|\sigma_1, s_1, P_1, \delta_1, \lambda_1)\pi(t_{14} - t_{11}, a_{14}|\sigma_2, s_2, P_2, \delta_2, \lambda_2)$$

and

$$L(Y, C_{-13}, c_{13} = 2|\theta)$$
$$= M\pi(t_{15} - t_{12}, a_{15}|\sigma_1, s_1, P_1, \delta_1, \lambda_1) \times \pi(t_{13} - t_{11}, a_{13}|\sigma_2, s_2, P_2, \delta_2, \lambda_2)\pi(t_{14} - t_{13}, a_{14}|\sigma_2, s_2, P_2, \delta_2, \lambda_2)$$

where M represents all the terms that are *identical* in the 2 equations.

*A remark on numerical implementation.*To avoid rounding errors and keep numerical precision we work with the log-likelihoods

$$l_b = \log [L(Y, C_{-i}, c_i = b|\theta)]$$

look for l_{max}, the maximum (with respect to b) of l_b and obtain $\pi_{post}(c_i = b|\theta, C_{-i}, Y)$ with

$$\pi_{post}(c_i = b|\theta, C_{-i}, Y) = \frac{\exp(l_b - l_{max})}{\sum_{q=1}^{K} \exp(l_q - l_{max})}$$

A proposed value $b \in \{1, \ldots, K\}\backslash\{a\}$ is drawn from the probability mass $g_{C_{-i}}(c_i = b|c_i = a)$ using the inverse cumulative distribution function (Fishman 1996; Gelman et al. 1995; Robert and Casella 1999).

Boundary treatment. From the statisticians' viewpoint the spike trains we are dealing with are *censored* data. They mean by that that there are missing data without which we cannot, strictly speaking, compute our likelihood function. The missing data here are the last spikes from each neuron that occurred before our recording started and the next spikes after our recording stopped. There are proper approaches to deal with such censored data (Gelman et al. 1995; Robert and Casella 1999) and we plan to implement them in a near future. In the present study we used a simple and quick way to deal with the end effects in our data. We used periodic boundary conditions; that is, the last spike from each neuron was considered as the spike preceding the first spike of the same neuron.

SCALE PARAMETER TRANSITION KERNEL. We consider here cases where θ_i of *Eq. A14* corresponds to one of the K scale parameters, s_q, where q stands for the neuron index. We will first show that the corresponding posterior conditional: $\pi_{post}(\theta_i = s_q|\theta_{-i}, C, Y)$ can be written in a closed form. From *Eq. 15* we have

$$\pi_{post}(\theta_i = s_q|\theta_{-i}, C, Y) \propto L(Y, C|\theta_i = s_q, \theta_{-i})\pi_{prior}(\theta_i = s_q, \theta_{-i})$$

Going back to the structure of the likelihood function (*Eq. 9*), one sees, using *Eqs. 5* and *10*, that the likelihood function with the labels set and all model parameters set except the scale parameter of one neuron q depends only on the events that are labeled as belonging to neuron q, and on the shape parameter of q, σ_q. If we write $\{i_{q,1}, \ldots, i_{q,n_q}\}$, the ISI computed from the spike train of neuron q in configuration C, we get

$$\pi_{post}(\theta_i = s_q|\theta_{-i}, C, Y) \propto \exp\left[-\frac{1}{2} \sum_{j=1}^{n_q} \left(\frac{\log i_{q,j} - \log s_q}{\sigma_q} \right)^2 \right]$$

If we introduce: $\overline{\log i_q} = 1/n_q \sum_{j=1}^{n_q} \log i_{q,j}$, a short manipulation shows that

$$\pi_{post}(\theta_i = s_q|\theta_{-i}, C, Y) \propto \exp\left[-\frac{1}{2} \frac{n_q(\overline{\log i_q} - \log s_q)^2}{(\sigma_q)^2} \right]$$

In words: the posterior density of the log of the scale parameter s_q is Normal with mean: $\overline{\log i_q}$, and variance, $(\sigma_q)^2/n_q$. Our prior on s_q is a uniform distribution between s_{min} and s_{max} (PRIOR DENSITY); we can therefore, using the Gibbs algorithm, generate s_q as follows:

1.) Given C and Y: compute n_q and $\overline{\log i_q}$
2.) Generate $u \sim$ Norm $[\overline{\log i_q}, (\sigma_q)^2/n_q]$
3.) Compute $s = \exp(u)$
4.) If $s \in [s_{min}, s_{max}]$, set $s_q = s$; otherwise, go back to 2

SHAPE PARAMETER TRANSITION KERNEL. Here θ_i is the shape parameter of neuron q, $\theta_i = \sigma_q$. We again have to show that the posterior conditional can be written in a closed form. Using the approach of the previous section we quickly get

$$\pi_{post}(\theta_i = \sigma_q|\theta_{-i}, C, Y) \propto \frac{1}{(\sigma_q)^{n_q}} \exp\left[-\frac{n_q}{2} \frac{(\overline{\log i_q} - \log s_q)^2}{(\sigma_q)^2} \right]$$

Then an identification with an inverse-Gamma density gives us the answer

$$\pi_{inverse-Gamma}(\kappa|\alpha, \gamma) = \frac{\gamma^\alpha}{\Gamma(\alpha)} \kappa^{-(\alpha+1)} \exp\left(-\frac{\gamma}{\kappa} \right)$$

if we use $\kappa = (\sigma_q)^2$, $\alpha = (n_q/2) - 1$, and $\gamma = n_q/2(\overline{\log i_q} - \log s_q)^2$. To generate σ_q with the Gibbs algorithm we need to know how to generate random number with an inverse-Gamma density. It turns out that most analysis platforms (Scilab, MATLAB) and *C* libraries like GSL do not have an inverse-Gamma random number generator but do have a Gamma random number generator. A Gamma density can be written

$$\pi_{Gamma}(\zeta|\alpha, \gamma) = \frac{\gamma^\alpha}{\Gamma(\alpha)} \zeta^{\alpha-1} \exp(-\gamma\zeta)$$

and it is not hard to show that if ζ is Gamma with parameters α and γ, then $1/\zeta$ is inverse-Gamma with the same parameters. To generate σ_q using the Gibbs algorithm, we therefore perform

1.) Generate $\zeta \sim$ *Gamma* $[(n_q/2) - 1, n_q/2(\overline{\log i_q} - \log s_q)^2]$
2.) if $\sigma_{min} \leq \sqrt{1/\zeta} \leq \sigma_{max}$, then $\sigma_q = \sqrt{1/\zeta}$ otherwise go back to 1.

AMPLITUDE PARAMETERS TRANSITION KERNELS. As explained in *The Metropolis–Hastings algorithm* the amplitude parameters: **P**, δ, λ cannot be generated with the Gibbs algorithm because the corresponding posterior conditionals cannot be written in a closed form. For these parameters we had recourse to a piecewise linear proposal kernel. The principle of our method is illustrated here with a simple example, which can be thought of as the generation of the maximal peak amplitude of one of the neurons (q) on, say, the first recording site. Using a discrete set of points on the domain allowed by the corresponding priors: $\{P_{q,1,1} = P_{min}, P_{q,1,2}, \ldots, P_{q,1,m} = P_{max}\}$, an unnormalized value of the posterior conditional is computed

$$\pi_{post}(P_{q,1,i}|\theta_{-P_{q,1}}, C, Y) \propto L(Y, C|P_{q,1,i}, \theta_{-P_{q,1}})$$

Again the structure of the likelihood (*Eq. 9*) allows us to write

$$L(Y, C|P_{q,1,i}, \theta_{-P_{q,1}}) \propto L(Y_q, C|P_{q,1,i}, P_{q,2}, P_{q,3}, P_{q,4}, \delta_q, \lambda_q)$$

where we are assuming we are dealing with a tetrode recording. These unnormalized posterior values are then normalized such that the piecewise linear function:

$$g_{P_{q,1}}(P) = \pi_{post}(P_{q,1,i}|\theta_{-P_{q,1}}, C, Y) + \frac{\pi_{post}(P_{q,1,i+1}|\theta_{-P_{q,1}}, C, Y) - \pi_{post}(P_{q,1,i}|\theta_{-P_{q,1}}, C, Y)}{P_{q,1,i+1} - P_{q,1,i}} (P - P_{q,1,i})$$

where $P \in [P_{q,1,i}, P_{q,1,i+1}]$ sums to 1. Of course when we start the algorithm we do not know what is the best location for the discrete points $P_{q,1,i}$. We therefore used during a first run 100 regularly spaced points, although it turns out that computing at *each MC step* 100 values of the posterior conditional (for each amplitude parameter of the model) takes time. It is therefore very interesting, to speed up the sorting procedure, to use fewer points, which basically means forgetting about the points located where the posterior conditional is extremely small. To achieve this goal, at the end of each run, the last 1000 generated values of the parameter of interest ($P_{q,1}$) were used to get an estimate of the posterior marginal cumulative distribution of the parameter. Then 10 quantiles were located on this cumulative distribution, one discrete point was taken slightly smaller than the smallest generated value for that parameter, and the two extreme values al-

lowed by the priors were used as discrete points as well. The new piecewise proposal kernel was therefore based on a sparser approximation.

Slow relaxation and REM

One of the nice features of the MCMC approach, and in fact the key theoretical result justifying its growing popularity, is that the user can be sure that empirical averages (*Eq. 19*) computed from the samples it generates do converge to the corresponding expected values (*Eq. 18*). The not-so-nice feature, forgetting about the actual implementation details, which can be tedious, is that the user can run his/her MCMC algorithm only for a finite amount of time. Then the fact that empirical averages are *approximations* cannot be ignored. There are 3 sources of errors in such estimates. The first that comes to mind is the one resulting from the finite sample size, that is, the one we would have even if our draws where directly generated from π_{post}. The difference between the MCMC-based estimates and estimates based on direct samples is that the variance of the estimators of the former have to be corrected to take into account the correlation of the states generated by the Markov chain, as explained in *Empirical averages and SDs*. The second source of error is a bias induced by the state $[\theta^{(0)}, C^{(0)}]$ with which the chain is initialized (Fishman 1996; Sokal 1989). The bad news concerning this source of error is that there is no general theoretical result providing guidance on the way to handle it, but the booming activity in the Markov chain field already produced encouraging results in particular cases (Liu 2001). The common wisdom in the field is to monitor parameters (and labels) evolution, and/or their functions like the energy (*Eq. 16*). Based on examination of evolution plots (e.g., Fig. 4, *A* and *B*, Fig. 6*A*) and/or on application of time-series analysis tools, the user will decide that "equilibrium" has been reached and discard the parameters values before equilibrium. More sophisticated tests do exist (Robert and Casella 1999) but they were not used in this paper. These first 2 sources of error are common to all MCMC approaches. The third one appears only when the energy function exhibits several local minima. In the latter case, the Markov chain realization can get trapped in a local minimum, which could be a poor representation of the whole energy function. This sensitivity to local minima arises from the local nature of the transitions generated by the MH algorithm. That is, in our case, at each MC time step, we first attempt to change the label of spike 1, then the one of spike 2, and so forth, then the one of spike *N*, then we try to change the first component of $\theta(P_{1,1})$, and so on until the last component (σ_K). That implies that if we start in a local minimum and if we need to change, say, the labels of 10 spikes to reach another lower local minimum, we could have a situation in which the first 3 label changes are energetically unfavorable (giving, for instance, an acceptance probability, *Eq. A3*, of 0.1 per label change), which would make the probability to accept the succession of changes very low (0.1^3), meaning that our Markov chain would spend a long time in the initial local minimum before "escaping" to the neighboring one. Stated more quantitatively, the average time the chain will take to escape from a local minimum with energy E_{min} grows as the exponential of the energy difference between the energy E^* of the highest energy state the chain has to go through to escape and E_{min}

$$\tau_{escape} \propto \exp[\beta(E^* - E_{min})]$$

Our chains will therefore exhibit an Arrhenius behavior. To sample more efficiently such state spaces, the replica exchange method (REM) (Hukushima and Nemoto 1996; Mitsutake et al. 2001), also known as the parallel tempering method (Falcioni and Deem 1999; Hansmann 1997; Yan and de Pablo 1999), considers *R* replicas of the system with an increasing sequence of temperatures (or a decreasing sequence of β) and a dynamic defined by 2 types of transitions: usual MH transitions performed independently on each replica according to the rule defined by *Eq. A10* and a replica exchange transition. The key idea is that the high temperature (low β) replicas will be able to easily cross the energy barriers separating local minima (in the example above, if we had a probability of 0.1^3 to accept a sequence of labels switch for the replica at $\beta = 1$, the replica at $\beta = 0.2$ will have a probability $0.1^{3\times0.2} \approx 0.25$ to accept the same sequence). What is needed is a way to generate replica exchange transitions such that the replica at $\beta = 1$ generates a sample from π_{post} defined by *Eq. 17*. Formally the REM consists in simulating, on an "extended ensemble," a Markov chain whose unique stationary density is given by

$$\pi_{ee}(\theta_1, C_1, \ldots, \theta_R, C_R) = \pi_{post,\beta_1}(\theta_1, C_1) \cdots \pi_{post,\beta_R}(\theta_R, C_R) \qquad (A26)$$

where "*ee*" in π_{ee} stands for "extended ensemble" (Iba 2001), *R* is the number of simulated replicas, $\beta_1 > \cdots > \beta_R$ for convenience, and

$$\pi_{post,\beta_i}(\theta_i, C_i) = \frac{\exp[-\beta_i E(\theta_i, C_i)]}{Z(\beta_i)} \qquad (A27)$$

That is, compared to *Eq. 17*, we now explicitly allow β to be different from 1. To construct our "complete" transition kernel we apply our previous procedure; that is, we construct it as a sequence of parameter, label, and interreplica-specific MH transitions. We already know how to obtain the parameter and label specific transitions for each replica. What we really need is a transition to exchange replicas, say the replicas at inverse temperature β_i and β_{i+1}, such that the detailed balance is preserved (*Eq. A13*)

$$\begin{aligned}
&\pi_{ee}(\theta_1, C_1, \ldots, \theta_i, C_i, \theta_{i+1}, C_{i+1}, \ldots, \theta_R, C_R) T_{i,i+1}(\theta_{i+1}, C_{i+1}, \\
&\qquad \theta_i, C_i | \theta_i, C_i, \theta_{i+1}, C_{i+1}) \\
&= \pi_{ee}(\theta_1, C_1, \ldots, \theta_{i+1}, C_{i+1}, \theta_i, C_i, \ldots, \theta_R, C_R) T_{i,i+1}(\theta_i, C_i, \theta_{i+1}, C_{i+1} | \\
&\qquad \theta_{i+1}, C_{i+1}, \theta_i, C_i)
\end{aligned}$$

which leads to

$$\begin{aligned}
&\frac{T_{i,i+1}(\theta_i, C_i, \theta_{i+1}, C_{i+1} | \theta_{i+1}, C_{i+1}, \theta_i, C_i)}{T_{i,i+1}(\theta_{i+1}, C_{i+1}, \theta_i, C_i | \theta_i, C_i, \theta_{i+1}, C_{i+1})} \\
&= \frac{\pi_{ee}(\theta_1, C_1, \ldots, \theta_i, C_i, \theta_{i+1}, C_{i+1}, \ldots, \theta_R, C_R)}{\pi_{ee}(\theta_1, C_1, \ldots, \theta_{i+1}, C_{i+1}, \theta_i, C_i, \ldots, \theta_R, C_R)} \\
&= \frac{\pi_{post,\beta_i}(\theta_i, C_i) \pi_{post,\beta_{i+1}}(\theta_{i+1}, C_{i+1})}{\pi_{post,\beta_i}(\theta_{i+1}, C_{i+1}) \pi_{post,\beta_{i+1}}(\theta_i, C_i)} \\
&= \exp\{-(\beta_i - \beta_{i+1})[E(\theta_i, C_i) - E(\theta_{i+1}, C_{i+1})]\}
\end{aligned}$$

Again we write $T_{i,i+1}$ as a product of a proposal transition kernel and an acceptance probability. Here we have already explicitly chosen a deterministic proposal (we only propose transitions between replicas at neighboring inverse temperatures), which gives us

$$\begin{aligned}
&\frac{A_{i,i+1}(\theta_i, C_i, \theta_{i+1}, C_{i+1} | \theta_{i+1}, C_{i+1}, \theta_i, C_i)}{A_{i,i+1}(\theta_{i+1}, C_{i+1}, \theta_i, C_i | \theta_i, C_i, \theta_{i+1}, C_{i+1})} \\
&\qquad = \exp\{-(\beta_i - \beta_{i+1})[E(\theta_i, C_i) - E(\theta_{i+1}, C_{i+1})]\}
\end{aligned}$$

It is therefore enough to take

$$\begin{aligned}
A_{i,i+1}(\theta_i, C_i, \theta_{i+1}, C_{i+1} | \theta_{i+1}, C_{i+1}, \theta_i, C_i) = \min(1, \exp\{-(\beta_i - \beta_{i+1})[E(\theta_i, C_i) \\
- E(\theta_{i+1}, C_{i+1})]\}) \qquad (A28)
\end{aligned}$$

The reader sees that if the state of the "hot" replica (β_{i+1}) has a lower energy $[E(\theta_i, C_i)]$ than the "cold" one, the proposed exchange is always accepted. The exchange can pictorially be seen as cooling down the hot replica and warming up the cold one. Fundamentally this process amounts to making the replica that is at the beginning *and* at the end of the replica exchange transition at the cold temperature to jump from one local minimum (θ_{i+1}, C_{i+1}) to another one (θ_i, C_i). That is precisely what we were looking for. The fact that we can as well accept unfavorable exchanges (i.e., raising the energy of the

"cold" replica and decreasing the one of the "hot" replica) is the price we have to pay for our algorithm to generate samples from the proper posterior (we are not doing optimization here).

For the replica exchange transition to work well we need to be careful with our choice of inverse temperatures. The typical energy of a replica (i.e., its expected energy) increases when β decreases (Fig. 6*A*). We will therefore typically have a positive energy difference: $\Delta E = E_{hot} - E_{cold} > 0$ between the replicas at low and high β before the exchange. That implies that the acceptance ratio (*Eq. A28*) for the replica exchange will be typically smaller than 1. Obviously, if it becomes too small, exchanges will practically never be accepted. To avoid this situation we need to choose our inverse temperatures such that the typical product: $\Delta\beta\Delta E$, where $\Delta\beta = \beta_{cold} - \beta_{hot}$, is close enough to zero (Hukushima and Nemoto 1996; Iba 2001; Neal 1994). In practice we used preruns with an a priori too large number of βs, checked the resulting energy histograms and kept enough inverse temperatures to have some overlap between successive histograms (Fig. 6*B*).

The replica exchange transitions were performed between each pair β_i, β_{i+1} with an even, respectively odd, i at the end of each even, respectively odd, MC time step. With the replica exchange scheme, each MC time step was therefore composed of a complete parameter and label transition for each replica, followed by a replica exchange transition. This scheme corresponds to the one described by Hukushima and Nemoto (1996). A rather detailed account of the REM can be found in Mitsutake et al. (2001). Variations on this scheme do exist (Celeux et al. 2000; Neal 1994).

Random number generators for the simulated data

A vector $\mathbf{n}$ from an isotropic t density with $\nu(>2)$ degrees of freedom and a SD equal to 1 (for each component) was generated with the following algorithm (Fishman 1996; Gelman et al. 1995):

1) Generate: $\mathbf{z} \sim \text{Norm}(\mathbf{0}, 1)$
2) Generate: $x \sim \chi^2(\nu)$
3) Take: $\mathbf{n} = \sqrt{(\nu/(\nu-2)x}\,\mathbf{z}$

The Normal and χ^2 (pseudo-)random number generators are built-in features of Scilab (and MATLAB).

An *isi* from a log-Normal density with scale parameter s and shape parameter σ was generated with the following algorithm:

1) Generate: $z \sim \text{Norm}(0, \sigma^2)$
2) Take: $isi = s\exp(z)$

For the neuron with a Gamma *ISI* density, the Gamma random number generator of Scilab was used.

ACKNOWLEDGMENTS

We thank C. van Vreeswijk, N. Brunel, O. Mazor, L. Forti, A. Marty, J. Chavas, and M.-G. Lucas for comments and suggestions on the manuscript.

GRANTS

This work was supported in part by a grant from the Ministère de la Recherche (ACI Neurosciences Intégratives et Computationnelles, pré-projet, 2001–2003), by a grant from inter–Établissements Publiques Scientifiques et Techniques (Bioinformatique), and by the région Ile-de-France (programme Sésame). M. Delescluse was supported by a fellowship from the Ministère de l'Education National et de la Recherche.

REFERENCES

Ball FG, Cai Y, Kadane JB, and O'Hagan A. Bayesian inference for ion-channel gating mechanisms directly from single-channel recordings, using Markov chain Monte Carlo. *Proc R Soc Lond A Phys Sci* 455: 2879–2932, 1999.

Besag J. Markov Chain Monte Carlo for Statistical Inference [online]. Working Paper No. 9. Seattle, WA: Center for Statistics and Social Sciences, Univ. of Washington, 2001. http://www.csss.washington.edu/Papers/wp9.pdf [28 Nov. 2003].

Binder K and Young AP. Spin glasses: experimental facts, theoretical concepts and open questions. *Rev Mod Phys* 58: 801–976, 1986.

Brémaud P. *Markov Chains: Gibbs Fields, Monte Carlo Simulations and Queues.* New York: Springer-Verlag, 1998.

Celeux G, Hurn M, and Robert C. Computational and inferential difficulties with mixture posterior distributions. *J Am Stat Assoc* 95: 957–970, 2000.

Falcioni M and Deem MW. A biased Monte Carlo scheme for zeolite structure solution. *J Chem Phys* 110: 1754–1766, 1999.

Fee MS, Mitra PP, and Kleinfeld D. Automatic sorting of multiple-unit neuronal signals in the presence of anisotropic and non-Gaussian variability. *J Neurosci Methods* 69: 175–188, 1996a.

Fee MS, Mitra PP, and Kleinfeld D. Variability of extracellular spike waveforms of cortical neurons. *J Neurophysiol* 76: 3823–3833, 1996b.

Fishman GS. *Monte Carlo, Concepts, Algorithms and Applications.* New York: Springer-Verlag, 1996.

Frenkel D and Smit B. *Understanding Molecular Simulation. From Algorithms to Applications.* San Diego, CA: Academic Press, 2002.

Gelman A, Carlin JB, Stern HS, and Rubin DB. *Bayesian Data Analysis.* Boca Raton, FL: Chapman & Hall/CRC, 1995.

Gelman A and Meng XL. Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Stat Sci* 13: 163–185, 1998.

Geman S and Geman D. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans Pattern Anal Mach Intell* 6: 721–741, 1984.

Gray CM, Maldonado PE, Wilson M, and McNaughton B. Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. *J Neurosci Methods* 63: 43–54, 1995.

Green PJ. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* 82: 711–732, 1995.

Hansmann UHE. Parallel tempering algorithm for conformational studies of biological molecules. *Chem Phys Lett* 281: 140–150, 1997.

Harris KD, Henze DA, Csicsvari J, Hirase H, and Buzsaki G. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *J Neurophysiol* 84: 401–414, 2000.

Hastings WK. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57: 92–109, 1970.

Hukushima K and Nemoto K. Exchange Monte Carlo and application to spin glass simulations. *J Phys Soc Jpn* 65: 1604–1608, 1996.

Iba Y. Extended ensemble Monte Carlo. *Int J Mod Phys C* 12: 623–656, 2001.

Janke W. Statistical Analysis of Simulations: Data Correlations and Error Estimation [online]. 2002. http://www.fz-juelich.de/nic-series/volume10 [28 Nov. 2003].

Johnson DH. Point process models of single-neuron discharges. *J Comput Neurosci* 3: 275–299, 1996.

Johnson DH, Tsuchitani C, Linebarger DA, and Johnson MJ. Application of a point process model to responses of cat lateral superior olive units to ipsilateral tones. *Hearing Res* 21: 135–159, 1986.

Landau D and Binder K. *A Guide to Monte Carlo Simulations in Statistical Physics.* Cambridge, UK: Cambridge Univ. Press, 2000.

Lewicki MS. Bayesian modeling and classification of neural signals. *Neural Comput* 6: 1005–1030, 1994.

Lewicki MS. A review of methods for spike-sorting: the detection and classification of neural action potentials. *Network Comput Neural Syst* 9: R53–R78, 1998.

Liu JS. *Monte Carlo Strategies in Scientific Computing.* New York: Springer-Verlag, 2001.

Matsumoto M and Nishimura T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans Model Comput Simul* 8: 3–30, 1998.

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, and Teller E. Equations of state calculations by fast computing machines. *J Chem Phys* 21: 1087–1092, 1953.

Mitsutake A, Sugita Y, and Okamoto Y. Generalized-ensemble algorithms for molecular simulations of biopolymers. *Biopolymers* 60: 96–123, 2001.

National Institute of Standards and Technology (NIST). NIST/SEMATECH e-Handbook of Statistical Methods [online]. http://www.itl.nist.gov/div898/handbook/index.htm [28 Nov. 2003].

Neal RM. Probabilistic Inference Using Markov Chain Monte Carlo Methods [online]. Technical Report CRG-TR-93-1. Toronto, Canada: Department of Computer Science, Univ. of Toronto, 1993. http://www.cs.toronto.edu/~radford/papers-online.html [28 Nov. 2003].

Neal RM. Sampling from Multimodal Distributions Using Tempered Transitions [online]. Technical Report No. 9421. Toronto, Canada: Department of Statistics, Univ. of Toronto, 1994. http://www.cs.toronto.edu/~radford/papers-online.html [28 Nov. 2003].

Newman MEJ and Barkema GT. *Monte Carlo Methods in Statistical Physics*. Oxford, UK: Oxford Univ. Press, 1999.

Nguyen DP, Frank LM, and Brown EN. An application of reversible-jump Markov chain Monte Carlo to spike classification of multi-unit extracellular recordings. *Network Comput Neural Syst* 14: 61–82, 2003.

Pouzat C, Mazor O, and Laurent G. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. *J Neurosci Methods* 122: 43–57, 2002.

Quirk MC and Wilson MA. Interaction between spike waveform classification and temporal sequence detection. *J Neurosci Methods* 94: 41–52, 1999.

Robert CP and Casella G. *Monte Carlo Statistical Methods.* New York: Springer-Verlag, 1999.

Rosales R, Stark A, Fitzgerald WJ, and Hladky SB. Bayesian restoration of ion channel records using hidden Markov models. *Biophys J* 80: 1088–1103, 2001.

Sahani M. *Latent Variable Models for Neural Data Analysis* (PhD thesis). Pasadena, CA: California Institute of Technology, 2002.

Sokal AD. Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms. Cours de Troisième Cycle de la Physique en Suisse Romande [online]. http://citeseer.nj.nec.com/soka196monte.html [28 Nov. 2003].

Wu FY. The Potts model. *Rev Mod Phys* 54: 235–268, 1982.

Yan Q and de Pablo JJ. Hyper-parallel tempering Monte Carlo: application to the Lennard-Jones fluid and the restricted primitive model. *J Chem Phys* 111: 9509–9516, 1999.

5.2 Deuxième article : validation expérimentale de l'approche MCMC

Le second article entend démontrer la flexibilité de l'approche MCMC présentée dans le premier et montrer ses performances particulières sur des données réelles que les méthodes existantes traitent avec difficulté.

Les données sont celles de cellules de Purkinje (PCs) en présence de DHPG (un agoniste des récepteurs glutamate métabotropiques) dans la solution extracellulaire. Les activités de ces cellules sont enregistrées avec une électrode multisite. Le DHPG favorise la décharge spontanée en *bursts* des PCs. Ce type de données ne peut être analysé par les méthodes de *spike-sorting* existantes, du fait de la forte décroissance des PAs au cours de ces *bursts*. Dans cet article, nous procédons en deux temps :

1. Nous montrons qu'un modèle de Markov caché (*Hidden Markov Model*, HMM) à trois états s'ajuste bien aux décharges individuelles de PCs en présence de DHPG.
2. Nous introduisons ce modèle dans l'algorithme MCMC présenté dans le premier article, en lieu et place du modèle de statistique de décharge qui y est décrit. Afin de tester notre algorithme modifié sur ces données et d'évaluer quantitativement ses performances, une électrode de *patch* placée sur le soma de l'une des PCs vues par l'électrode multisite enregistre de façon non ambiguë le train de PAs de cette cellule. Nous montrons que l'un des trains de PAs reconstitués par notre algorithme est à 98% identique à ce train de référence, et ce, en dépit de la forte décroissance des amplitudes des PAs au cours des *bursts* de cette cellule. Les performances de l'algorithme d'*Expectation-Maximization* sont inférieures.

Dans cet article, nous montrons donc qu'équipée d'un modèle de génération de données pertinent, notre méthode peut s'adapter à des données qui posent problème aux méthodes actuelles. Nous montrons en particulier qu'un modèle de Markov caché rend compte de façon satisfaisante des décharges de neurones "multi-état", oscillant entre plusieurs types de décharge (quiescence, haute fréquence, fréquence intermédiaire). L'utilisation du DHPG est un moyen efficace de systématiser ce type de décharge dans les PCs, afin de donner plus de poids à la démonstration.

Par ailleurs, lors de sa soumission, cet article était accompagné d'un *compendium* R regroupant les données, les codes utilisés spécifiquement pour cet article ainsi qu'une *vignette R* permettant aux chercheurs évaluant ce travail de reproduire eux-mêmes entièrement cette analyse. Ce compendium est disponible sur le site *web* du laboratoire[1]. Le quatrième article, tout en exposant les résultats obtenus sur les enregistrements multiples de PCs, présente le principe, fonctionnement et l'intérêt d'une vignette et d'un compendium.

[1] `http ://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Compendium.html`

ELSEVIER

Journal of Neuroscience Methods xxx (2005) xxx–xxx

JOURNAL OF NEUROSCIENCE METHODS

www.elsevier.com/locate/jneumeth

Efficient spike-sorting of multi-state neurons using inter-spike intervals information

Matthieu Delescluse, Christophe Pouzat

Laboratoire de Physiologie Cérébrale, CNRS UMR 8118, UFR Biomédicale de l'Université René Descartes, 45 rue des Saints-Pères, 75006 Paris, France

Received 6 April 2005; received in revised form 17 May 2005; accepted 22 May 2005

Abstract

We demonstrate the efficacy of a new spike-sorting method based on a Markov chain Monte Carlo (MCMC) algorithm by applying it to real data recorded from Purkinje cells (PCs) in young rat cerebellar slices. This algorithm is unique in its capability to estimate and make use of the firing statistics as well as the spike amplitude dynamics of the recorded neurons. PCs exhibit multiple discharge states, giving rise to multi-modal inter-spike interval (ISI) histograms and to correlations between successive ISIs. The amplitude of the spikes generated by a PC in an "active" state decreases, a feature typical of many neurons from both vertebrates and invertebrates. These two features constitute a major and recurrent problem for all the presently available spike-sorting methods. We first show that a hidden Markov model with three log-normal states provides a flexible and satisfying description of the complex firing of single PCs. We then incorporate this model into our previous MCMC based spike-sorting algorithm [Pouzat C, Delescluse M, Viot P, Diebolt J. Improved spike-sorting by modeling firing statistics and burst-dependent spike amplitude attenuation: a Markov chain Monte Carlo approach. J Neurophysiol 2004;91:2910–28] and test this new algorithm on multi-unit recordings of bursting PCs. We show that our method successfully classifies the bursty spike trains fired by PCs by using an independent single unit recording from a patch-clamp pipette.

Keywords: Markov chain Monte Carlo; Multi-electrode; Hidden Markov model; Purkinje cell

1. Introduction

Multi-site extracellular recordings are extensively used by laboratories that aim at studying neuronal populations activity and a variety of recently developed technologies enable the experimentalist to do so in many preparations: cultures (Gross et al., 1993, 1997), slices (Egert et al., 2002; Oka et al., 1999), in vivo (Baker et al., 1999; Csicsavari et al., 2003; Drake et al., 1988; Nicolelis et al., 1997). But in order to be really informative and fully exploitable, such recordings require the difficult spike-sorting problem to be solved: the resolution of a mixture of activities into well separated individual spike trains. This problem has an already long history (Lewicki, 1998), but has not yet received any fully satisfying solution (Brown et al., 2004; Buzsaki, 2004). In particular, until recently (Pouzat, 2005; Pouzat et al., 2004), all of the available methods made exclusively use of the information provided by the waveform of individual spikes,[1] ignoring that of their occurrence times. Many neurons have however fairly reproducible firing features that can often be summarized by their inter-spike interval (ISI) probability density. This temporal information can greatly improve classification performance and allows the investigator to take into account the dependence of the spike amplitude upon the ISI, like for instance during a burst (where a spike amplitude reduction is typically observed on an extracellular electrode as well as on an intracellular one).

Abbreviations: DHPG, (*S*)-3,5-dihydroxyphenylglycine; HMM, hidden Markov model; ISI, inter-spike interval; MC, Monte Carlo; MCMC, Markov chain Monte Carlo; mGluR1, group 1 metabotropic glutamate receptor; PC, Purkinje cell; REM, replica exchange method; S.D., standard deviation

Corresponding author. Tel.: +33 1 42 86 38 28; fax: +33 1 42 86 38 30.
E-mail address: christophe.pouzat@univ-paris5.fr (C. Pouzat).

[1] Except Fee et al. (1996a) where the presence of a refractory period was used in an ad hoc way.

0165-0270/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.jneumeth.2005.05.023

In that context, we recently proposed a new Bayesian method based on a Markov chain Monte Carlo (MCMC) approach (Pouzat, 2005; Pouzat et al., 2004). This method is built on a data generation model that includes both a description of non-Poisson neuronal discharge statistics and a description of spike waveform dynamics. In these papers, we chose a *single* log-normal density to model individual ISIs distributions. It is nevertheless clear that this is not the only model that can be considered: the MCMC framework allows the experimentalist to use the model that is best supported by the data from the neuronal type he is studying. In particular, when one is dealing with neurons exhibiting several states resulting in bursty discharges the unimodal log-normal density is not appropriate anymore, as observed for example in thalamic relay cells (McCormick, 1998) and in cerebellar Purkinje cells (PCs) (Loewenstein et al., 2005).

The primary goal of the present paper is to show how our spike-sorting method, modified to take into account such multi-state neurons, performs on real data that would make any other automatic method fail. We chose a challenging data set recorded from cells firing bursts of spikes. Such data could be obtained in young rat cerebellar slices by applying a multi-site electrode along the PCs layer in the presence of the group I metabotropic glutamate receptor (mGluR1) agonist (*S*)-3,5-dihydroxyphenylglycine (DHPG). In these pharmacological conditions, PCs fire bursts of two or more action potentials (APs) with dramatically decreasing amplitudes (e.g. 50%), in alternance with long periods of silence (on the order of 1 min) (Netzeband et al., 1997).

2. Methods

2.1. Experimental procedure

Slices preparation and loose cell-attached patch-clamp recording were done as previously described (Pouzat and Hestrin, 1997). Sagittal slices (180 μm thick) were taken from the vermis of cerebella from rats aged 11–14 days. Single and multiple unit(s) recordings were made from PCs in these slices visualized through a 63× objective in an upright microscope equipped with Nomarski optics (Axioscope, Carl Zeiss, Germany). These conditions allowed easy resolution of the various layers and cell types within the cerebellar cortex. During recording, the slices were maintained at room temperature (20–22 °C). They were continuously perfused with BBS which contained (mM): 130 NaCl, 2.5 KCl, 2 $CaCl_2$, 1 $MgCl_2$, 1.3 NaH_2PO_4, 26 $NaHCO_3$ and 25 glucose. This solution was continuously bubbled with a mixture of 95% O_2 and 5% CO_2, the solution pH being thus kept at 7.4.

The mGluR1 agonist DHPG (40 μM) was applied dissolved in the bathing solution. All the recordings analyzed in this paper were made in these conditions. DHPG was purchased from Tocris Neuramin Ltd. (Bristol, UK). It was dissolved in distilled water at a concentration of 5 mM. The DHPG stock solution was stored frozen, and the final concentration was obtained by diluting the stock solution in the saline, just before its use in the experiment. Note that spontaneous bursting also occurred without DHPG at this age, but less systematically.

Single unit recordings were performed in loose cell-attached using a glass micropipette filled with the following solution (mM): 145 NaCl, 2.5 KCl, 2 $CaCl_2$, 1 $MgCl_2$, 10 HEPES acid. Pipette resistance ranged from 2 to 4 MΩ. The pipette was positioned in loose cell-attached on the soma of a PC. It was connected to a patch-clamp amplifier (Axoclamp 2B, Axon Instruments Inc., USA). This amplifier was connected to one of the 8 channels of two 4-channel differential ac amplifiers (AM systems, model 1700, Carlsborg, WA), also used for multi-unit data recordings (see below). The signal was band-pass filtered between 300 and 5000 Hz and amplified 1000 times. Such single unit recordings were performed in two types of experiments. First, they were made alone in order to gather several examples of individual PCs spike trains during spontaneous activity. Second, they were made together with multiple units recordings where they served as a reference recording to which the spike-sorting output was compared.

Multi-unit recordings were performed using silicon probes (also called "multi-site electrodes" in the sequel) kindly provided by the Center for Neural Communication Technology of the University of Michigan. A schematic drawing of the tip (first four recording sites) of the probe is shown in Fig. 4. The 16 recording sites are linearly placed on the electrode 50 μm apart. This electrode was positioned along the PCs layer. The spontaneous spiking activities of these PCs could routinely be recorded on the first eight sites of the electrode, with an excellent signal-to-noise ratio (see Fig. 4B1, B2, C1 and C2 for some examples). The analysis detailed in the present paper was made on the first four recording sites. A glass micropipette was positioned in loose cell-attached on the soma of one of the PCs whose spontaneous activity was recorded by the multi-site electrode.

The multi-site electrode was connected to a custom-made impedance matching preamplifier. The preamplifier was connected to the two 4-channel differential ac amplifiers mentioned above. The signals were bandpass filtered between 300 and 5000 Hz and amplified 2000 times. All data were acquired at 15 kHz using a 16 bit A/D card (PD2MF-64-500/16H, United Electronics Industries, Watertown, MA) and stored on disk for subsequent analysis.

2.2. Data analysis

2.2.1. Events detection and representation

Multi-unit data. Data recorded on the first four recording sites of the multi-site electrode were analyzed. A first set of large events were detected as local maxima with a peak value exceeding a preset high threshold (five times the standard deviation (S.D.) of the whole trace), and normalized (peak amplitude, at 1, temporal average, at 0) to give a "spike template". Each trace was then filtered with this template

(by convolution with the template in reversed time order). Events were detected on the filtered trace as local maxima whose peak value exceeded a preset threshold (a multiple of the S.D. of the filtered trace). After detection, each event was described by its occurrence time and its peak amplitude measured on four recording sites. To simplify calculations and reduce the complexity of our algorithm, the peak amplitude(s) were "noise whitened" as described in Pouzat et al. (2002) (see also *SpikeOMaticTutorial 1*[2]). A spike detected on a given recording site can be seen on its immediate neighbouring sites (50 μm apart) with reduced amplitudes, but never on further sites. This is consistent with an exponential decay of the signal with decay constant 30 μm (Gray et al., 1995; Segev et al., 2004).

Single unit data. For data recorded by the glass micropipette, events were detected the same way. Each event was described by its occurrence time and its peak amplitude. We normalized the peak amplitudes of the spikes by the S.D. of 2000 noise "peak" amplitudes taken in the same recording. When single unit data were recorded together with multi-unit data, events detected on the micropipette trace were described by their occurrence times only.

2.2.2. Data generation model for statistical inference

To perform spike-sorting our algorithm makes statistical inference on the parameters of the data generation model described in this section. This model is based on the following assumptions:

1. The sequence of spike times from a given neuron is a realization of a hidden Markov point process (Camproux et al., 1996; Guclu and Bolanowski, 2004).
2. The spike amplitudes generated by a neuron depend on the elapsed time since the previous spike of this neuron.
3. The measured spike amplitudes are corrupted by a Gaussian white noise which sums linearly with the spikes and is statistically independent of them.

Assumptions 2 and 3 are identical to those made in Pouzat et al. (2004) and Pouzat (2005). Assumption 3 requires a prior noise whitening of the data (Pouzat et al., 2002). In assumption 1, the homogeneous renewal point process assumption that was made in Pouzat et al. (2004) and Pouzat (2005) is changed into the more complex one of a hidden Markov point process (see below and in Appendix A.1.1). Our data generation model can be divided in two parts that respectively rely on assumptions 1 and 2 which are presented next.

2.2.2.1. Inter-spike interval density. We resort to a hidden Markov model (HMM) with three states to account for the empirical ISI density of the recorded cells. In this HMM context, we can see a sequence of ISIs (a spike train) produced by a given neuron as the observable output of a "hidden" sequence of "states" of this neuron (this denomination arises from the fact that the state in which the neuron is, is not directly observable from the data). The probability density from which each ISI is drawn depends on the underlying state. In our particular implementation, the ISI density of each state is a log-normal density characterized by two parameters: a scale parameter s (in s) and a shape parameter σ (dimensionless). With this notation, the general formula for the probability density function of the log-normal distribution is

$$f(\mathrm{isi}) = \frac{1}{\mathrm{isi}\,\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\log\left(\frac{\mathrm{isi}}{s}\right)}{\sigma}\right)^2\right) \quad (1)$$

The log-normal density is a relevant alternative to the exponential density usually used to model spike trains. It is unimodal, exhibits a refractory period, rises fast and decays slowly.

After the generation of each event, a "transition" to any of the three possible states is performed stochastically. In addition to the six parameters for the three log-normal densities mentioned above, we have therefore to consider the transition matrix (q_{ij}) between these states, which contains another six parameters. We thus have 12 parameters to specify the ISI density for each neuron.

A scheme summarizing this model is shown in Fig. 1A. In a spike train, each event (ISI) is generated by one of the three possible probability densities according to the state in which the neuron is: if the neuron is in state 1 (resp. 2, 3) it generates a short (resp. intermediate, long) ISI from the red (resp. blue, green) density. The transition from a given state to any other, including itself, is possible, as indicated by the different arrows between states. In the sequel we will constantly refer to the same color code for the states of single unit data: state 1 in red, state 2 in blue, state 3 in green. They will be also called "short", "intermediate" and "long" states, respectively. A more formal presentation of the HMM is to be found in Appendix A.1.1.

2.2.2.2. Spike amplitude dynamics. We use here the same spike amplitude dynamics as in Pouzat et al. (2004) and Pouzat (2005). We consider events described by their occurrence time and their peak amplitude measured on one (single site recordings) or four recording sites (multi-site recordings). We model the dependence of the amplitude on the ISI by an exponential relaxation (Fee et al., 1996b):

$$A(\mathrm{isi}) = P(1 - \delta\exp(-\lambda\mathrm{isi})) \quad (2)$$

where isi is the ISI, λ the inverse of the relaxation time constant (measured in s^{-1}), P the vector of the maximal amplitude of the event on each recording site (this is a four-dimension, resp. one-dimension, vector for multi-site, resp. single site, recordings) and δ is the dimensionless maximal modulation. This model implies that the modulation of the amplitudes of an event is the same on the different recording sites. This is an important feature of the amplitude modulation observed experimentally (Gray et al., 1995). Added to

[2] http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/SOM.html.

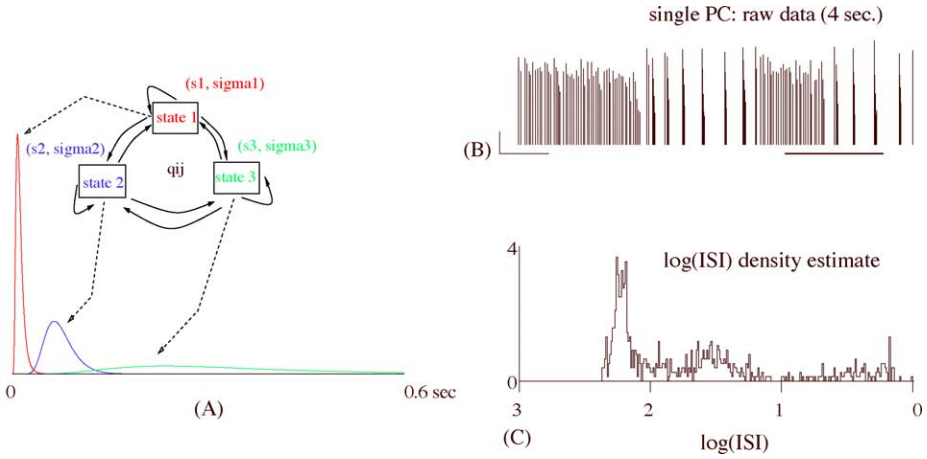

Fig. 1. Model for the ISI density compared to a real spike train. (A) ISI density model: a hidden Markov model with three states. Every state is a log-normal density with two parameters: a scale parameter s (in s) and a shape parameter (dimensionless) σ: (0.01 s, 0.5) state 1 (0.07 s, 0.3) state 2 and (0.3 s, 0.5) state 3. (B) spontaneous activity of a single PC in presence of bath-applied DHPG (40 μM) in loose cell-attached. Normalized peak amplitudes of the detected events are shown (duration: 4 s). The thick horizontal bar on the right indicates the part of the train shown in Fig. 2B. Horizontal scale bar: 0.5 s. Vertical scale bar: 5 (in units of noise S.D.). (C) $\log_{10}$ (isi) histogram of the same spike train as in (A) (1 min, 763 spikes). Bin width: 0.01.

the 12 parameters used for the ISIs, the number of parameters per neuron in our model amounts to 18 for multi-site recordings and 15 for single site recordings.

2.2.3. The Markov chain Monte Carlo approach

We have shown in our previous papers that the spike-sorting problem with a data generation model similar to the one presented here can be viewed as a one dimensional Potts spin-glass in a random magnetic field (Pouzat, 2005; Pouzat et al., 2004). This analogy allowed us to tailor the dynamic Monte Carlo algorithms developed by physicists (Frenkel and Smit, 2002; Newman and Barkema, 1999) and the Markov chain Monte Carlo (MCMC) developed by statisticians (Robert and Casella, 1999) for analogous problems to our particular needs. In essence the statistical inference in our case relies on the construction of a Markov chain (not to be mistaken for the HMM which is modeling the ISI density) whose space S is the product of two spaces: the space of the model parameters defined in our data generation model (and presented in the previous section) and the space of spike train configurations, where a configuration is defined by specifying a neuron of origin (a "label") and a neuron state for each spike. The latter, neuron state, is the new model ingredient introduced in the present paper (Section 2.2.2). Thus, a state of our Markov chain in this space is determined by two vectors: vector θ of model parameters (a $18K$, resp. $15K$, dimensional vector for multi-site, resp. single site, recordings, where K is the number of neurons, see Section 2.2.2) and vector C of the configuration, specifying a label and a neuron state for each spike (a $2N$-dimensional vector, where N is the number of detected spikes being analyzed). The construction of this Markov chain is done in such a way that it samples our space S from the posterior density of the model parameters and configurations given the data Y, noted $\pi_{\text{post}}(\theta, C|Y)$: at each step t of the algorithm a new state $[\theta^{(t)}$,

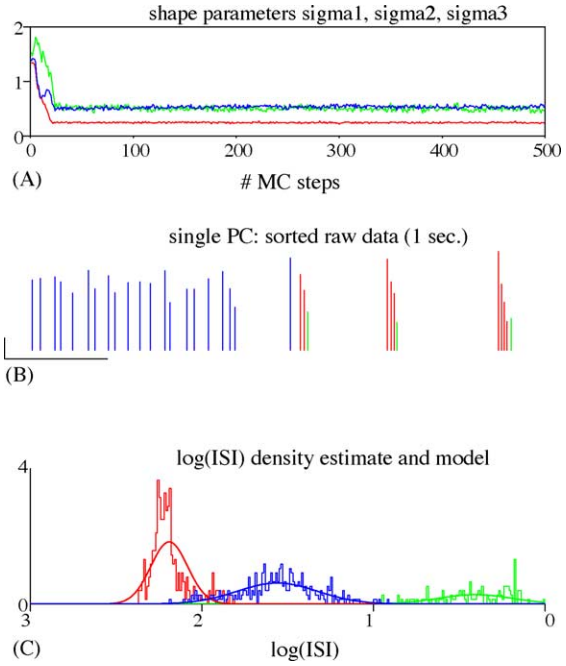

Fig. 2. Sorting ISI modes in a cell engaged in complex bursting behavior. MCMC output after a 1000-MC step run on the spike train shown in Fig. 1. (A) Evolution of the shape parameters $\sigma_1, \sigma_2, \sigma_3$ during this 1000-step run. Only the first 500 MC steps are shown (same color code as in Fig. 1). (B) Spike label analysis of the episode indicated by the thick horizontal bar of Fig. 1B. Each spike is colored according to its most probable HMM state determined by the algorithm (same color code as in Fig. 1 and part (A)). Horizontal scale bar: 0.2 s. Vertical scale bar: 5 (in units of noise S.D.). (C) $\log_{10}$ (isi) histogram of each state for the most probable configuration (same color code as in Fig. 1 and parts (A) and (B)). Note that each state corresponds to one mode of the histogram of Fig. 1C. The three model densities whose parameters have been set at their average values computed on the last 200 iterations are superimposed on the histogram.

$C^{(t)}$] of the Markov chain (not to be mistaken for the neuron states of the HMM used to model the ISI density) is generated from the state at step $t-1$, [$\theta^{(t-1)}$, $C^{(t-1)}$] according to the procedure described in Appendix A.1.2. The new components of the algorithm are the generation of the neuron states of the HMM (when generating the new configuration) and the generation of the transitions q_{ij} between these neuron states (which are components of the vector θ of parameters), using a Dirichlet distribution (Robert and Casella, 1999). This way of generating a new state from the previous one ensures that the Markov chain converges to a unique stationary distribution given by $\pi_{\text{post}}(\theta, C|Y)$ (Pouzat et al., 2004).

As described in our previous model, the "energy landscape" explored by our Markov chain exhibits some "glassy" features. It is therefore necessary, in general, to use the replica exchange method (REM, also known as parallel tempering method) (Hansmann, 1997; Hukushima and Nemoto, 1996; Iba, 2001) described in Pouzat et al. (2004) and Pouzat (2005). The method is not fully automatic yet and requires that the user chooses the number of active neurons in the data by individually scanning models with different numbers of neurons (Pouzat et al., 2004).

2.2.4. Output analysis

Once the simulated Markov chain has reached equilibrium, i.e. the chain is sampling from its stationary distribution which is our desired posterior density, we can estimate the values of the parameters and labels, as well as errors on these estimates. This is done by averaging the value of a given parameter θ_i over the N_T algorithm steps performed, after discarding the first N_D steps (burn-in) necessary to reach equilibrium:

$$\bar{\theta}_i = \frac{1}{N_T - N_D} \sum_{t=N_D}^{N_T} \theta_i^{(t)} \tag{3}$$

However, the successive Markov chain states generated by our algorithm are correlated, that is, the values of a given parameter θ_i at successive steps are not independent. Therefore, the standard deviation (S.D.) of this parameter must be corrected for this autocorrelation (Janke, 2002; Sokal, 1989). As explained in detail by Janke (2002) the correction is made by multiplying the empirical variance $\sigma^2(\theta_i)$ of this parameter by the *integrated autocorrelation time* $\tau_{\text{autoco}}(\theta_i)$:

$$\tau_{\text{autoco}}(\theta_i) = \frac{1}{2} + \sum_{l=1}^{L} \frac{\rho(l;\theta)}{\rho(0;\theta)} \tag{4}$$

where L is the lag at which ρ starts oscillating around 0 and ρ is the autocorrelation function of θ_i:

$$\rho(l;\theta_i) = \frac{1}{N_T - N_D - 1} \sum_{t=N_D}^{N_T} \left(\theta_i^{(t)} - \bar{\theta}_i\right)\left(\theta_i^{(t+L)} - \bar{\theta}_i\right) \tag{5}$$

We then have

$$\text{Var}(\theta_i) = 2\tau_{\text{autoco}}\sigma^2(\theta_i) \tag{6}$$

2.2.5. Software availability

Our codes are freely available (under the Gnu Public Licence) and can be found together with tutorials on our web site (see Footnote 2). The data presented in this paper are also freely available,[3] as well as a compendium which enables the interested reader to reproduce the whole analysis detailed in Section 3.

3. Results

We proceed in two steps. We first detail and justify the data generation model we chose to account for PCs firing statistics. We performed loose cell-attached recordings of PCs in cerebellar slices in the presence of DHPG, and we show that a hidden Markov model (HMM) with three log-normal states fits reasonably well the ISI histograms of the individual spike trains obtained. For such single neuron data, our algorithm is based on the construction of a Markov chain on the space of the HMM parameters and single spike train configurations, where a configuration is defined by specifying one of the three states for each spike. A Monte Carlo (MC) simulation is then used to estimate the posterior density of the HMM parameters and of single spike train configurations.

The second step is the inclusion of this model of neuronal discharge into our general spike-sorting algorithm before running it on multi-unit recordings of bursting PCs. Besides our multi-site electrode positioned along the PCs layer, a glass micropipette independently caught the activity of one of these PCs in loose cell-attached. We can therefore show that our spike-sorting method reliably isolates the activity of this reference cell, although it is firing bursts of spikes with decreasing amplitudes and exhibits a muti-modal ISI histogram.

3.1. Single unit recordings of Purkinje cells in loose cell-attached

We performed single unit recordings of spontaneously active PCs ($n = 12$) using a glass micropipette (2–4 MΩ) in loose cell-attached in the presence of bath-applied DHPG (40 μM). In these conditions, PCs systematically fire bursts of variable lengths, in alternance with long periods of silence (on the order of 1 min). After detection of the events, the inter-spike interval (ISI) histogram of each spike train was plotted. All of them were multi-modal. They had a principal mode corresponding to the most frequent ISI of the cell in normal condition (typically 60 ms), as well as a mode at longer ISIs (hundreds of ms) whose width was variable. The

[3] http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Compendium.html.

third mode at short ISIs (5–10 ms) corresponds to the ISIs which are found in these bursts of two or more spikes. In such bursts the amplitudes of the spikes are strongly reduced (see Section 3.3).

Fig. 1B shows 4 s of a typical spike train of a PC in DHPG. Note the presence of bursts of spikes of dramatically decreasing amplitudes. The ISI histogram of this train (763 spikes, 1 min of recording) is plotted in Fig. 1C. The multi-modal character of this histogram is unambiguous. This type of activity (usually on the order of 1 min) alternates with silent periods with a similar duration of 1 min.

3.2. A three-state hidden Markov model fits well empirical inter-spike interval densities

We used our MCMC algorithm without the REM (see Section 2) to fit our three-state HMM parameters, as well as the amplitude parameters (see Section 3.3, for the fit of amplitude parameters), from this single unit spike train. In this section, where no spike-sorting is performed, our algorithm only fits the model parameters of the single cell recorded and attributes one of the three HMM states to each spike in this single unit spike train. Fig. 2A shows the evolution of the dimensionless shape parameters σ_1 (red), σ_2 (blue), σ_3 (green) during a 1000-MC steps run. Only the first 500 MC steps are displayed, but there was absolutely no change in the evolution of these parameters between steps 500 and 1000. All other parameters (scale parameters s_1, s_2, s_3 and transition parameters q_{ij}) had similar evolutions. Note that the algorithm reaches equilibrium very fast (after about 20 MC steps). The average values of s_1, s_2, s_3 (autocorrelation corrected S.D.s given in parenthesis, see Section 2.2.4) computed on the last 200 iterations were 6 (0.08) ms, 28 (1) ms, 392 (22) ms, respectively. These scale values are to be compared to the location of the three ISI histogram's modes described in what follows. The average values of σ_1, σ_2, σ_3 (autocorrelation corrected S.D.s given in parenthesis, see Section 2.2.4) were 0.246 (0.01), 0.538 (0.026), 0.494 (0.041), respectively.

Fig. 2B shows part of the spike train of Fig. 1B. At each step the algorithm attributes one of the three possible states to each spike. The configuration (i.e. the labeling of each spike with a neuron state's number) shown in Fig. 2B is the most frequent one computed over the last 200 steps of the 1000-step run displayed in A. We use the same color code as in Fig. 1A. As expected, spikes in bursts are attributed by the algorithm to a short state (red) label, except for the last spike of the burst which is followed by a long ISI and is thus attributed to a long state (green) label. Spikes occurring during regular spiking and separated by intermediate ISIs are attributed the intermediate state (blue) label.

In Fig. 2C, the ISI histogram of this spike train has been subdivided and colored according to the state of the neuron: all ISIs generated when the neuron was in the short state (resp. intermediate, long) are plotted in red (resp. blue, green), as expected. Superimposed on this histogram are the three model ISI densities whose parameters have been set to their

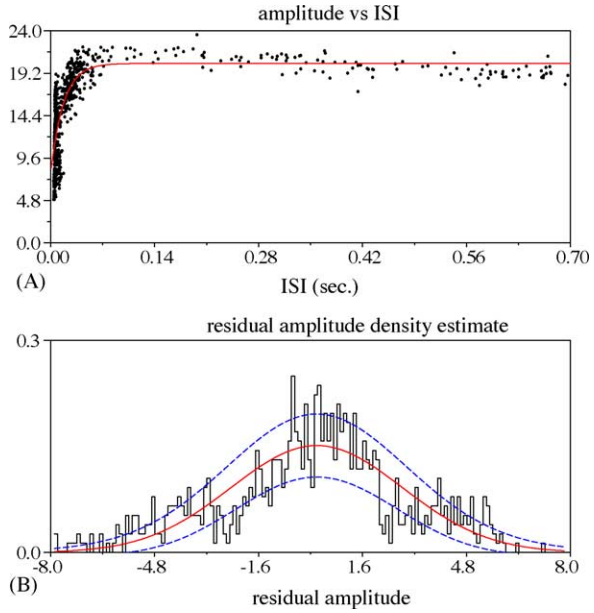

Fig. 3. Model for the spike amplitude dynamics. (A) Events normalized peak amplitudes plotted with respect to the ISI preceding the event (dots). The fitted exponential relaxation is superimposed (solid line). Ordinates in units of noise S.D. (B) Histogram of the residual amplitudes. The Gaussian fit is superimposed (solid line). The two dashed lines are 1S.D. away from the expected histogram.

average values computed on the last 200 MC steps and given above. The reader sees that the initial ISI histogram displayed in Fig. 1C is reasonably well fitted by this mixture of three log-normal densities (the most striking deviation between the actual data and the fit being observed for the short state). Moreover, besides its ability to describe a multi-modal ISI histogram, the HMM can also account for the dependence between successive ISIs through its transition matrix. In our case, a long state is always followed by a short state, as shown more quantitatively in Appendix A.2.1.

These results show that the model we propose can satisfyingly, but not perfectly, account for the discharge considered here. In Fig. 7, we provide the interested reader with a goodness-of-fit test based on Kolmogorov–Smirnov plots. It shows that this spike train clearly supports this three log-normal state HMM when compared to models with two and one state(s).

Between two periods of silence, a period of PC activity in DHPG always evolves from a tonic firing at about 15 Hz (second mode of the ISI histogram) to a bursty firing with 150 Hz-bursts (first mode of the ISI histogram)[4] separated by intervals of several hundreds of ms (third mode of the ISI histogram). This is well illustrated in Fig. 1B which represents a transition from this tonic to bursty firing. The minute of activity shown in Figs. 1–3 depicts perfectly this evolution. However, in the case where only the short bursts are recorded, only two modes are prevailing (the short and the long ones).

[4] We are therefore approximating a non-stationary discharge dynamics with a stationary one.

Such a case is obvious in the multi-unit data analyzed in Section 3.4.

3.3. Spike amplitudes relax exponentially with respect to inter-spike interval duration

The second part of our data generation model concerns the dependence of spike amplitude upon the time elapsed since the last spike of the same neuron. We also checked whether our PC spike trains supported this hypothesis. We use here the same data set as in Section 3.2. In Fig. 3A, the normalized peak amplitude of each detected spike is plotted against the ISI that preceded it. Recall that this single unit data were obtained with one recording site, so that only one peak amplitude per spike is to be considered. The exponential relaxation with parameter values determined by our algorithm is superimposed on the data points. The reader is referred to Section 2.2.2 where this exponential model is presented. For this particular train, parameter values are (S.D.s given in parenthesis): $P = 20.3$ (0.03), $\delta = 0.617$ (0.007), $1/\lambda = 17$ (0.4) ms. P is given in noise S.D. and δ is dimensionless.

Several issues now must be addressed with respect to the peak amplitude variance of the neurons we measured. First, the variability of spikes amplitudes at short ISIs (around 5 ms) seems to be larger than those at intermediate or long ISIs. This "over-variability" is mainly a visual effect for a narrow range of short ISIs is significantly more represented. This over-represented population necessarily samples the Gaussian distribution more thoroughly. Second, a group of points with an abscissa around 10 ms as well as points with abscissa greater than 400 ms are clearly below the exponential fit, while points with abscissa in 50–400 ms range are slightly above it. Third, these two significant deviations compensate each other. This is shown by the histogram of the residual amplitudes displayed in Fig. 3B to which a fitted Gaussian density with an S.D. equal to 2.64 is superimposed. The S.D. of the residual amplitude histogram has been both added to and subtracted from the fitting Gaussian curve (upper and lower dashed line, respectively). One sees that an exponential relaxation of amplitudes looks like a good first approximation of the actual amplitude dynamics. One sees as well that our third hypothesis is only an approximation, for our residual here exhibit a larger variability (an S.D. of 2.64) than the one expected from the measured background noise (S.D. of 1). In particular, these data exhibit a slight but clear decrease of amplitude with long ISIs, whereas our model keeps a fixed, maximal amplitude for these ISIs.

3.4. Multi-unit data sorted by our algorithm

3.4.1. Multi-unit data and reference neuron

We performed multi-unit recordings of spontaneously active PCs in the presence of DHPG using a multi-site electrode that we positioned along the PC layer. A glass micropipette was placed in loose cell-attached next to site 3 of the multi-site electrode in order to independently monitor the activity of one of the PCs from the recorded population. Such data were kept only if the glass micropipette unambiguously recorded the activity of a single cell with an excellent signal-to-noise ratio. This cell is called "reference cell" and the detected events of these recordings serve as "reference events" to which the output of our algorithm is compared. In what follows, we show the performance of our spike-sorting algorithm on a representative example of these recordings (58 s, 2739 events detected). Each detected event is described by its time of occurrence and the four peak amplitudes on the four recording sites after noise whitening.

3.4.2. The reference neuron is reliably labeled as unit 1

The following results were obtained after a 1000-MC steps with the REM and the following "inverse temperatures": $\beta = 1$, 0.975, 0.95, 0.925, 0.9, 0.875, 0.87 followed by 1000 steps with $\beta = 1$ only. This required about 33 min on a 3 GHz PC (Pentium IV) running Linux. Plots of energy evolution and parameters evolution showed that all parameters had reached their equilibrium value after roughly 500 MC steps (not shown). We computed the average value of each model parameter using the last 200 MC steps. We also forced the soft classification produced by our algorithm into the most likely classification using the last 200 MC steps (Pouzat et al., 2004).

Fig. 4A shows two separate periods of 2 s from these data (peak amplitudes of the detected events after noise whitening, see Section 2). Each row corresponds to one recording site[5] of the Michigan probe (sites 1 to 4 from top to bottom), as depicted on the left. Each event is colored according to its most probable neuron of origin: neuron 1 in black, neuron 2 in deep blue, neuron 3 in green, neuron 4 in light blue, neuron 5 in red, neuron 6 in brown. A raster plot of the reference events is displayed in the upper part of the site 3 panel. From these plots, it is obvious that the black unit (unit 1) reconstructed by our algorithm corresponds to the reference cell (see details below). Note that the event amplitudes of this cell are strongly reduced within the bursts so that they become similar to those produced by unit 5 (red). This of course makes the separation between these two units really difficult.

Raw data from site 3 are displayed in Fig. 4B2 (crosses are drawn on top of the detected events), showing one of the bursts of unit 1 as well as a "background" cell (for example, the first three detected events on this panel come from this background cell). As the spike amplitude decreases within the burst of unit 1, these spikes end up being of the same size as this background cell. The latter is labeled as unit 5 by our algorithm (red on panel A). The corresponding raw data recorded by the independent micropipette are displayed in Fig. 4B1. Note the huge signal amplitude, as well as the

[5] In fact to a mixture of all of them (noise whitening has been performed), but the contribution of one site is still predominant.

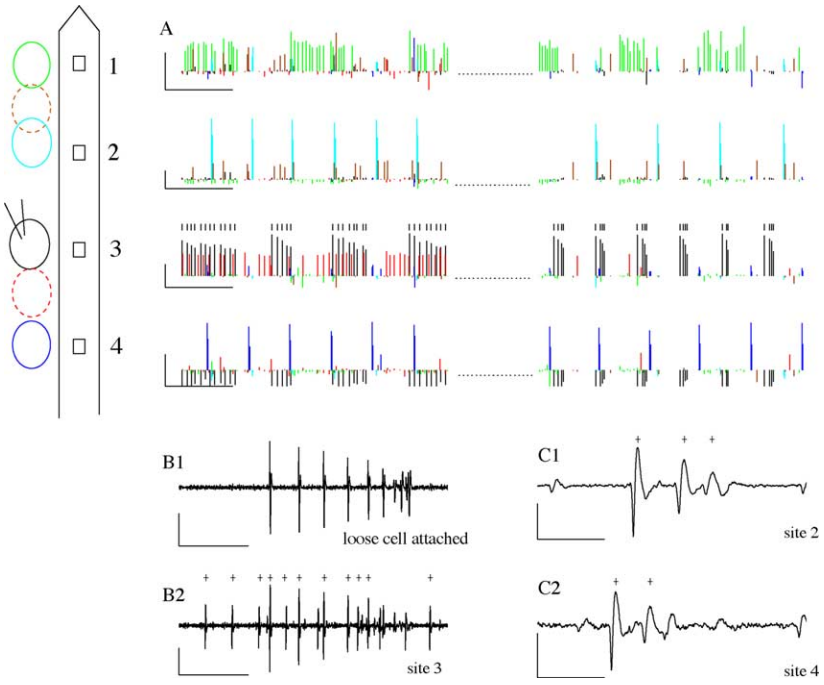

Fig. 4. Spike-sorting on a mixture of several PC spike trains simultaneously recorded (1). (A) spontaneous activity of PCs with bath-applied DHPG recorded on the first four sites of the electrode. Two separate periods of 2 s taken from the same minute of recording are shown (peak amplitudes of the detected events after noise whitening). Each event is colored according to the label determined by our algorithm. These labels are the most probable ones over the last 200 steps of a 1000-step run. An independent recording was performed next to site 3 by a patch-clamp pipette in loose cell attached. A raster plot of this reference neuron's activity is displayed in the upper part of the panel of site 3. A scheme of the first four sites of the electrode as well as the recorded PCs with the positioned extracellular glass pipette are drawn on the left. Horizontal scale bar: 0.5 s. Vertical scale bar: 10 (in units of noise S.D.). (B1) Raw data recorded by the extracellular glass pipette in loose cell attached, showing a typical burst. Horizontal scale bar: 100 ms. Vertical scale bar: 0.5 mV. (B2) Corresponding raw data recorded by site 3 of the multi-site electrode, showing the detected burst. Crosses are drawn on top of the detected events (the same holds for (C1) and (C2)). Horizontal scale bar: 100 ms. Vertical scale bar: 0.25 mV. (C1) Raw data recorded by site 2 of the multi-site electrode, showing a typical triplet. Horizontal scale bar: 10 ms. Vertical scale bar: 0.5 mV. (C2) Raw data recorded by site 4 of the multi-site electrode, showing a typical triplet (detected as doublet). Horizontal scale bar: 10 ms. Vertical scale bar: 0.25 mV.

decreasing spike amplitudes along the burst. The activity recorded by the micropipette unambiguously comes from a unique PC. These two panels B1 and B2 allow a direct comparison of the signal received by site 3 of the microelectrode to the one received by the pipette: the latter records the burst seen on panel B2 only, and not the background cell. They also illustrate the fact that not all events of the reference cell are detected on site 3: the very last spikes of each burst fired by the reference cell are much smaller and below our detection threshold. For that reason, among the 766 reference events detected on the micropipette trace during this minute of data, 641 are detected on the trace of site 3. Among these 641 events, 629 are attributed to unit 1 by our algorithm (98.1%). Overall, 637 events are attributed to unit 1 so that eight unit 1 events are not reference events (false positives, 1.3%). The 12 reference events not labeled as unit 1 are labeled as unit 5 (red). For comparison, a classical Gaussian mixture model (GMM) fitted with the Expectation-Maximization (EM) algorithm (Pouzat et al., 2002), attributes only 542 reference events to unit 1 (84.5%), the 99 remaining ones being attributed to unit 5. To illustrate this comparison, Fig. 5 displays the Wilson plots, where the amplitude on site 4 is plotted against its amplitude on site 3, after running the EM and the MCMC algorithms separately: the Gaussian mixture

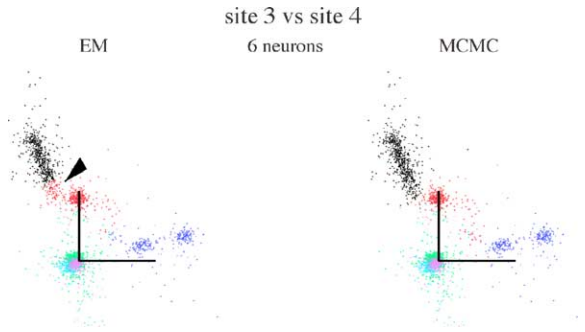

Fig. 5. Comparison between the sorting based on a classic Gaussian mixture model (GMM) and the elaborated model proposed in this paper. Wilson plots (peak amplitudes of site 3 vs. site 4) after sorting based on a classic GMM (using the EM algorithm, left) and on our new data generation model (using an MCMC algorithm, right) with six neurons. The analysis performed with the MCMC algorithm for this number of neurons is given in details in the article. The GMM fails to group all the points of the reference neuron (black cluster), so that the cluster is partially cut and attributed to the red one (arrow head): the GMM attributes 84.5% of the reference events to the reference neuron (vs. 98.1% for the MCMC algorithm, see Section 3.4.2). This illustrates how inappropriate the GMM is with non-Gaussian, elongated clusters. In such cases, implementing both the discharge statistics of the neuron and its amplitude dynamics in the spike-sorting method very satisfyingly solves the problem.

model partially truncates the elongated cluster of our reference neuron, whereas our elaborated and more realistic model does not. This excellent performance of our algorithm in such a difficult situation shows how powerful it is to incorporate the temporal information into the spike-sorting procedure through an appropriate model for the neuronal discharge statistics.

Similar results were found in five other data sets of bursting PCs, where the MCMC algorithm with the present HMM model outperformed the EM algorithm. In three of these data sets, an extra pipette separately recorded a reference cell: in these cases, our algorithm was able to rebuild more than 96% of the bursts of the reference cells, with less than 3% false positives.

3.4.3. Units 2 and 4 give rise to pairs of separated clusters on Wilson plots

Two other units deserve being examined. Unit 2 (deep blue, site 4) and unit 4 (light blue, site 2) produce doublets of spikes of very different amplitudes. Nevertheless, in both cases, these events are recognized as coming from the same cell. The corresponding raw data recorded on sites 2 and 4 are displayed in Fig. 4C1 and C2 respectively. These two panels show one typical burst of each cell: in both cases, these bursts are in fact triplets of spikes. In the case of site 4, the third spike of each burst remains below detection threshold, so that only a doublet is detected (crosses on top of the detected events). All these doublets are correctly identified as coming from unit 2 (deep blue, Fig. 4A). In the case of site 2, the third spike of each burst is detected, but is wrongly attributed to unit 6 (brown, Fig. 4A), instead of being attributed to unit 4, like the first two spikes of the triplet (light blue, Fig. 4A). This misclassification is essentially due to the fact that our model of spike waveform dynamics is not accurate enough for the data from this neuron, as discussed in Appendix A.2.2 and Fig. 8. This misclassification should moreover serve as a warning against a blind use of our algorithm which would consist in taking the output for granted without checking its relevance at all. The plots displayed in Figs. 4A and 6A and B should be drawn after each run in order to assess the quality of the sorting. In particular the ISI histograms of the sorted neurons must show a clear refractory period and an overall shape that is similar to ISI histograms of single cell recordings that can be obtained separately.

Fig. 6A shows Wilson plots of the data with the same color code as in Fig. 4A. Only two plots out of six are displayed. As in Fig. 4A, unit 1 that corresponds to the reference cell is in black. Note the elongation of this cluster. Note also the two distinct, well separated clusters of unit 4 (light blue) on the left-hand plot.

3.4.4. Empirical and modeled ISI densities

We finally display the ISI histograms of units 1 and 2, as well as the one of the reference events detected on site 3 of the multi-site electrode (Fig. 6B). The similarity between the histogram of unit 1 and that of the reference cell is another illustration of the 98% performance of the algorithm on this unit. Note the three modes of this histogram. The three ISI model densities of units 1 and 2 with parameters set at their average values computed over the last 200 MC steps are superimposed on their respective ISI histograms. For unit 1, the three scale values are (autocorrelation corrected S.D.s given in parenthesis) {13 (1.6) ms, 35 (1.5) ms, 314 (5) ms}, whereas the three shape values are {0.395 (0.059), 0.262 (0.035), 0.194 (0.013)}. The scale values approximately locate the different modes of the histogram. For unit 2, we have {5 (0.06) ms, 92 (42) ms, 374 (12) ms} and {0.148 (0.009), 1.62 (0.479), 0.224 (0.028)}. Only 20 events of unit 2 (out of 301) are found to be in state 2. They correspond to the few bins between the two modes of this histogram. As pointed out in the next to last paragraph of Section 3.2, this unit only fires 150 Hz bursts during this minute of recording. The more tonic firing that always occurs before was already over for this unit by the time the recording started. This is why almost no intermediate ISI is to be seen in this case. Each model density being of course normalized by the proportion of events in each state for a given unit, the curve corresponding to state 2 is almost null everywhere and does not appear on the plot. This shows that, although the ISI histogram of unit 2 is essentially bimodal, the behavior of the algorithm is not altered at all. Our three-state HMM can well accommodate any bi- or unimodal ISI histogram. Like in Section 3.2, this shows how well the HMM accounts for the discharge statistics of bursting cells that have tri- or bimodal ISI histograms.

4. Discussion

We have shown here how the spike-sorting algorithm we recently proposed (Pouzat, 2005; Pouzat et al., 2004), modified for multi-state neurons, performs on real, challenging data. In this data set, i.e. PCs in presence of DHPG, the recorded cells were firing bursts of spikes whose amplitudes were strongly reduced, producing distinct, well separated clusters in the Wilson plots (deep blue and light blue clusters in Fig. 6), as well as very elongated ones (black cluster in Fig. 6). To check the performance of the algorithm, the activity of one of the recorded PCs was independently and simultaneously monitored by a loose cell-attached glass micropipette and served as reference spike train. We showed that our algorithm did properly classify more than 98% of this reference spike train, despite the obvious decrease of spike amplitudes (Figs. 4A, B1 and B2 and 6A) and the tri-modal ISI histogram (Fig. 6B). We showed as well that it did associate the pairs of distinct clusters mentioned above, obviously produced by a single neuron. In such situations, existing methods require that the experimentalist a posteriori groups by himself the spikes that have been wrongly assigned to different neurons due to their changing amplitudes. None of them can automatically give such an output on these data.

The excellent performance of our method relies on its ability to take into account the information provided by the occurrence time of the spikes, as well as their amplitude dynamics. To our knowledge, this is the only method that makes use of these real spike trains properties. It is moreover built on a proper probability model for data generation which, in that case, implies that convergence proofs of the algorithm do exist (Pouzat, 2005; Pouzat et al., 2004). Our MCMC based approach provides as well meaningful confidence intervals on the model parameters and on the spike labels, a feature which should not be overlooked.

We have also illustrated here the flexibility of the MCMC framework. In our previous reports (Pouzat, 2005; Pouzat et al., 2004) we used a simpler model for the discharge dynamics of the neurons: a single log-normal density modeled the neurons ISI histograms. Here we first showed that a multi-state HMM discharge model (Camproux et al., 1996; Guclu and Bolanowski, 2004) was well supported by our single unit data (see Section 3.2). We then included this model in our spike-sorting algorithm and ran it on the multi-unit data. The message is that once one knows how to write down an MCMC algorithm for a "reduced" problem like generating the parameters of the three states discharge model of a single neuron, it is straightforward to incorporate it into the full spike-sorting algorithm. Therefore, if the experimentalist, based on single unit data (obtained for instance with patch or sharp electrode recordings) thinks that another discharge model would be better, the spike-sorting algorithm does not need to be rewritten from scratch, only a sub-part of it needs to be modified. We nevertheless think that the data generation model presented here will turn out to be a good compromise between accuracy of the description of real data and ease and speed of implementation. We did not seek to relate each individual state of our HMM to any particular biophysical event or set of events. This model has to be considered as a statistical, descriptive tool that captures the key features of the observed neuronal bursty firing. It is not limited to bursty firings though: in our model, three states are available to describe the empirical ISI density, but of course, one or two of these states can be unused for a unit that has a uni- or bimodal ISI histogram (see unit 2 in Section 3.4.4). Therefore, our model can account for uni- and bimodal ISI histograms as well, of course better than a Poisson model would. In addition, the number of states is not fixed at all and the experimentalist can choose it himself as an input to the algorithm, a priori and for each neuron.

PCs are known to tonically fire action potentials as well as bursts of spikes spontaneously in slices at 35 °C, even when fast synaptic inputs are completely blocked (Womack and Khodakhah, 2002). This spontaneous activity is preserved at room temperature but bursts of spikes are less frequent. We facilitated the spontaneous bursting behavior of PCs by adding DHPG to the bathing solution (Netzeband et al., 1997). This enables us to get multi-unit data in which most of the cells fire bursts of spikes of decreasing amplitudes and helps demonstrate the ability of our spike-sorting method to automatically isolate these bursts.

For now the method is not fully automatic in the sense that it requires the user to choose the number of neurons a priori and give it as an input to the algorithm. As discussed in Pouzat et al. (2004), the general frame of the method provides a way to reliably compare models with different numbers of neurons. This still ongoing work will be reported in a near future.

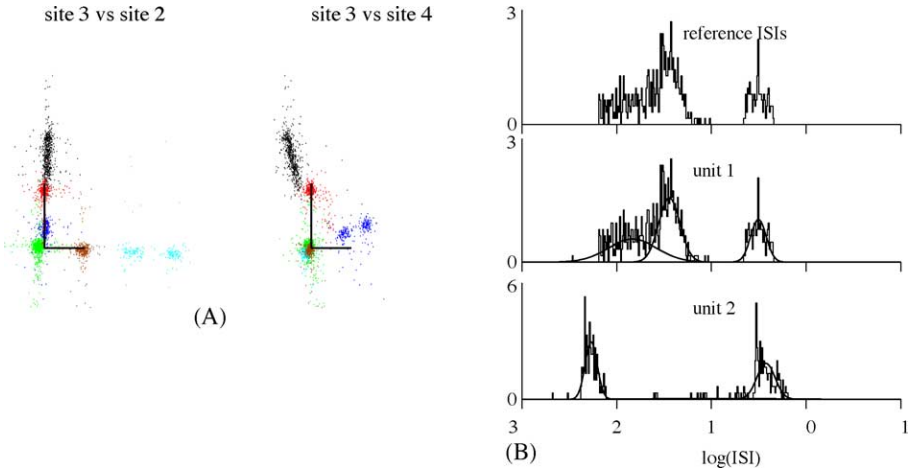

Fig. 6. Spike-sorting on a mixture of several PC spike trains simultaneously recorded (2). (A) Wilson plots (peak amplitudes of site 3 against sites 2 and 4, respectively) showing the whole recorded sample (2739 events, 58 s). Each event is colored according to the most probable label determined by our algorithm (same colors as in Fig. 4A). The reference unit is in black. Note the two pairs of separate clusters: in light blue (site 2) and deep blue (site 4) on the left-hand and right-hand plots, respectively. Note also the very elongated cluster on both plots in black (site 3). On each plot the scale bars meet at amplitude (0, 0) and are of size 10 (in units of noise S.D.). (B) $\log_{10}$ (isi) histograms of units 1 (black cluster in A) and 2 (deep blue cluster in A). Their three respective model densities whose parameters have been set at their average values computed on the last 200 iterations are superimposed on their respective histograms. The $\log_{10}$ (isi) histogram of the reference unit is also shown and to be compared to unit 1 histogram.

Acknowledgments

We thank Alain Marty and Ofer Mazor for comments and suggestions on the manuscript. This work was supported in part by a grant from the Ministère de la Recherche (ACI Neurosciences Intégratives et Computationnelles, pré-projet, 2001–2003) and by a grant inter-EPST (Bioinformatique). Matthieu Delescluse was supported by a fellowship from the Ministère de l'Education Nationale et de la Recherche. Multi-channel silicon probes were kindly provided by the University of Michigan Center for Neural Communication Technology. The manuscript was typed with LYX[6]. C codes were developed in Emacs an debugged with DDD[7].

Appendix A

A.1. Methods

A.1.1. A formal presentation of the HMM

We extend here our previous model (Pouzat, 2005; Pouzat et al., 2004) by introducing multiple *discharge states*, $D = d$, with $d \in \{1, \ldots, N_{ds}\}$, for each neuron (in this paper we set the number of states N_{ds} at 3). The goal is to be able to describe at the same time *multi-modal ISI* densities (i.e. densities with several peaks) and dependence between successive ISIs similar to what goes on during a burst, where (single) "silent periods" (long ISIs) are followed by many short ISIs (see Appendix A.2.1 for an example). Following Camproux et al. (1996) we assume that successive ISI are independent conditioned on the neuron's discharge state, d. After the emission of each spike (i.e. the generation of each ISI) the neuron can change its discharge state or keep the same. The inter-discharge state dynamics is given by a Markov matrix, $Q = (q_{ij})$. We moreover assume that the ISI distribution of each state is log-normal. In other words we assume that the neuron after its mth spike is in state d and that the ISI between this spike and the next one is distributed as

$$\mathrm{ISI}|d \sim \text{log-normal}(s_d, \sigma_d) \tag{7}$$

Eq. (7) should be read as

$$\pi_{\mathrm{isi}}\left(\mathrm{ISI} = \mathrm{isi}|S = s_d, \sum = \sigma_d\right) = \frac{1}{\mathrm{isi}\,\sigma_d}\sqrt{\frac{1}{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{\log(\mathrm{isi}) - \log(s_d)}{\sigma_d}\right)^2\right) \tag{8}$$

where S is a scale parameter (measured in s) and Σ is a shape parameter (dimensionless). These random variables do depend on the value taken by the random variable D. After the ISI has been generated, the neuron can "move" to any of its N_{ds} states according to a probabilistic dynamics described by

$$P(D^{(m+1)} = j|D^{(m)} = i) = q_{ij} \tag{9}$$

You see therefore that if we work with a neuron with three discharge states we have 12 independent ISI parameters to estimates: two pairs (s, σ) per state and N_{ds} $(N_{ds} - 1)$ state transition parameters (do not forget that matrix (q_{ij}) is stochastic and therefore its rows sum to 1).

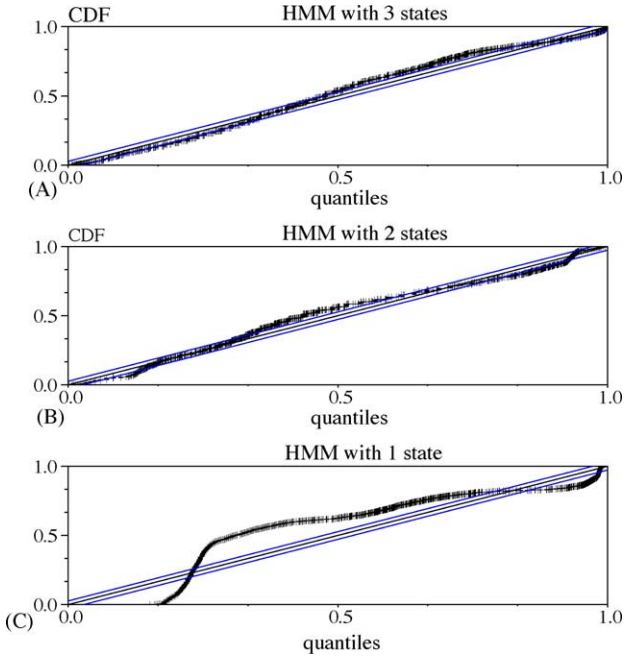

Fig. 7. Kolmogorov–Smirnov (K–S) plots of the hidden Markov model fits to the PC spike train shown in Figs. 1 and 2. (A) HMM with three log-normal states. (B) HMM with two log-normal states. (C) HMM with one log-normal state. In A, B and C, the solid 45° line represents exact agreement between the model and the data. The 45° lines on both sides are the 95% confidence bounds for exact agreement between the model and experimental data based on the Kolmogorov–Smirnov statistics. Although not perfect, a HMM with three states obviously fits better the PC spike train than a HMM with two or one state(s). CDF stands for cumulative distribution function. For details about Kolmogorov–Smirnov tests, see Brown et al. (2001).

A.1.2. Generating a new state in the Markov chain

In this section, we use Y to designate the data. At step t, state $[\theta^{(t)}, C^{(t)}]$ is drawn from state $[\theta^{(t-1)}, C^{(t-1)}]$ by successively drawing each model parameter and each spike label and spike state according to the procedures described below. To simplify notations we omit the step index (t) of the generated state. We note C_{-i} the configuration specifying the labels and neuron states for all the spikes except spike number i. Similarly we note $\theta_{-\alpha}$ the vector of all model parameters except parameter a. Each parameter a has a uniform prior on a defined segment $[a_{\min}, a_{\max}]$ relevant for this parameter (Pouzat et al., 2004). A step of our algorithm is performed once all spike labels and states, as well as all model parameters have drawn as specified below. This defines the new state in the Markov chain.

[6] http://www.lyx.org.
[7] http://www.gnu.org/software/ddd/.

A.1.2.1. Labels and neuron states

For each spike i of the spike train, a label $l_i \in \{1, \ldots, K\}$ and a neuron state $m_i \in \{1, 2, 3\}$ are drawn from their posterior conditional density:

$$l_i, m_i | Y, \theta, C_{-i} \propto \frac{\pi_{\text{post}}(l_i, m_i | Y, \theta, C_{-i})}{\sum_{l_j, m_j} \pi_{\text{post}}(l_j, m_j | Y, \theta, C_{-i})} \tag{10}$$

A.1.2.2. Amplitude parameters

For each neuron q, the amplitude parameters ($P_{q,1}$, $P_{q,2}$, $P_{q,3}$, $P_{q,4}$, δ_q, λ_q) are drawn with a Metropolis–Hastings step, using piecewise linear approximations of their respective posterior conditional densities as proposals (Pouzat et al., 2004). Let us take the case of λ_q, for example, to illustrate the procedure. Let $\tilde{\pi}_{\text{post}}(\lambda_q | Y, \theta_{-\lambda_q}, C)$ be its posterior conditional density and $\tilde{\pi}_{\text{approx}}(\lambda_q | Y, \theta_{-\lambda_q}, C)$ be its piecewise linear approximation. Let λ be the current value of λ_q.

First, $\tilde{\lambda}$ is drawn from the proposal density:

$$\tilde{\lambda} | Y, \theta_{-\lambda_q}, C \propto \tilde{\pi}_{\text{approx}}(\lambda_q | Y, \theta_{-\lambda_q}, C) \tag{11}$$

Then, this value λ is accepted with probability A equal to

$$A = \min\left(1, \frac{\pi_{\text{post}}(\tilde{\lambda} | Y, \theta_{-\lambda_q}, C)\tilde{\pi}_{\text{approx}}(\lambda | Y, \theta_{-\lambda_q}, C)}{\pi_{\text{post}}(\lambda | Y, \theta_{-\lambda_q}, C)\tilde{\pi}_{\text{approx}}(\tilde{\lambda} | Y, \theta_{-\lambda_q}, C)}\right) \tag{12}$$

If $\tilde{\lambda}$ is accepted, then $\lambda \rightarrow \tilde{\lambda}$.

Else $\lambda \rightarrow \lambda$.

A.1.2.3. Log-normal parameters

Scale parameters. For each neuron q and each neuron state r ($r \in \{1, 2, 3\}$) of neuron q, the scale parameter of the log-normal density is noted s_q^r.

First u is drawn from (Pouzat et al., 2004):

$$u | Y, \theta_{-s_q^r}, C \propto \text{norm}\left(\overline{\log i_q}, \frac{(\sigma_q^r)^2}{n_q}\right) \tag{13}$$

where n_q is the number of ISI of neuron q, σ_q^r the shape parameter of neuron q in neuron state r, and $\overline{\log i_q} = 1/n_q \sum_{j=1}^{n_q} \log i_{q,j}$, $i_{q,j}$ being the ISI index j of neuron q.

Then, if $s = \exp(u) \in [s_{\min}, s_{\max}]$, we set $s_q^r = s$.

Else we draw another u.

Shape parameters. For each neuron q and each neuron state r ($r \in \{1, 2, 3\}$) of neuron q, the shape parameter of the log-normal density is noted σ_q^r.

First u is drawn from (Pouzat et al., 2004):

$$u | Y, \theta_{-\sigma_q^r}, C \propto \text{gamma}\left(\frac{n_q}{2} - 1, \frac{n_q}{2}(\overline{\log i_q} - \log s_q^r)^2\right) \tag{14}$$

with the same notations as for the scale parameters.

Then, if $\sigma_{\min} \leq \sqrt{1/u} \leq \sigma_{\max}$ we set $\sigma_q^r = \sqrt{1/u}$.

Else we draw another u.

A.1.2.3. Transition parameters

For each neuron q, the transition parameters between the three HMM neuron states form a 9×9 matrix whose three rows are drawn successively.

Let $m = (m_1, \ldots, m_N)$ be the spike train configuration of a neuron q, where m_k is the neuron state of spike k of this neuron. Let n_{ij} be the number of spikes of this neuron which are in state m_j following a spike of this neuron in state m_i. The row number i of the transition matrix is then drawn from the Dirichlet distribution $D_3(1 + n_{i1}, 1 + n_{i2}, 1 + n_{i3})$ (Robert and Casella, 1999).

A.1.3. Implementation details

Codes were written in C. We used the free softwares Scilab[8] and R[9] to generate output plots as well as the graphical user interface. The GNU Scientific Library[10] (GSL) was used for vector and matrix manipulation routines and (pseudo-) random number generators. More specifically, the GSL implementation of the Mersenne Twister of Matsumoto and Nishimura (1998) was used to generate random variates. Codes were compiled with the *intel* icc compiler[11] and run on a PC (Pentium IV 3 GHz) running Linux.

A.2. Supplementary analysis

A.2.1. Dependence between successive ISIs in the single unit spike train

The transition matrix (q_{ij}) of the most probable configuration (i.e. the attribution of a neuron state to each spike) of the single unit spike strain shown in Figs. 1 and 2 is given below. The lowest and largest values taken by each transition element over the last 200 MC-steps are given in square brackets. The neuron state numbers (i.e. here, the row and column numbers) are those of Figs. 1 and 2, that is: states 1, 2 and 3 for the short, intermediate and long ISIs, respectively. The dependence between ISIs is obvious: a long ISI is always followed by a short ISI ($q_{31} = 1$), a short ISI is either followed by another short ISI (within a burst), or by a long ISI almost exclusively ($q_{11} = 0.69$ and $q_{13} = 0.3$). This is in agreement with the existence of bursts separated by longer intervals. This may be related to the refractory period after high frequency discharge in burst. If successive ISIs were independent, rows would be identical, each column being equal to the proportion of the state.

	State 1	State 2	State 3
State 1	0.69 [0.62; 0.76]	0.01 [0; 0.03]	0.3 [0.23; 0.38]
State 2	0.02 [0; 0.06]	0.98 [0.94; 0.99]	0 [0; 0.03]
State 3	1 [0.91; 1]	0 [0; 0.08]	0 [0; 0.04]

[8] http://scilabsoft.inria.fr.
[9] http://www.r-project.org.
[10] http://sources.redhat.com/gsl.
[11] http://www.intel.com/software/products/compilers/.

A.2.2. Why are the third spikes of unit 4 triplets wrongly attributed to unit 6?

First of all, the event amplitudes of unit 6 are very similar to the amplitudes of the third spikes of unit 4 triplets, which considerably complicates the separation between these two units. In fact, the likelihood of the data is significantly smaller when these spikes are rightly labeled as unit 4 than when they are labeled as unit 6, which explains the output of the algorithm. This is due to the fact that our model of waveform dynamics is not sufficiently supported by data from unit 4, so that, with this model, its third spikes in bursts are more likely to come from unit 6, whose events are of similar amplitude, as shown in Fig. 4A, site 2. This point is described in detail in Fig. 8. Second, as illustrated on the raw data of site 2 (Fig. 4C1), the third spikes of these bursts have an overall different waveform (note that the valley preceding the peak almost disappears). In this case, we are not dealing with a simple homothetic scaling of the waveform. That is why we also ran the algorithm using three points per site and per event, instead of the peak amplitude only. This was not sufficient to correctly label these spikes as unit 4. In fact, using three points per site and per event instead of the peak amplitude did not change the output of the algorithm in this case. Third, the spikes at stake here are really small spikes that might even not be detected in other circumstances. Whatever the spike-sorting method, small events are always less reliably labeled and the experimentalist has to leave them out and keep the unambiguous ones. We certainly do not claim that our method can overcome this limit. In this case, any reasonable experimentalist who has been dealing with spike-sorting would not take into account these events.

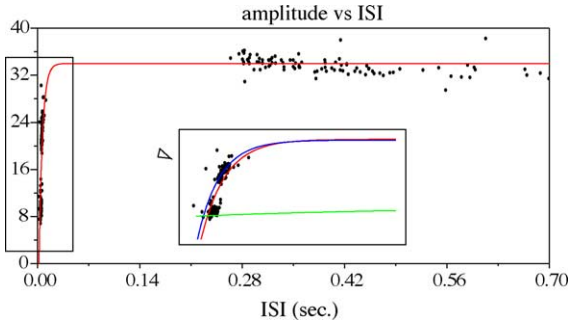

Fig. 8. Amplitude dynamics of spikes within the triplets of unit 4 shown in light blue in Fig. 4A. The third spikes of unit 4 triplets (labeled as unit 6 by our algorithm) and the spikes labeled as unit 4 are pooled together to form the real spike train of unit 4. The peak amplitudes (after noise whitening) of these spikes are plotted with respect to the ISI preceding the spike (black dots). The fitted exponential relaxation is superimposed (red solid line). Ordinates in units of noise S.D. The inset shows an expanded version of the left-hand part of the plot and shows that the model does not perfectly account for the waveform dynamics of this unit. This leads to a relatively low likelihood of this spike train, whereas removing the third spikes (lowest group of points) of each triplet from unit 4 and attributing them to unit 6 allows a better global fit (blue solid line for unit 4 and green solid line for unit 6). The likelihood of the whole is lower in the latter case, i.e. when the third spikes of unit 4 triplets are labeled as unit 6. This explains why the algorithm fails to fully reconstruct unit 4 triplets.

References

Baker SN, Philbin N, Spinks R, Pinches EM, Wolpert DM, MacManus DG, et al. Multiple single unit recording in the cortex of monkeys using independently moveable microelectrodes. J Neurosci Meth 1999;94:5–17.

Brown EN, Barbieri R, Ventura V, Kass RE, Frank LM. The time-rescaling theorem and its application to neural spike train data analysis. Neural Comp 2001;14:325–46.

Brown EN, Kass RE, Mitra PP. Multiple neural spike train analysis: state-of-the-art and future challenges. Nat Neurosci 2004;5:456–61.

Buzsaki G. Large scale recording of neuronal ensembles. Nat Neurosci 2004;5:446–51.

Camproux AC, Saunier F, Chouvet G, Thalabard JC, Thomas G. A hidden Markov model approach to neuron firing patterns. Biophys J 1996;71:2404–12.

Csicsavari J, Henze DA, Jamieson B, Harris KD, Sirota A, Bartho P, et al. Massively parallel recording of unit and local field potentials with silicon-based electrodes. J Neurophysiol 2003;90:1314–23.

Drake KL, Wise KD, Farraye J, Anderson DJ, Bement SL. Performance of planar multisite micro-probes in recording extracellular single-unit intracortical activity. IEEE Trans Biomed Eng 1988;35:719–32.

Egert U, Heck D, Aertsen A. Two-dimensional monitoring of spiking networks in acute brain slices. Exp Brain Res 2002;142:268–74.

Fee MS, Mitra PP, Kleinfeld D. Automatic sorting of multiple-unit neuronal signals in the presence of anisotropic and non-Gaussian variability. J Neurosci Meth 1996a;69:175–88.

Fee MS, Mitra PP, Kleinfeld D. Variability of extracellular spike waveforms of cortical neurons. J Neurophys 1996b;76:3823–33.

Frenkel D, Smit B. Understanding molecular simulation. In: From algorithms to applications. San Diego: Academic Press; 2002.

Gray CM, Maldonado PE, Wilson M, McNaughton B. Tetrode markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. J Neurosci Meth 1995;63:43–54.

Gross GW, Harsch A, Rhoades BK, Reust DL, Goepel W. Odor, drug and toxin analysis with neuronal networks in vitro: extracellular array recording of network responses. Biosens Bioelectron 1997;12:373–93.

Gross GW, Rhoades BK, Reust DL. Schwalm FU Stimulation of monolayer networks in culture through thin-film indium-tin oxide recording electrodes. J Neurosci Meth 1993;50:131–43.

Guclu B, Bolanowski SJ. Tristate Markov Model for the firing statistics of rapidly-adapting mechanore-ceptive fibers. J Comput Neurosci 2004;17:107–26.

Hansmann UHE. Parallel tempering algorithm for conformational studies of biological molecules. Chem Phys Lett 1997;281:140–50.

Hukushima K, Nemoto K. Exchange Monte Carlo method and application to spin glass simulations. J Phys Soc Jpn 1996;65:1604–8.

Iba Y. Extended Ensemble Monte Carlo. Int J Mod Phys C 2001;12:623–56.

Janke W. Statistical analysis of simulations: data correlations and error estimation; 2002, http://www.fz-juelich.de/nic-series/volume10 [March 29, 2005].

Lewicki MS. A review of methods for spike-sorting: the detection and classification of neural action potentials. Network Comput Neural Syst 1998;9:R53–78.

Loewenstein Y, Mahon S, Chadderton P, Kitamura K, Sompolinsky H, Yarom Y, et al. Bistability of cerebellar Purkinje cells modulated by sensory stimulation. Nat Neurosci 2005;8:202–11.

Matsumoto M, Nishimura T. ACM Trans Model Comp Sim 1998;8:3–30.

McCormick D. Membrane properties and neurotransmitter actions. In: Shepherd GM, editor. The synaptic organization of the brain. New York, NY: Oxford University Press; 1998. p. 37–75.

Netzeband JG, Parsons KL, Sweeney DD, Gruol DL. Metabotropic glutamate receptor agonists alter neuronal excitability and Ca2+ levels via the phospholipase C transduction pathway in cultured Purkinje neurons. J Neurophysiol 1997;78:63–75.

Newman MEJ, Barkema GT. Monte Carlo methods in statistical physics. Oxford: Oxford University Press; 1999.

Nicolelis MAL, Ghazanfar AA, Faggin BM, Votaw S, Oliveira LM. Reconstructing the engram: simultaneous, multisite many single neuron recordings. Neuron 1997;18:529–37.

Oka H, Shimono K, Ogawa R, Sugihara H, Taketani M. A new planar multielectrode array for extracellular recording: application to hippocampal acute slice. J Neurosci Meth 1999;93:61–7.

Pouzat C. Technique(s) for spike-sorting. In: Dalibard J, editor. Methods and models in neurophysics. Berlin: Springer-Verlag; 2005. p. 729–86.

Pouzat C, Delescluse M, Viot P, Diebolt J. Improved spike-sorting by modeling firing statistics and burst-dependent spike amplitude attenuation: a Markov chain Monte Carlo approach. J Neurophysiol 2004;91:2910–28.

Pouzat C, Hestrin S. Developmental regulation of basket/stellate cell → Purkinje cell synapses in the cerebellum. J Neurosci 1997;17:9104–12.

Pouzat C, Mazor O, Laurent G. Using noise signature to optimize spike-sorting and to asses neuronal classification quality. J Neurosci Meth 2002;122:43–57.

Robert CP, Casella G. Monte Carlo statistical methods. New York: Springer-Verlag; 1999.

Segev R, Goodhouse J, Puchalla J, Berry MJ. Recording spikes from a large fraction of the gaglion cells in a retinal patch. Nat Neurosci 2004;7:1155–62.

Sokal AD. Monte Carlo in Statistical Mechanics: Foundations and New Algorithms. Cours de Troisième Cycle de la Physique en Suisse Romande; 1989, http://citeseer.nj.nec.com/sokal96monte.html [March 29, 2005].

Womack M, Khodakhah K. Active contribution of dendrites to the tonic and trimodal patterns of activity in cerebellar Purkinje neurons. J Neurosci 2002;22:10603–12.

5.3 Troisième article : le logiciel *SpikeOMatic*

Toutes les routines nécessaires à nos méthodes de *spike-sorting*, depuis la détection des PAs jusqu'à l'exploitation des résultats des algorithmes d'*Expectation-Maximization* (EM) et MCMC, font l'objet d'un logiciel sous licence GPL, disponible gratuitement sur le site internet du laboratoire : *SpikeOMatic* (voir chapitre 4, section 4.4). Afin de rendre possible son utilisation par d'autres personnes que ses concepteurs, nous avons rédigé un manuel de l'utilisateur. Ce manuel doit être autonome vis-à-vis des deux articles précédents. Nous commençons donc par exposer les modèles de génération de données utilisés par les deux algorithmes, EM et MCMC, implémentés dans ce logiciel. Cette première partie du manuel reprend donc et résume les exposés méthodologiques des deux premiers articles. Nous décrivons ensuite pas à pas le déroulement d'une analyse *complète* de données multiunitaires, avec la méthode EM, et avec la méthode MCMC. Cette analyse "de démonstration" est réalisée sur les données du deuxième article. Elle a recours à toutes les fonctions implémentées dans notre logiciel. L'utilisation de chacune d'entre elles y est donc décrite. Dans la mesure où elle détaille linéairement la procédure de *spike-sorting*, cette deuxième partie du manuel est proche de la section 4.2 du chapitre des méthodes du présent manuscrit. Grâce à ce manuel, un expérimentateur peut utiliser notre logiciel sans connaître les méthodes sous-jacentes.

Ce manuel a la forme d'une vignette R. Comme on le verra dans le quatrième article, une vignette est un "méta-fichier" permettant d'associer dans un même document (i) un texte classique de description et d'explication, (ii) des lignes de code qui sont exécutées, (iii) les résultats des codes insérés. Cette forme convient donc particulièrement à l'élaboration d'un manuel d'utilisation de logiciel. Le *Journal of Statistical Software* publie de tels manuels ; le logiciel lui-même, ainsi que son utilisation, font évidemment l'objet d'une évaluation au même titre que le manuel. Celui-ci n'a pas encore été soumis.

Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. http://www.jstatsoft.org/

An R Package for Spike-Sorting: SpikeOMatic

Matthieu Delescluse
Université René Descartes
CNRS UMR 8118

Christophe Pouzat
Université René Descartes
CNRS UMR 8118

Abstract

We describe an **R** package that allows to perform automatic spike-sorting on raw multi-unit data obtained in extracellular recordings. We resort to two different data generation models. The first one is a classical *Gaussian mixture model* (*GMM*) (Pouzat *et al.* (2002)), on which statistical inference is made with the *Expectation-Maximization* (*EM*) algorithm. The second model is significantly more sophisticated and is described here as the *Dynamic Hidden Markov Model* (*DHMM*). Statistical inference on this model is possible using a *Markov Chain Monte Carlo* (*MCMC*) algorithm presented in detail in Pouzat *et al.* (2004) and outlined in the present paper. The package implements both algorithms and their use is illustrated on real multi-unit data.

Keywords: Bayesian inference, Markov Chain Monte Carlo, maximum likelihood, Expectation-Maximization.

Contents

1. Introduction

In electrophysiology multi-site extracellular recordings are used to study neuronal populations activity. In such experiments the activities of several neurons are simultaneously tracked, giving rise to a mixture of spike trains that has first to be decomposed into the spike trains of individual neurons. This difficult problem is known as the "spike-sorting" problem. We developed the software **SpikeOMatic** to perform spike-sorting on multi-unit data. This is an R package, together with a C library. In this paper we demonstrate the use of this software and give details on how to achieve a complete spike-sorting on raw multi-unit data.

SpikeOMatic implements two different spike-sorting techniques to make statistical inference on two data generation models with different complexities. The first model is a classical *Gaussian mixture model* (*GMM*) whose maximum likelihood is estimated using the *Expectation-Maximization* (*EM*) algorithm, as described in Pouzat *et al.* (2002) and outlined in section 2. This method proves to be sufficient with certain data sets (Pouzat *et al.* (2002)). It nevertheless falls short in other cases (Delescluse and Pouzat) and in general we need to consider a more realistic data generation model. The latter is described in section 3 as the *Dynamic Hidden Markov Model* (*DHMM*). To make inference on this more complex model, we developed a *Markov Chain Monte Carlo* (*MCMC*) algorithm whose outline is also given in section 3. All details on this method are to be found in Pouzat *et al.* (2004) and Pouzat (2005).

Sections 4, 5 and 6 go step by step through a complete analysis of real multi-unit data using **SpikeOMatic** and the **SpikeOMatic Library** in the R environment. In these sections function calls, resulting output and explanatory texts are intermingled. We thus show how to use the main functions of our software and in which order. An on-line help gives complementary information if needed and is obtained by typing `?function_name` as usual in R.

2. EM for spike-sorting

This algorithm maximizes the log-likelihood function $\mathcal{L}$ of the spike train Y, given a multivariate Gaussian mixture model M, with K components (Pouzat *et al.* (2002)).

We note $Y = (\mathbf{e}_1, ..., \mathbf{e}_N)$ the data sample, each spike being represented by a D-dimensional vector $\mathbf{e}_i$.

The probability P_i for model M to have generated spike $\mathbf{e}_i$ is given by:

$$P_i = \sum_{j=1}^{K} \pi_j \cdot p(\mathbf{e}_i \mid \mathbf{u}_j, \Sigma_j) \tag{1}$$

where π_j is the proportion of component j in model M, $\mathbf{u}_j$ and Σ_j being respectively the mean and covariance matrix of this component, *i.e* $p(\mathbf{e}_i \mid \mathbf{u}_j, \Sigma_j)$ is given by:

$$p(\mathbf{e}_i \mid \mathbf{u}_j, \Sigma_j) = \frac{1}{2\pi^{D/2}} \cdot \exp(-\frac{1}{2} \cdot (\mathbf{e}_i - \mathbf{u}_j)^T \Sigma_j^{-1} (\mathbf{e}_i - \mathbf{u}_j)) \tag{2}$$

Assuming the independence of the N spikes the probability of observing spike train Y given model M is the product of the probabilities to observe each spike separately, that is:

$$\mathcal{P}(Y \mid M) = \prod_{i=1}^{N} P_i$$

The log-likelihood is then the logarithm of this quantity:

$$\mathcal{L}(Y \mid M) = \sum_{i=1}^{N} \log(P_i)$$

$\mathcal{L}$ is maximized using the *Expectation-Maximization* (*EM*) algorithm (Dempster *et al.* (1977)). It thus finds the best model for a given number of neurons K.

To choose the actual number of neurons, we compare models with different number of neurons using the *Bayesian Information Criterion* (BIC), proposed by Schwarz (1978). The BIC penalizes an increase in the number of components by adding to $(-\mathcal{L})$ a term proportional to $\nu \cdot \log(N)$, where ν is the number of parameters in the model. Therefore we keep the model whose value of K minimizes the BIC (or maximizes its opposite).

3. MCMC for spike-sorting

3.1. Why does SpikeOMatic go MCMC?

The Gaussian mixture model considered in the previous section is a simple data generation model that falls short in many cases (Delescluse and Pouzat). You want to consider using the *MCMC* extension of **SpikeOMatic** when the neurons you record from exhibit one or both of the following features (Pouzat *et al.* (2004), Pouzat (2005)):

- The spike amplitude of the neurons depends on the inter-spike interval (*ISI*). For instance, one commonly observes during a burst a reduction of the spike amplitude upon short *ISI*.
- You want to include in the spike-sorting procedure information about the discharge statistics of the neurons, because, among other things, you can then perform good sorting even for neurons whose clusters overlap strongly on *Wilson plots* (see Pouzat (2005)).

The capability do deal with such situations, at least our solution to these situations, comes at a price. We need to use an explicit data generation model which is significantly more sophisticated than the "simple" *GMM*. As already stated, we will dub this model, the *Dynamic Hidden Markov Model*. The first term (*Dynamic*) refers to the spike amplitude dynamics, the next two terms (*Hidden Markov*) refer to the neuron's discharge statistics. Statistical inference on the *GMM* could be easily done with the *EM*, whereas the *DHMM* requires the rather heavy computational machinery of the *MCMC* approach.

3.2. Data generation model: the Dynamic Hidden Markov Model

The Dynamic Hidden Markov Model we present here is our current compromise between model complexity (the number of parameters of the model) and its flexibility to account

"reasonably" well for actual neuronal data. It is not designed to provide a perfect fit, but to capture essential features of the data. These features are the spike amplitude dynamics and a possibly dynamic (*e.g.*, bursty) inter-spike interval generation.

Spike amplitude relaxation

In what follows we assume that the relation between the amplitude $\mathbf{a}$ of a spike (with no recording noise) and the elapsed time since the last spike of the same neuron, the inter-spike interval (*isi*), is given by an exponential relaxation (Fee *et al.* (1996)):

$$\mathbf{a}(isi \mid \mathbf{P}_{max} = \mathbf{p}_{max}, \Delta = \delta, \Lambda = \lambda) = \mathbf{p}_{max} \cdot (1 - \delta \cdot \exp(-isi \cdot \lambda)), \tag{3}$$

where $\mathbf{P}_{max}$ is the maximal possible amplitude of the spikes (that is, the one we would observe following a long enough *isi*), Δ is a modulation factor (dimensionless) and Λ is the inverse of the relaxation time constant (measured in s^{-1}). Notice that we use vectors to represent spikes' amplitudes. This is because we typically use several amplitude samples to represent a spike (*e.g.*, in the sequel we will use the peak amplitude of the spikes on each of four recording sites, our $\mathbf{a}$ and $\mathbf{p}_{max}$ objects will therefore be vectors in a 4-dimensional space). This relaxation model means that for each neuron we will have to estimate $n_s + 2$ *amplitude parameters* (n_s is the number of amplitude samples used to describe a spike, 4 in the sequel).

We will now assume that our measured amplitudes are corrupted by a *Gaussian white noise* (with a standard deviation of 1) which is statistically independent of the signal (the spikes). In other words we will have to *whiten the noise* before running our spike-sorting procedure (Pouzat *et al.* (2002)) as explained in Sect. 4.5. Then the probability (density) to observe a spike, $\mathbf{s}$, following the previous one (from the same neuron) by *isi* (seconds) becomes:

$$\pi_{amp}(\mathbf{S} = \mathbf{s} \mid ISI = isi, \mathbf{P}_{max} = \mathbf{p}_{max}, \Delta = \delta, \Lambda = \lambda) = (2\pi)^{-\frac{n_s}{2}} \cdot \exp\left(-\frac{1}{2} \parallel \mathbf{s} - \mathbf{a}(isi) \parallel^2\right), \tag{4}$$

where $\mathbf{a}(isi)$ is given by Eq. 3 and $\parallel \mathbf{v} \parallel$ represents the Euclidean norm of vector $\mathbf{v}$.

We will try to consistently use symbol $\pi_{something}$ to designate a *properly normalized* probability (density) function. We will use capital letters to designate *random variables* and small cap letters to designate their *realizations*.

Neuronal discharge statistics

We will extend here our previous model (Pouzat *et al.* (2004)) by introducing multiple *discharge states*, $D = d$, with $d \in \{1, \ldots, N_{ds}\}$, for each neuron. The goal is to be able to describe at the same time *multi-modal ISI* densities (*i.e.*, densities with several peaks) and dependence between successive *ISIs* similar to what goes on during a burst, where (single) "silent periods" (long *ISIs*) are followed by many short *ISIs*. Following Camproux et al. (Camproux *et al.* (1996)) we will assume that successive *isi* are independent *conditioned on the neuron's discharge state*, d. After the emission of each spike (*i.e.*, the generation of each *isi*) the neuron can change its *discharge states* or keep the same. The inter discharge state dynamics is given by a *Markov matrix*, $Q = (q_{i,j})$. We will moreover assume that the *isi* distribution of each state is *log-normal*. In other words we assume that the neuron after its mth spike is in state d and that the *isi* between this spike and the next one is distributed as:

$$ISI \mid d \sim \text{log-normal}(s_d, f_d) \tag{5}$$

Eq. 5 should be read as:

$$\pi_{isi}(ISI = isi \mid S = s_d, F = f_d) = \frac{1}{isi \cdot f_d \cdot \sqrt{2\pi}} \cdot \exp\Big(-\frac{1}{2}(\frac{\log(isi) - log(s_d)}{f_d})^2\Big) \tag{6}$$

where S is a *scale* parameter (measured in s) and F is a *shape* parameter (dimensionless). These random variables do depend on the value taken by the random variable D. After the *isi* has been generated, the neuron can "move" to any of its N_{ds} states according to a probabilistic dynamics described by:

$$\mathrm{P}(D^{m+1} = k \mid D^m = j) = q_{j,k} \tag{7}$$

You see therefore that if we work with a neuron with 3 discharge states we have 12 independent *isi parameters* to estimates: 2 pairs (s, f) per state and $N_{ds} \cdot (N_{ds} - 1)$ state transition parameters (do not forget that matrix $(q_{i,j})$ is *stochastic* and therefore its rows sum to 1).

More details about this model will hopefully be soon readily available to everyone (Delescluse and Pouzat).

3.3. The MCMC approach

This general class of methods has been used for fifty years by physicists (Metropolis *et al.* (1953) who call it *Dynamic Monte Carlo* and for 20 years by statisticians (Fishman (1996), Geman and Geman (1984), Liu (2001), Robert and Casella (1999)) who call it *Markov Chain Monte Carlo*. Before describing its principle, we need to specify notations concerning the *model parameters* and what will be call a *configuration*.

Model parameters

You have probably noticed and you have perhaps been surprised by the fact that in Eq. 3, 4 & 6 we treat our model parameters:

$$\{\mathbf{P}_{max}, \Delta, \Lambda, S_j, F_j, (q_{i,j})\} \tag{8}$$

like *random variables*. That is because we are using a Bayesian approach. Luckily for you, you do not have to take side in the religious war opposing *frequentists* and *Bayesian* to use our algorithm. If you're deeply convinced that the likelihood approach is the beginning and the end of any respectable statistical analysis you can view our use of the Bayesian framework as an easy way to introduce constraints on our model parameters. Considering as well that the nature of our problem would require a Monte Carlo algorithm (that is an *MCMC* method) to tackle the associated maximum likelihood problem, it just turns out that going fully Bayesian represents a slight extra computational cost.

Fine, so let us say we consider for now a model with K neurons. Our task includes the estimation of the parameters of every neuron. We will call Θ this complete parameter vector:

$$\Theta = (\mathbf{P}_{max,1}, \Delta_1, \Lambda_1, S_{d,1}, F_{d,1}, (q_{i,j})_1, \ldots, \mathbf{P}_{max,K}, \Delta_K, \Lambda_K, S_{d,K}, F_{d,K}, (q_{i,j})_K) \tag{9}$$

Configuration

We are of course directly interested (not to say mainly interested) in the neuron of origin of each detected spike. Or more precisely, the posterior probability for each neuron of the model to have generated each spike. In order to do that, we will associate to every detected spike, $j \in \{1, \ldots, N\}$, 2 random variables: L_j and $D_j \mid l_j$. L_j takes value in $\{1, ..., K\}$ and its distribution is what we are fundamentally looking for: the probability for each neuron of the model to have generated spike j. $D_j \mid l_j$ is the discharge state we introduced in Sect. 3.2.2 (we are just making explicit here the fact that D_j depends on the value taken by L_j). We will therefore have to estimate the posterior distribution of what we will call the *configuration*:

$$C = ((L_1, D_1), \ldots, (L_N, D_N)) \tag{10}$$

The fundamental reason to use a simulation based inference (that is, a Monte Carlo method) is the number of elements of the space on which C takes value: $(KN_{ds})^N$. This becomes astronomical for realistic values: $K = 10$, $N_{ds} = 3$ and $N = 1000$.

Posterior density and the MCMC method

MCMC algorithms solve this inference problem by generating a Markov Chain on the space defined by the problem's unknowns, that is, in our case, the space on which (Θ, C) are defined. This Markov Chain is built such that it converges to a *unique* distribution which is precisely the posterior distribution of the unknowns given the data (see Pouzat *et al.* (2004) and Pouzat (2005) for details). In our case:

$$\pi_{post}(\theta, c \mid \mathcal{D}) = \frac{L(\mathcal{D}, c \mid \theta) \cdot \pi_{prior}(\theta)}{Z}, \tag{11}$$

where $\mathcal{D}$ represents the data (spike amplitudes and occurrence time), $L(\mathcal{D}, c \mid \theta)$ is the likelihood of the data *and* the configuration given the model parameters, $\pi_{prior}(\theta)$ is the prior density of the model parameters and Z is the normalizing constant:

$$Z = \sum_{c \in \mathcal{C}} \int_{\theta} d\theta L(\mathcal{D}, c \mid \theta) \pi_{prior}(\theta), \tag{12}$$

where $\mathcal{C}$ is the space on which C is defined.

Formally our Markov Chain tells us how we move from the present "system" state, (θ, c), to a new one, (θ', c'), by specifying a transition kernel: $T((\theta, c) \rightarrow (\theta', c'))$. What we do on our computer is a Monte Carlo simulation of this Markov Chain (that's what this fancy acronym *MCMC* means). We will therefore end up with a sequence of states:

$$\{(\theta^0, c^0), (\theta^1, c^1), \ldots, (\theta^M, c^M)\} \tag{13}$$

where (θ^0, c^0) is the initial state and M is the number of Monte Carlo (MC) steps performed. Then if we want to get the most likely neuron of origin for spike 100 we compute the following quantity:

$$\max_{j \in \{1, \ldots, K\}} \frac{1}{M - m} \sum_{i=m}^{M} \delta_{l^i_{100}, j} \tag{14}$$

where $\delta_{l^i_{100},j}$ is the Kroenecker's symbol and m is the step at which we start "taking measurements", *i.e* the number of steps we judge sufficient to forget about the initial state (see Pouzat *et al.* (2004) and Pouzat (2005)).

Energy

Now, as a user of *MCMC* methods you should be aware that the convergence theorems hold for an *infinite* number of steps. On your computer you can of course perform only a *finite* number of steps. You therefore have to make sure that the behavior of your simulated Markov Chain is *compatible* with a chain at "steady-state" or "equilibrium". In this tutorial we will do that qualitatively by monitoring the evolution of the *energy* of the Markov Chain that we define by:

$$\mathrm{E}(\theta, c) = -\log\left(L(\mathcal{D}, c \mid \theta)\pi_{prior}(\theta)\right) \tag{15}$$

You can qualitatively see E as a χ^2 value, that is, the smaller it is, the better the fit. Of course, if the Markov Chain has reached equilibrium, E should oscillate around a constant mean value. `But be careful, for this is a necessary, not a sufficient condition for equilibrium`.

The Replica Exchange Method

One more "theoretical" point before we can start playing with the algorithm: the energy landscape of our systems are typically rough and our Markov Chain dynamics is local. That means we can get trapped in local energy minima for a long time. To circumvent this problem we have recourse to the *Replica Exchange Method* (*REM*) (Hukushima and Nemoto (1996), Mitsutake *et al.* (2001)) which consists in simulating multiple replica of our system at higher and higher "temperatures", so that the replica at high temperature can escape from local minima (see Pouzat *et al.* (2004) and Pouzat (2005) for details).

The temperature parameter, or rather the inverse temperature parameter, β, will show up in the following way: A Markov Chain simulated at a single fixed $\beta \geq 0$ will have its stationary density given by:

$$\pi_{post,\beta}(\theta, c \mid \mathcal{D}) = \frac{\exp\left(-\beta \cdot \mathrm{E}(\theta, c)\right)}{Z(\beta)}, \tag{16}$$

where $\mathrm{E}(\theta, c)$ is defined by Eq. 15. The symbol β and the formalism we use here come from statistical physics, where: $\beta = (\mathrm{kT}^{-1})$, with, k, the Boltzmann constant and T, the absolute temperature.

What we will do in fact is simulating a Markov Chain on an extended space. If we call $\mathcal{P}$ the space on which Θ is defined (we already used $\mathcal{C}$ for the space on which C is defined) and if we use N_β different temperatures $((\beta_1, \ldots, \beta_{N_\beta}))$, then our new Markov Chain is defined on:

$$\prod_{p=1}^{N_\beta}(\mathcal{P} \times \mathcal{C}) \tag{17}$$

and its stationary density is given by:

$$\prod_{p=1}^{N_\beta} \pi_{post,\beta_p} \tag{18}$$

To get the statistics of interest we then just need to use the sub-chain at $\beta = 1$.

Last check before we start

In order to perform an *MCMC* based analysis we will have to call a stand alone C function `som_mcmc`. We first need to find out if the function is accessible from the directory where we are now running R. We can make this check as follows:

```
> system("som_mcmc --version")
```

4. Start and load data

This section describes the procedure to follow before running either the *EM* (section 5) or the *MCMC* (section 6) algorithm. After loading and reading the data into **R**, three steps have to be performed: *detecting spikes*, *making spike sweeps*, *reducing and whitening the spike sweeps*, as described in Pouzat *et al.* (2002).

4.1. Loading the SpikeOMatic Package in R

Once in R, if you want to use a specific package like **SpikeOMatic**, you must load it using:

```
> library(SpikeOMatic)
```

You can then read your data into the R environment.

4.2. Reading the data into R

The case we will illustrate in this paper comes from a linear probe recording (see Delescluse and Pouzat and Fig. 3 of Pouzat (2005) for information about the recording settings) from a young rat cerebellar slice. The four recording sites of the probe (which are 50 μm apart) were positioned along the Purkinje cell layer. We will therefore be dealing with 4 data files: `PK_1.fl.gz, PK_2.fl.gz, PK_3.fl.gz, PK_4.fl.gz`. On your machine they are located in folder: /usr/local/lib/R/library/SpikeOMatic/doc. 58 seconds of recording are used. Data were sampled at 15 kHz and filtered between 0.3 and 5 kHz. The data are compressed and in float format.

Loading the data can be done interactively by first calling function `select.data.files`:

```
> data.names <- select.data.files()
```

which makes a window pop up inviting you to select the data files that can then be read with `read.data.files` (see below).

It is also possible to manually build a character vector like the `data.names` above as follows (we are proceeding here in that way because we want this tutorial to be self generative):

```
> vigdir <- system.file("doc", package = "SpikeOMatic")
> data1.name <- paste(vigdir, "/PK_1.fl.gz", sep = "")
> data2.name <- paste(vigdir, "/PK_2.fl.gz", sep = "")
> data3.name <- paste(vigdir, "/PK_3.fl.gz", sep = "")
> data4.name <- paste(vigdir, "/PK_4.fl.gz", sep = "")
> data.names <- c(data1.name, data2.name, data3.name, data4.name)
```

After selecting the data in one or the other way, we can read them by calling `read.data.files`:

```
> data <- read.data.files(trace.names = data.names, trace.length = 3e+05,
+     size.of = 4)
```

The argument `trace.length` specifies the number of sample points you read from the files `data.names`. Argument `size.of` is the size of one sample point (in bytes).

We can visualize the data in an interactive way with `display.raw.data.gui`:

```
> display.raw.data.gui(data)
```

4.3. Detecting spikes

Spikes are detected with function `find.spikes.with.template`, which uses a self generated spike template or `find.spikes.no.template` which does not. We will use here the template based detection with a fairly large detection threshold to avoid having too many detected events (you are free to try other detection settings and you should do it). A template is first extracted by selecting at least hundred of the largest spikes on the trace with the largest standard deviation (SD). This template, T_j, is then used to filter each individual trace s_i as follows:

$$s_i^* = \sum_{k=1}^{J} s_{i+k-\frac{J}{2}} \cdot T_k \tag{19}$$

where J is the template length (forced to be even by the implementation). Events are detected on the filtered traces and the detection threshold is a multiple of the whole SD of the *filtered* traces.

```
> analyze <- find.spikes.with.template(trace.matrix = data, threshold = 5,
+     minimal.distance = 5)
```

This function creates an R object, called "analyze", which is a list with the following components:

- $ `find.spikes.call`: the actual call used to find the spikes.
- $ `raw.data`: a reference ("pointer") to the variable containing the raw data.
- $ `spike.found`: the number of detected spikes.

- $ `threshold`: the detection threshold used.
- $ `minimal.distance`: the minimal distance for 2 spikes detected on 2 different sites to be considered as different (if several recording sites are used simultaneously, see help page for details).
- $ `spike.pos`: a vector with the positions (in sample indices) of the detected spikes.
- $ `template`: a "non available" value with this type of spike detection.

This object will contain the full analysis performed on these data: all subsequent functions will store their respective outputs as new components added to this list, while using the already existing components to perform their task. Therefore, this object "analyze" will systematically be the first argument of every function. After each function call the command `str(analyse)` allows to visualize the current content of this object. At this stage it reads:

```
> str(analyze)

List of 7
 $ find.spikes.call: language find.spikes.with.template(trace.matrix = data, threshold = 5
 $ raw.data        : symbol data
 $ spike.found     : int 979
 $ threshold       : num 5
 $ minimal.distance: num 5
 $ spike.pos       : num [1:979]   97  224 1193 1281 1351 ...
 $ template        : num [1:60] -3.10e-03  7.62e-04 -2.38e-05 -2.45e-03 -2.71e-04 ...
```

In that case, 979 spikes are detected.

You should now check detection quality with function `display.detection.gui`:

```
> display.detection.gui(analyze)
```

which makes a GUI pop up allowing you to compare the raw trace and events detection.

4.4. Making spikes and noise sweeps

If detection is judged satisfying the next stage is the extraction of spikes and noise sweeps. That is, a sweep of a given length is cut around the detected spikes on the raw traces. The peak of the spikes is moreover placed at a user specified position within the sweeps. Then "noise sweeps" are extracted in between the spike sweeps. A single command does the whole job, `make.sweeps`:

```
> make.sweeps(spikes = analyze, sweep.length = 45, peak.location = 20,
+     nb.noise.evt = 2000)
```

Five components are added to the first input argument (`spikes`):

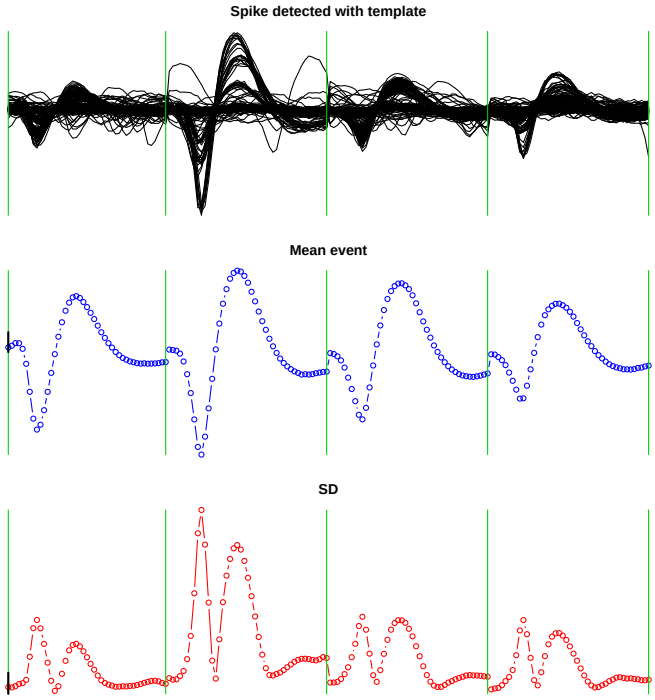

Figure 1: A quick display of the spike sweeps for spikes detected with template. 200 (that is the default value of the argument `nb.to.display`) of the 979 detected spikes are shown. The scale bar on the central panel (`Mean event`) goes from 0 to 1 in noise SD units. The scale bar on the lower panel (`SD`) goes from 1 to 2 in noise SD units.

- $ `sweep.length`: the length of the extracted sweeps, *on each recording site*, in sample points.
- $ `peak.location`: the location of the peak of the spike within the sweep.
- $ `nb.noise.evt`: the number of extracted noise sweeps.
- $ `spike.sweeps`: the actual spike sweeps, a matrix with as many rows as there are spikes and whose number of columns is given by: $sweep.length \cdot nb.recording.sites$.
- $ `noise.sweeps`: the noise sweeps, a matrix build like the `spike.sweeps` matrix.

As usual the `str()` command allows us to quickly visualize the list.

The spike sweeps we have just generated can be visualized with function `display.spike.sweeps`:

```
> display.spike.sweeps(spikes = analyze, main.top = "Spike detected with template")
```

This generates Fig. 1.

Function `make.sweeps` also extracts 2000 noise sweeps in between the spike sweeps. Estimates of the noise auto- and cross-correlation functions can be obtained and displayed with function `display.noise.correlation`:

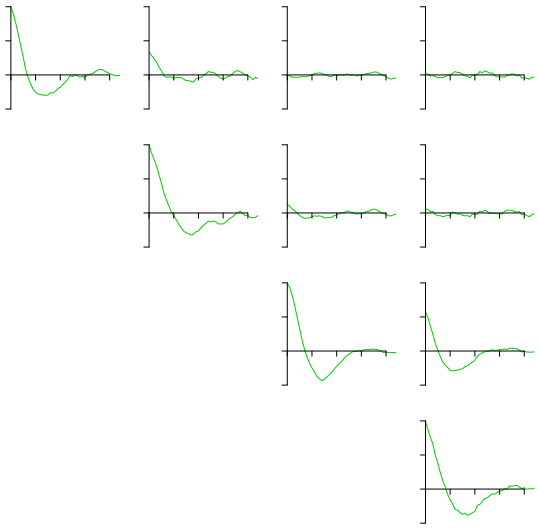

Figure 2: Noise auto- and cross-correlation functions. The auto-correlation functions on each recording site are displayed on the diagonal of the plot. The cross-correlation functions are displayed in an (hopefully obvious way) on the upper diagonal part.

```
> display.noise.correlation(noise = analyze)
```

The result is shown on Fig. 2.

4.5. Whitening

Now that spikes and noise sweeps have been obtained, the spikes sweeps can be reduced and whitened with the function `reduce.and.whiten`. By "reduced" we mean that few coordinates of the original *events space* will be selected based on the SD of the whole sample along these coordinates. The number of coordinates or *indices* to keep *per recording site* is specified by an argument: `nb.samples.per.site`, the selection of these indices can be done automatically (if argument `automatic` is set to `TRUE`) or manually (if argument `automatic` is set to `FALSE`). In the latter case a window with a plot of the whole sample SD pops up and the user is invited to click on the points of his choice. One does need to select points (indices) on a single site only, the corresponding points on the other recording sites will be automatically extracted.

A third possibility is to give the vector of indices of amplitude samples of each spike sweep that will be kept as an `automatic` argument. That is what we are doing here. We first specify this vector before calling `reduce.and.whiten`. In this analysis, we keep the peak amplitude on each recording site, *i.e* the 20th point of the spike sweep on each site (recall that the length of a spike sweep on one site is 45 points):

```
> selected.points <- c(20, 65, 110, 155)
> reduce.and.whiten(spike.list = analyze, automatic = selected.points,
+     nb.samples.per.site = 1)
```

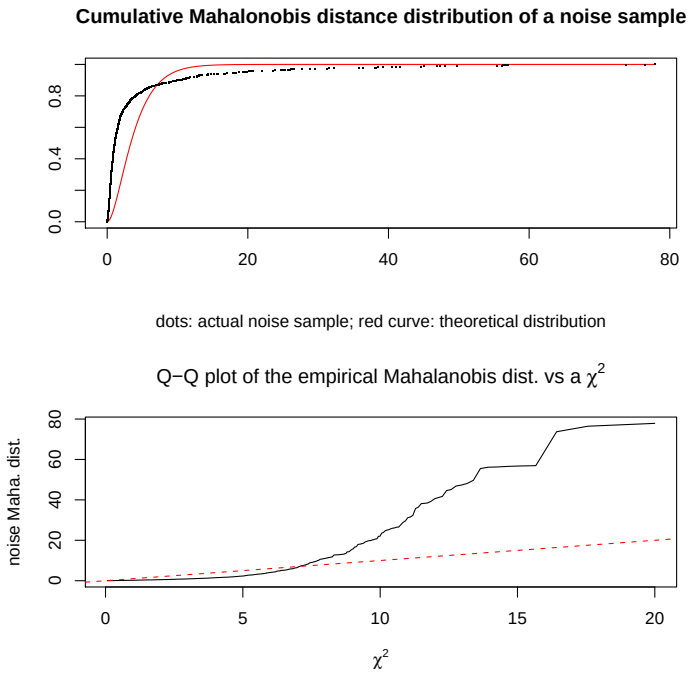

Figure 3: Comparison between theoretical and empirical noise properties. The upper panel shows the cumulative distribution of the *Mahalanobis* distances of an actual noise sample (reduced and whitened), in black, and the expected χ^2 distribution, in red. The lower panel is a tougher test of the difference between the two distributions. The presence of more large *Mahalanobis* distances (upper bend of the black curve) than expected is mainly due to the presence of sub- detection threshold spikes.

Three new components are added to the *list* `analyze`:

- $ `selected.indices`: a one dimensional array with the selected indices.
- $ `spike.sweeps.white`: a matrix with reduced and whitened spike sweeps.
- $ `noise.mahalanobis`: a one dimensional array with the *Mahalanobis* distances (to the origin of the *reduced events space*) of a sub-sample (half) of the reduced and whitened noise sweeps.

`reduce.and.whiten` not only reduces and whitens the spike sweeps but generates also diagnostic plots as shown on Fig. 3.

A Wilson plot of the reduced and whitened sample can be generated with `display.wilson` as shown on Fig. 4. On these Wilson plots the spike amplitude on one recording site is plotted against the spike amplitude on another site.

```
> display.wilson(analyze, , , , main = "Wilson plot with noise whitening")
```

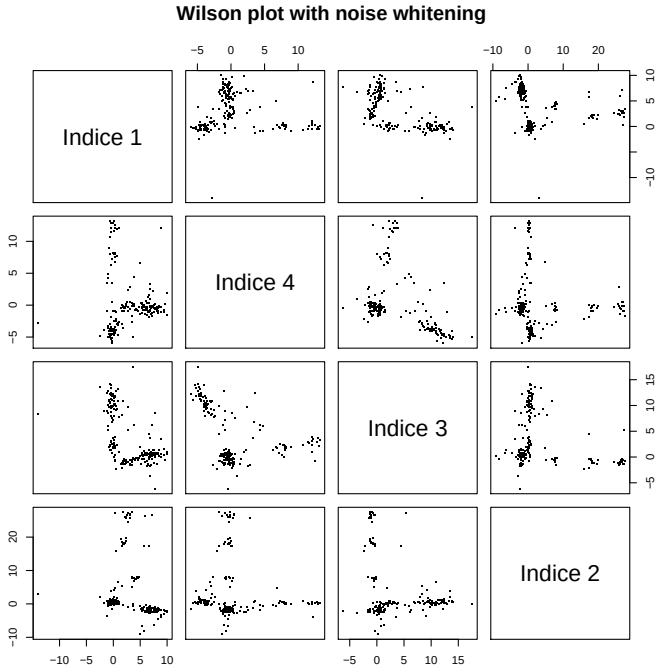

Figure 4: A new "view" on the spike sample: a Wilson plot after *events space* reduction and noise whitening.

5. Gaussian mixture based spike-sorting

Once the spike sweeps are reduced and whitened, the spike-sorting can be performed with the Expectation-Maximization algorithm for Gaussian mixture models, *GMM*, or with the Markov Chain Monte Carlo algorithm for the Dynamic Hidden Markov Model, *DHMM*. We start with the *GMM* procedure.

5.1. Model estimation

The model estimation is done by function `get.model`. We will force here `get.model` to work with a model made of 6 neurons. You are of course encouraged to try out models with more or less neurons, using the `nb.clusters.min` and `nb.clusters.max` arguments. To compare models, we use the Bayesian Information Criterion (`BIC`): the model that has to be chosen is the one minimizing this criterion (Schwarz (1978)). In the present case, if you let `get.model` run with different numbers of neurons, you will see that the model minimizing the BIC has 28 neurons, which is much greater that the real number that one can roughly estimate to be smaller than 10. That is due to the very elongated clouds of some neurons and to the multiple *distinct* clouds generated by other neurons (as can be guessed for Fig. 4). In this case this criterion is overestimating the number of neurons present in the data.

```
> em.model <- get.model(spike.list = analyze, tolerance.factor = 5,
+     nb.clusters.min = 6, nb.clusters.max = 6, nb.iterations = 50,
+     nb.tries = 30)

86  spikes were rejected on a total of  979 .
[1] ***********************************************
[1] Exploring now a model with 6
[3] clusters
[1] The best BIC value obtained with this model is:
[1]                   17640.4635737601
[1] ***********************************************
```

Function `get.model` generates 3 graphs that we do not reproduce here. The first one displays the "clean" sub-sample of the spike sweeps which is automatically generated using the `tolerance.factor` argument of `get.model`. This "clean" sub-sample -and not the whole sample- is used to estimate the model more reliably. The second graph provides a diagnostic of the EM output, that is the log-likelihood evolution of the best trial (to ensure that the algorithm has converged), as well as the BIC value of the best trial for each model explored (one model only in the current case). This allows the user to choose the model minimizing the BIC. The third plot shows the locations of the cluster centers on a Wilson plot.

The output object of `get.model` contains all the information relevant to the model as well as information necessary to detect spikes, extract sweeps and reduce *events space* dimension on a subsequent acquisition epoch:

- $ `find.spikes.call`: a copy of the corresponding argument of `analyze`, see section 4.3 on page 9.
- $ `raw.data`: a copy of the corresponding argument of `analyze`, see section 4.3 on page 9.

- $ `threshold`: a copy of the corresponding argument of `analyze`, see section 4.3 on page 10.
- $ `minimal.distance`: a copy of the corresponding argument of `analyze`, see section 4.3 on page 10.
- $ `sweep.length`: a copy of the corresponding argument of `analyze`, see section 4.4 on page 11.
- $ `peak.location`: a copy of the corresponding argument of `analyze`, see section 4.4 on page 11.
- $ `nb.noise.evt`: a copy of the corresponding argument of `analyze`, see section 4.4 on page 11.
- $ `selected.indices`: a copy of the corresponding argument of `analyze`, see section 4.5 on page 13.
- $ `nb.clusters`: the best number of clusters (neurons), given the data (in the *BIC* sense).
- $ `nb.cluster.range`: the range of number of clusters explored.
- $ `nb.trials`: the number of different initial guesses used for each model (*i.e.*, for each number of neurons considered).
- $ `bic`: a vector with the best *BIC* value obtained for each model considered.
- $ `centers`: a matrix with the clusters centers in the reduced and whitened space. The matrix has as many rows as there are neurons in the model (`nb.clusters`) and as many columns as there are dimensions in the reduced space.
- $ `frequency`: the frequency associated with each neuron (cluster) of the model.
- $ `log.likelihood`: the log likelihood sequence for the EM run which ended up giving the best *BIC* value.
- $ `tolerance.factor`: the factor used to extract the *clean sample* from which model estimation was performed. In short, the spikes of the original sample, `analyze$spike.sweeps` (in the original space), were kept only if they fell within $+/-tolerance.factor \cdot spike.sweeps.sd$ of the average spike sweep. This is an easy way to get rid of the most obvious superpositions of spikes.
- $ `full.mean`: a matrix with the clusters centers in the original space. The matrix has as many rows as there are neurons in the model (`nb.clusters`) and as many columns as there are dimensions in the original space.

As usual, you can see the content of this list by typing `str(em.model)`.

5.2. Spikes Classification

The model being estimated, we can proceed to the spikes classification with function `classify.from.mixture`.

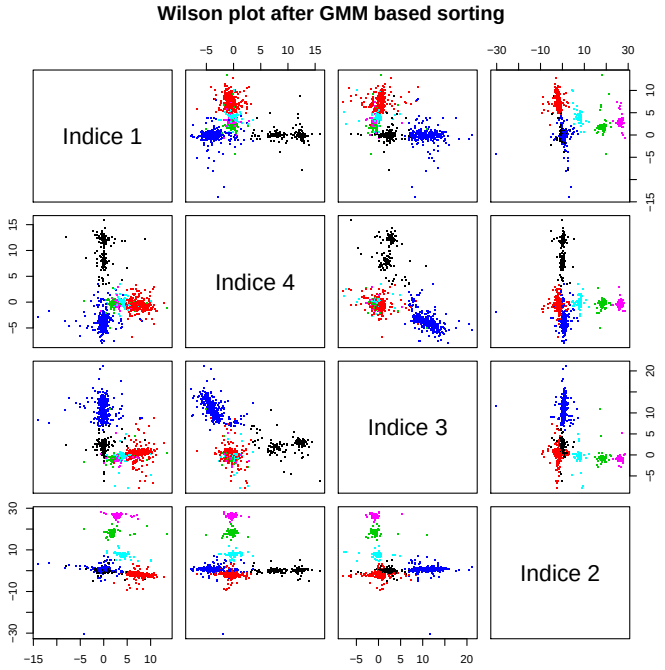

Figure 5: The Wilson plot of Fig. 4 after spike-sorting with an EM estimation of a *GMM* with 6 neurons.

```
> classify.from.mixture(spike.list = analyze, model.list = em.model)
```

This modifies `analyze` by adding two components to it:

- $ `cluster`: a one dimensional array with the "neuron" of origin of each spike, which is what we are essentially interested in.
- $ `spike.mahalanobis`: a one dimensional array with the *Mahalanobis* distance between each spike and its closest neuron (that is, its neuron of origin).

We can now check out the sorting results on a Wilson plot Fig. 5 where the spikes are color coded according to their respective labels:

```
> display.wilson(analyze, , sub.sample.size = length(analyze$spike.pos),
+     , main = "Wilson plot after GMM based sorting")
```

We can also look at each individual neuron with the function `display.neuron`:

```
> display.neuron(spike.list = analyze,
                 neuron.index = 1,
                 sampling.rate = 15000,
                 mahalanobis.threshold = 50)
```

This generates two series of plots (not shown). The first series is made of a plot where all the spikes attributed to neuron 1 (and whose distance to the cluster center - Mahalanobis distance - is less than 50) are superimposed, along with plots of the neuron's mean spike and of its standard deviation. The second series shows the Q-Q plot (like in figure 3) for this particular neuron, as well as its *ISI* histogram.

6. DHMM based spike-sorting

We now describe how to proceed to perform the *DHMM* based spike-sorting with the *MCMC* algorithm (Pouzat *et al.* (2004)). What follows can be done immediately after whitening (section 4.5), without performing a previous *GMM* based spike-sorting as we did here. Running the *MCMC* algorithm requires to build its input files first.

6.1. Building the MCMC inputs

Our *MCMC* algorithm, like any *MCMC* algorithm is iterative, which means it has to start from somewhere. It requires an initial state for the Markov chain to be defined. We can use a *GMM* based sorting to get initial guesses for the *DHMM* based sorting, or we can use the more direct, simpler and as it turns out much more efficient, "infinite temperature initialization", with a random initial state.

This necessary intitialization of the Markov chain is done by calling function `pre.mcmc`. If its argument `use.cluster` is set to `FALSE`, the random initialization is performed. That is what we are doing below. On the other hand, if `use.cluster` is set to `TRUE` this function computes an intial state for the Markov chain from the classification performed with the *EM* algorithm.

```
> pre.model <- pre.mcmc(spike.list = analyze, generic.input.name = "analyze",
+     nb.states = 2, sampling.frequency = 15000, use.cluster = FALSE,
+     nb.neurons = 6)
```

The argument `nb.states` specifies the number of neurons' discharge states (see section 3.2) to be used . The arguments `sampling.frequency` and `nb.neurons` respectively specify the sampling frequency of the recording and the number of neurons to be used for the *MCMC* algorithm.

To store the initial state of the Markov chain, function `pre.mcmc` has created three files, that will be given as inputs to the *MCMC* algorithm in the next step. The names of these files all begin with the string contained in argument `generic.input.name`:

- `analyze.configuration_1`, storing the configuration of the initial state.
- `analyze.neurons_1`, storing the model parameters of the initial state.
- `analyze.nrj_1`, storing the energy value of the initial state.

The model of this initial random Markov chain state is also stored in the R object `pre.model`.

Function `pre.mcmc` has also created another four input files required by the *MCMC* algorithm. They all begin with the string contained in argument `generic.input.name`:

- `analyze.spike_train`: the spike train to be sorted, built with vector `$spike.pos` and matrix `$spike.sweeps.white` of the R object `spike.list`. This is the data file used by the *MCMC* algorithm.
- `analyze.data_features`: number of spikes and number of points per spike sweep.
- `analyze.model_features`: number of neurons and number of states per neuron in the model.
- `analyze.priors`: the allowed ranges for each model parameter.

Function `pre.mcmc` has also added the following fields to the R object `analyze`:

`$sampling.frequency`: the sampling frequency.

`$start.time`: the beginning of the recording (in seconds).

`$stop.time`: the end of the recording (in seconds).

`$generic.input.name`: the generic name to store the input files of the *MCMC* algorithm.

It is possible to choose the priors of the model parameters by specifying the argument `priors` of function `pre.mcmc`. This argument is a list containing two components for each model parameter for its maximal and minimal values allowed. The priors are flat between these limits. By omitting this argument, the default values are used (this is what we did here). The priors are also stored in the R object `pre.model`, in the created field `priors`.

The file `analyze.spike_train`, as well as the files `analyze.data_features`, `analyze.model_features`, `analyze.priors` and the initialization files `analyze.configuration_1`, `analyze.neurons_1`, `analyze.nrj_1` can now be used as input for the *MCMC* algorithm.

6.2. Running the MCMC algorithm

We are ready to run the *MCMC* algorithm built with the **SpikeOMatic Library**. The R function `mcmc` starts up the algorithm. The function is called with the arguments described below. If some or all of the arguments are missing, windows will successively pop up to ask for each missing argument. In the example below we first define a vector of inverse temperatures, `beta.vector`, at which we want to run the algorithm. Setting the inverse temperatures, βs, is a bit tricky. You want to get a high enough temperature (a low enough β) to overcome the most common energy barriers in your system (one problem here is that you do not know *a priori* how high you should go). Then you want to have enough βs in between the most interesting one, $\beta = 1$, and the smallest one to see your replica traveling in the temperature space. For that you want to have overlap of the energy distributions of consecutive β values (see Pouzat *et al.* (2004) and Pouzat (2005)). But you do not want to have too many values neither in order to keep your running time as small as possible.

```
> beta.vector <- c(1, 0.95, 0.9, 0.875, 0.87, 0.865, 0.86, 0.855,
+     0.85)
> mcmc(spike.list = analyze, betas = beta.vector, number.of.steps = 1000,
+     generic.output.name = "analyze", algorithm.options = FALSE)
```

```
Spike Train File:  analyze.spike_train

Data Features File:  analyze.data_features
Model Features File:  analyze.model_features
Priors File:  analyze.priors
Betas File:  analyze.betas
Number of MCMC steps:  1000
Generic Output Name:  analyze

Initial State Files Generic Names:  analyze.configuration_ , analyze.neurons_  and  analyz


The output files are:
 analyze.configuration_betaValue
 analyze.neurons_betaValue
 analyze.nrj_betaValue
```

Here is a description of every argument of function `mcmc`:

- `spike list`: the R object in which the complete analysis we are achieving is being stored.
- `betas`: a vector containing the different βs (*i.e.*, inverse temperatures, at which the *MCMC* algorithm will be performed, in a decreasing order.
- `number.of.steps`: the number of *MCMC* steps to perform.
- `generic.output.name`: three files are created by the algorithm to store its output. Their names start with the string specified by this argument and ends with different suffices: `.configuration`, `.neurons` and `.nrj` (see below, section 6.3).
- `algorithm.options`: different dynamics are possible for the algorithm. By setting this argument to FALSE or by omitting it, these options are set to their default value that can be used for a start. If `algorithm.options` is set to TRUE a window pops up, allowing the user to tune the dynamics as he wants.

`mcmc` creates a file `analyze.betas` where the βs given in argument are written.

Three more fields corresponding to the arguments of this function are added to the R object `analyze`:

`$generic.output.name`: the generic name to store the output of the *MCMC* algorithm.

`$betas`: a one dimensional array containing the βs at which the algorithm will be performed.

`$number.of.steps`: a 2-element vector containing the total number of steps performed with this very same R object so far and the number of steps of the upcoming run.

This R function calls the C function from the **SpikeOMatic Library** that runs the algorithm with the inputs that have just been created. In our case, the function calls:

```
> system("som_mcmc -d analyze.data_features -m analyze.model_features -t analyze.spike_tra
-p analyze.priors -b analyze.betas -n 1000 -u 6 -o analyze -l analyze")
```

where the files `analyze.data_features`, `analyze.model_features`, `analyze.priors` and the file `analyze.spike_train` are the files created in section 6.1 and the file `analyze.betas` has just been created. The values of options `-n`, `-u` and `-o` are the arguments of the R function. The last option `-l` contains the generic name of the files `.configuration_1`, `.neurons_1` and `.nrj_1`, whose last state will be the initial state of the present *MCMC* run. This generic name is the string stored in the field `$generic.input.name` of the object `analyze`. It has been specified by the `generic.input.name` argument of function `pre.mcmc` and these files have been created when calling this function (section 6.1).

The standard output summarizes all the given inputs and reminds the user of the output file names that are now available for analysis. It also prints out continuously how the algorithm is progressing.

6.3. Output files

Three files are created to store the output of the algorithm *for each inverse temperature* (β) used. They all begin with the generic output name given in argument of the R function `mcmc`. They have different suffices:

- `.configuration`: a configuration is defined by giving both a neuron of origin (a label) *and* a state to each spike of the spike train (see section 3.3). The length of a configuration is thus equal to 2 times the total number of spikes in the train. The configurations of all steps are written as *unsigned char* one after the other in this file.
- `.neurons`: this file contains all the parameters of all the neurons at each step.
- `.nrj`: this file contains the value of the energy of the spike train at each step, given its configuration and the parameters' values.

Note that each of these suffices ends with the string `"_betaValue"` where `betaValue` is the value of β at which the run has been performed. If the output name is the same as the input name, the program writes the new output *at the end* of the existing files, without removing their content. This allows to perform successive *MCMC* runs, starting from the last state reached by the Markov chain in the immediately preceding run and storing the new run at the end of it, as explained in the next section.

6.4. Performing successive MCMC runs

As mentioned above, function `mcmc` starts the Markov chain at $\beta = 1$ with the last state stored in the files `.configuration_1`, `.neurons_1` and `.nrj_1`. The generic name of these files is given by the field `$generic.input.name` of the object `analyze` containing the whole analysis. This generic name is initially specified by the argument `generic.input.name` of function `pre.mcmc` (see section 6.1). There are two possibilities:

No MCMC run has been performed before. These files contain only the state computed by function `pre.mcmc`. This state is either random (the argument `use.cluster`

of this function is set to `FALSE`) or computed from the classification done with the `EM` algorithm (argument `use.cluster` set to `TRUE`).

One, or several, MCMC run have been performed before. These files contain the successive states of the Markov chain previously simulated. The state initializing the Markov chain of the new run is the last state in these files.

If we set the argument `generic.output.name` of function `mcmc` to the same string as `generic.intput.name`, the successive states of the new Markov chain will be written at the end of these files.

In the case where one or several *MCMC* runs have already been performed at different βs, the new Markov chain at each β is initialized with the last state reached at this β by the preceding run(s) (files `.configuration_b`, `.neurons_b` and `.nrj_b`, where "b" stands for one of the βs already used). It is of course possible to start a new run with additional βs, *i.e* βs that were not used in the preceding run(s). In this case, the initial state of the Markov chain at any additional β is the initial state of the closest greater β.

6.5. Reading the output files

Three functions allow to read the three output files of function `mcmc`, one for each output file: `display.energy.evolution`, `read.configuration` and `read.neurons`.

The first thing to do immediately after running the *MCMC* algorithm is to check for convergence and decide from where the *MCMC* sample should be used or if more *MCMC* steps are required. In particular, one has to choose the first *MCMC* step from where statistics will be performed. Plotting the evolution of the energy at a given temperature enables the user to decide if equilibrium has been reached or not and if so, when it has been reached. If a single inverse temperature was used, $\beta = 1$, it does not hurt to make an additional short run with smaller values as well, just to be surer the Markov Chain did not get trapped in a local energy minimum. Then function `display.energy.evolution` will allow you to visualize the evolution of the energy (or energies). What you want to see is a stable mean value for the trace(s). If several βs were used you want to see also overlaps between the energies at neighboring βs.

Reading the energy file

The function `display.energy.evolution`, reads `.nrj` files and plots them. Each `.nrj` file contains the successive values of the energy of the spike train at each *MCMC* step at a given inverse temperature, so that `display.energy.evolution` displays the evolution of the energy at given inverse temperatures. If some or all of the following arguments are missing, windows will successively pop up to ask for each missing argument.

```
> display.energy.evolution(spike.list = analyze, beta = analyze$betas)
```

where `analyze` is the R object containing the full analysis performed so far and `beta` is a vector containing the inverse temperatures whose energy evolutions are going to be plotted.

The function `display.energy.evolution` adds two components to the spike.list `analyze`:

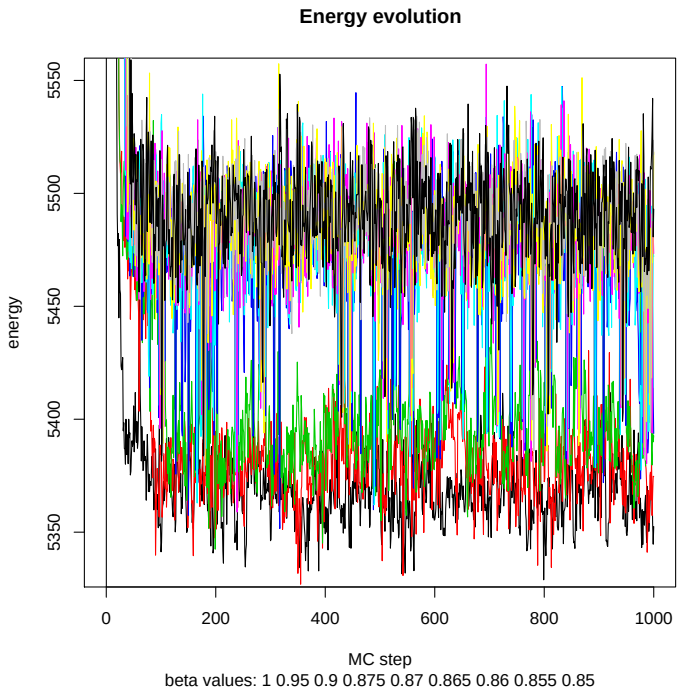

Figure 6: The energy of the Markov chain at all betas.

`$nrj.trace`: a list of one dimensional arrays, each of them containing the energy value of the whole spike train at each *MCMC* step at the corresponding β. The list has one component *per* β.

`$nrj.offset`: a list of scalars, each of them being the value of the first *MCMC* step at which the run began at the corresponding β. This allows to read and display energy vector of different lengths if the algorithm has been run at different βs with different total number of steps. The list has one component *per* β.

If the energy does not seem to have reached steady-state or if we want to be on the safe side we can run 1000 more iterations starting from where we stopped.

```
> mcmc(spike.list = analyze, betas = beta.vector, number.of.steps = 1000,
+     generic.output.name = "analyze", algorithm.options = FALSE)

Spike Train File:  analyze.spike_train

Data Features File:  analyze.data_features
Model Features File:  analyze.model_features
Priors File:  analyze.priors
Betas File:  analyze.betas
Number of MCMC steps:  1000
```

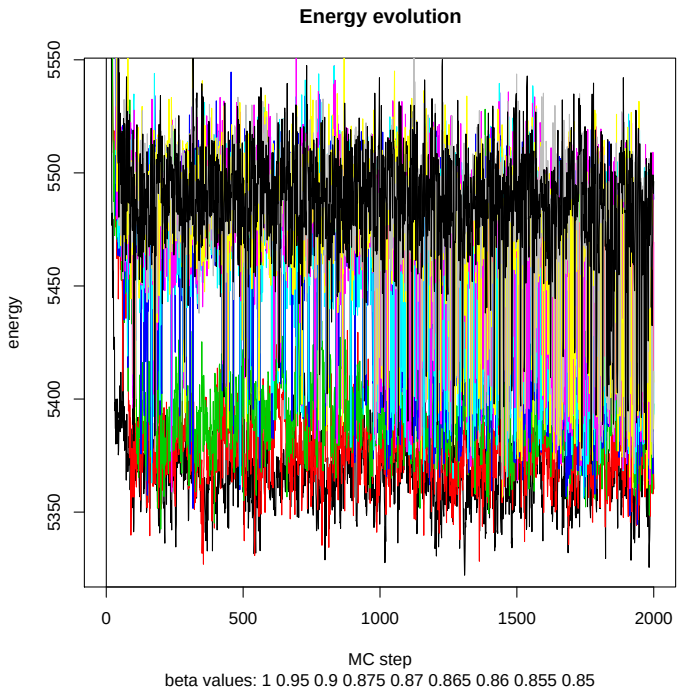

Figure 7: The energy of the Markov chain at all betas after 1000 more iterations.

```
Generic Output Name:  analyze

Initial State Files Generic Names:  analyze.configuration_ , analyze.neurons_  and  analyz


The output files are:
 analyze.configuration_betaValue
 analyze.neurons_betaValue
 analyze.nrj_betaValue
```

We can look at the energy evolution by calling again function `display.energy.evolution`, the result is shown on Fig. 7.

```
> display.energy.evolution(spike.list = analyze, beta = analyze$betas)
```

It very instructive as well to look at the energy histograms (to make sure that they overlap). Function `display.energy.histo` is made for that. It is called as follows:

```
> display.energy.histo(analyze, nb.bins = 25, start.measurements = 1000)
```

You can see on Fig. 8 that the energy distributions at neighboring βs do indeed overlap. You can see as well an energy domain where the system is almost never found (that is at some β

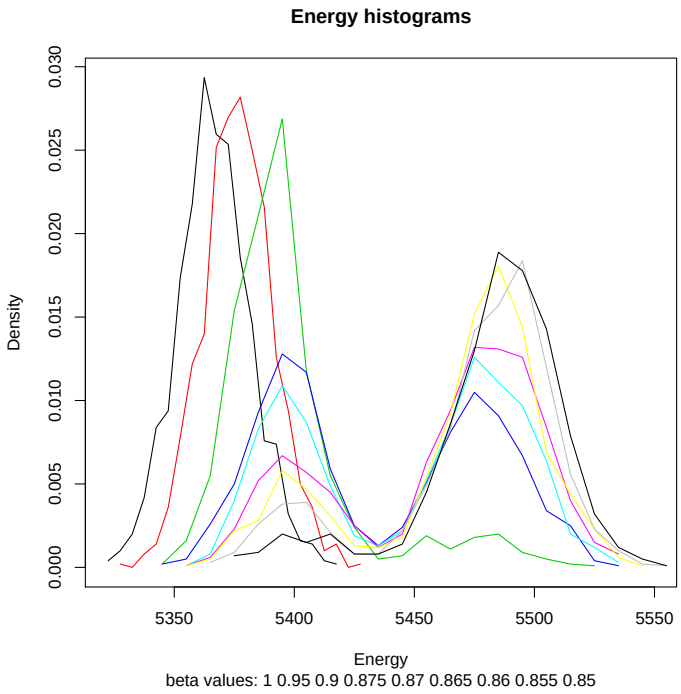

Figure 8: Energy distributions at different β values.

the energy distribution becomes bi-modal). You can try, as an exercise to figure out what's going on here... Don't worry we know.

We make another 1000 steps at $\beta = 1$ only, to compute model averages of the parameters at that *beta* (as described in the very last paragraph):

```
> mcmc(spike.list = analyze, betas = 1, number.of.steps = 1000,
+     generic.output.name = "analyze", algorithm.options = FALSE)

Spike Train File:  analyze.spike_train

Data Features File:  analyze.data_features
Model Features File:  analyze.model_features
Priors File:  analyze.priors
Betas File:  analyze.betas
Number of MCMC steps:  1000
Generic Output Name:  analyze

Initial State Files Generic Names:  analyze.configuration_ , analyze.neurons_  and  analyz


The output files are:
```

```
analyze.configuration_betaValue
analyze.neurons_betaValue
analyze.nrj_betaValue
```

Reading the configuration file

The function `read.configuration`, reads the `.configuration` file at a given inverse temperature β. It computes the most probable configuration of the spike train, both in terms of neuron of origin and in terms of neuron state. If some or all of the following arguments are missing, windows will successively pop up to ask for each missing argument.

```
> labeling <- read.configuration(spike.list = analyze, firstStep = 800, beta = 1)

Read Configuration:
Three new fields have been added to the spike.list you entered:
$cluster: a one dimensional array with the 'neuron' of origin of each spike.
$state: a one dimensional array with the 'state' of each spike.
```

where `analyze` is the R object containing the full analysis performed so far, `firstStep` is the value of the first *MCMC* step to consider to compute the most probable configuration and `beta` is the inverse temperature that we want to study among the available ones. Make sure that `firstStep` is smaller that the total number of steps performed.

After a short while, two new elements are added to the spike.list `analyze`, as printed out by the function itself:

`$cluster`: a one dimensional array with the "neuron" of origin of each spike.

`$state`: a one dimensional array with the "state" of each spike.

As usual, you can view the current content of our R object `analyze` by typing `str(analyze)`.

Finally the function returns a matrix, called `labeling` here, which contains for every detected spike (in columns) the number of MC-steps that assigned it to a given neuron (in lines) among the MC-steps considered to compute the configuration. That is, for spike number k (column k), line number i shows the number of MC-steps that assigned it to neuron i.

We can now plot the most probable configuration with `display.events`. If some or all of the arguments are missing, windows will successively pop up to ask for each missing argument.

```
> large.on.3 <- 1
> mean.amp.large.on.3 <- mean(analyze$spike.sweeps.white[analyze$cluster ==
+     1, 3])
> for (index in 2:6) {
+     new.mean <- mean(analyze$spike.sweeps.white[analyze$cluster ==
+         index, 3])
+     if (new.mean > mean.amp.large.on.3) {
+         large.on.3 <- index
+         mean.amp.large.on.3 <- new.mean
```

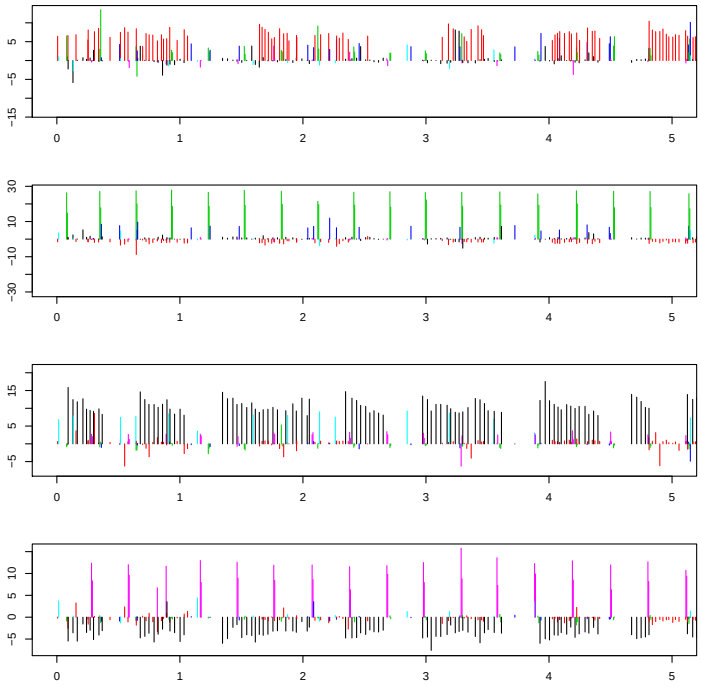

Figure 9: The spikes of all neurons are plotted - between the chosen time limits - and colored according to their most probable neurons of origin.

```
+     }
+ }
> display.events(spike.list = analyze, left = 0, right = 5, chosen.neuron = large.on.3)
```

Two windows will pop up. In the first one, the spikes of all neurons are plotted - between the chosen time limits `left` and `right` - and colored according to their most probable neurons of origin (Fig. 9). This gives an idea of the classification performed by the algorithm.

In the second window, the spikes of neuron index `large.on.3` (that is neuron 1) are plotted - between the chosen time limits - and colored according to their most probable states (Fig. 10). The three lines of code before the call for function `display.events` just find the index of the neuron which is large on site 3.

We can also see the Wilson plot after *MCMC* spike-sorting by calling again `display.wilson` to get Fig. 11.

```
> display.wilson(analyze, , sub.sample.size = length(analyze$spike.pos),
+     , main = "Wilson plot after DHMM based sorting")
```

Reading the neurons file

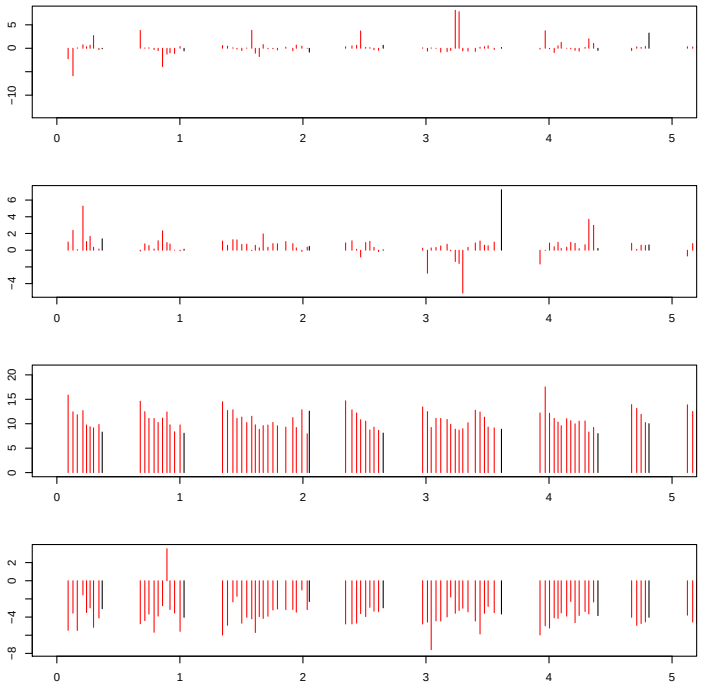

Figure 10: The spikes of the chosen neuron (here neuron # 1) are plotted - between the chosen time limits - and colored according to their most probable states. Not that the l̈ongs̈tate is in black, the ïntermediates̈tate is in blue and the s̈hortstate in green.

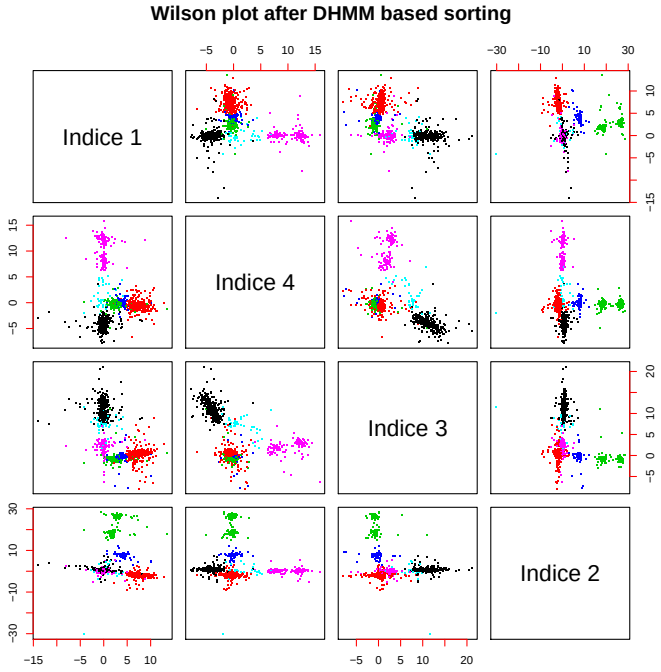

Figure 11: The Wilson plot of Fig. 5 after *DHMM* based spike-sorting. Can you see the difference on the elongated cluster along the axis of index 3.

The third output file, `.neurons` can be read with `read.neurons`. If some or all of the arguments are missing, windows will successively pop up to ask for each missing argument.

```
> read.neurons(spike.list = analyze, beta = 1)

Read Neurons:
A new field has been added to the spike.list you entered: $neurons.
This is a matrix containing the value of each parameter (column) at each step (line).
This matrix is now ready for analysis with BOA
```

After a short while, a new element is added to the spike.list `analyze`, as printed out by the function:

`$neurons`: a matrix containing the value of each parameter (column) at each *MCMC* step (line). That is, this matrix has as many rows as there are *MCMC* steps and as many columns as there are model parameters.

Each column of this matrix `spike.list$neurons` has the name of its model parameter, beginning with its neuron number like for instance "n1_" for parameters of neuron 1. Calling `colnames(analyze.$neurons)` prints out the names of all columns. The name of each row of the matrix is the *MCMC* step number. This matrix can be analysed with the R package "Bayesian Output Analysis" (BOA). For that purpose, we just need to assign it to an independent R object:

```
> neurons <- analyze$neurons
```

The matrix `neurons` can be imported into BOA for analysis. The user is referred to the BOA user's manual by Brian Smith[1] for the easy use of this package.

Now, our R object `analyze` contains the full analysis performed on the raw data that has been analyzed, as shown by the final call:

```
> str(analyze)

List of 27
 $ find.spikes.call   : language find.spikes.with.template(trace.matrix = data, threshold
 $ raw.data           : symbol data
 $ spike.found        : int 979
 $ threshold          : num 5
 $ minimal.distance   : num 5
 $ spike.pos          : num [1:979]   97  224 1193 1281 1351 ...
 $ template           : num [1:60] -3.10e-03  7.62e-04 -2.38e-05 -2.45e-03 -2.71e-04 ...
 $ sweep.length       : num 45
 $ peak.location      : num 20
 $ nb.noise.evt       : num 2000
 $ spike.sweeps       : num [1:979, 1:180]  0.6604 -0.5235 -0.0993  3.0030  0.3312 ...
```

[1] http://www.public-health.uiowa.edu/boa/Home.html

```
$ noise.sweeps       : num [1:2000, 1:180]  0.078 -0.055 -0.163 -3.265 -1.138 ...
$ selected.indices   : num [1:4] 20 65 110 155
$ spike.sweeps.white : num [1:979, 1:4] 6.44 1.03 6.51 1.01 6.64 ...
$ noise.mahalanobis  : num [1:1000]  2.033  2.503 77.861  1.620  0.581 ...
$ sampling.frequency : num 15000
$ start.time         : num 0
$ stop.time          : num 20
$ generic.input.name : chr "analyze"
$ generic.output.name: chr "analyze"
$ betas              : num 1
$ number.of.steps    : num [1:2] 3001 1000
$ nrj.trace          : NULL
$ nrj.offset         : NULL
$ cluster            : num [1:979] 2 5 3 3 2 1 1 5 2 1 ...
$ state              : num [1:979] 2 2 1 2 1 2 2 2 1 2 ...
$ neurons            : num [1:3001, 1:84] -0.0735  1.0307  0.0208  0.0208 -0.3494 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : chr [1:3001] "1" "2" "3" "4" ...
 .. ..$ : chr [1:84] "n1_P1" "n1_P2" "n1_P3" "n1_P4" ...
```

Finally, we can compute the mean value of each model parameter for $beta = 1$, after equilibrium has been reached (see the energy plot, figure 6) and build the *MCMC* model. This is done by `get.mcmc.model.para`:

```
> mcmc.model <- get.mcmc.model.para(spike.list = analyze, firstStep = 2200)
```

where `analyze` is the R object containing the full analysis performed so far, `firstStep` is the value of the first *MCMC* step to consider to compute the average value of each parameter. Make sure that `firstStep` is smaller than the total number of steps performed.

`get.mcmc.model.para` creates the same R object, named here `mcmc.model`, as does `pre.mcmc`. This object is a list of lists. There is one component *per* neuron and a "priors" component. Both are described below. Each neuron component contains all its parameters set to their average values computed over the MC-steps between step index `firstStep` and last step.

The neuron lists have the following components:

- `neuron.type`: the neuron type. For now it's always "log-normal".
- `neuron.states`: the number of states of the neurons.
- `peak.max`: the vector of maximal peak amplitudes.
- `delta`: the value of parameter delta.
- `lambda`: the value of parameter lambda.
- `scale`: a vector of scale values.
- `shape`: a vector of shape values.

- `transition`: a matrix of inter-state transition probabilities.

The `priors` component is a list containing the minimal and maximal values of each model parameter:

- P.max.max
- P.max.min
- delta.max
- delta.min
- lambda.max
- lambda.min
- scale.max
- scale.min
- shape.max
- shape.min

As shown here:

```
> str(mcmc.model)

List of 7
 $ neuron1:List of 8
  ..$ neuron.type  : int 1
  ..$ neuron.states: int 2
  ..$ peak.max     : num [1:4] -0.220  0.895 13.985 -5.073
  ..$ delta        : num 0.601
  ..$ lambda       : num 27.7
  ..$ scale        : num [1:2] 0.3037 0.0361
  ..$ shape        : num [1:2] 0.126 0.302
  ..$ transition   : num [1:4] 0.0276 0.9724 0.1440 0.8560
 $ neuron2:List of 8
  ..$ neuron.type  : int 1
  ..$ neuron.states: int 2
  ..$ peak.max     : num [1:4]  9.098 -2.429  0.539 -0.617
  ..$ delta        : num 0.240
  ..$ lambda       : num 4.12
  ..$ scale        : num [1:2] 0.0519 0.0255
  ..$ shape        : num [1:2] 1.508 0.445
  ..$ transition   : num [1:4] 0.409 0.591 0.286 0.714
 $ neuron3:List of 8
  ..$ neuron.type  : int 1
```

```
 ..$ neuron.states: int 2
 ..$ peak.max     : num [1:4]  3.098 26.365 -0.862 -0.375
 ..$ delta        : num 0.966
 ..$ lambda       : num 178
 ..$ scale        : num [1:2] 0.00652 0.32996
 ..$ shape        : num [1:2] 0.133 0.131
 ..$ transition   : num [1:4] 0.0160 0.9840 0.9833 0.0167
$ neuron4:List of 8
 ..$ neuron.type  : int 1
 ..$ neuron.states: int 2
 ..$ peak.max     : num [1:4]  4.1667  7.5998 -0.4702 -0.0785
 ..$ delta        : num 0.383
 ..$ lambda       : num 613
 ..$ scale        : num [1:2] 0.113 0.253
 ..$ shape        : num [1:2] 1.708 0.237
 ..$ transition   : num [1:4] 0.8070 0.1930 0.0477 0.9523
$ neuron5:List of 8
 ..$ neuron.type  : int 1
 ..$ neuron.states: int 2
 ..$ peak.max     : num [1:4]  0.0878 -0.3940  7.3655  1.2011
 ..$ delta        : num 0.444
 ..$ lambda       : num 526
 ..$ scale        : num [1:2] 0.195 0.328
 ..$ shape        : num [1:2] 1.35 1.08
 ..$ transition   : num [1:4] 0.475 0.525 0.256 0.744
$ neuron6:List of 8
 ..$ neuron.type  : int 1
 ..$ neuron.states: int 2
 ..$ peak.max     : num [1:4]  0.0366  0.1292  3.2222 15.7415
 ..$ delta        : num 0.492
 ..$ lambda       : num 2.62
 ..$ scale        : num [1:2] 0.00612 0.28459
 ..$ shape        : num [1:2] 0.136 0.290
 ..$ transition   : num [1:4] 0.0315 0.9685 0.8982 0.1018
$ priors :List of 10
 ..$ P.max.max : num 40
 ..$ P.max.min : num -40
 ..$ delta.max : num 2
 ..$ delta.min : num -1
 ..$ lambda.max: num 1000
 ..$ lambda.min: num 0.01
 ..$ scale.max : num 2
 ..$ scale.min : num 0.001
 ..$ shape.max : num 3
 ..$ shape.min : num 0.05
```

This model can be viewed with function `display.model`. This enables the user to visualize

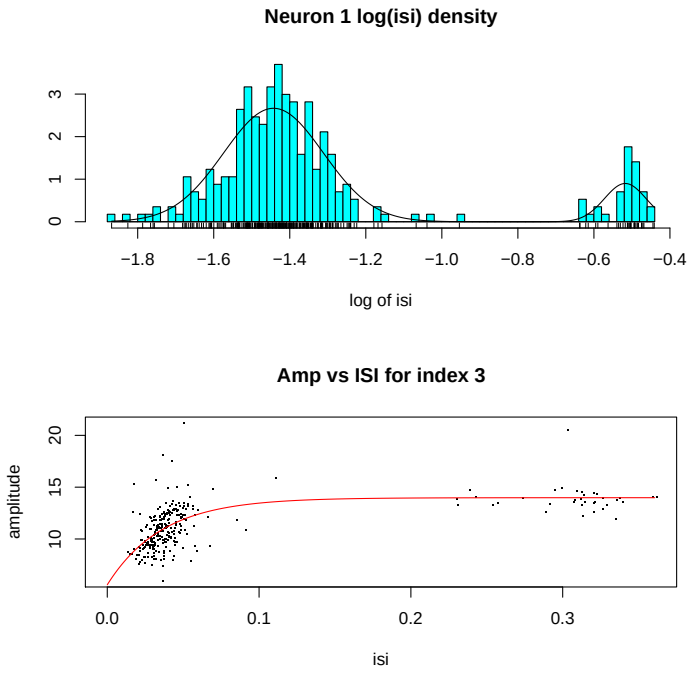

Figure 12: Model fit of neuron index `large.on.3`. *Top* ISI histogram of neuron index `large.on.3` with its model ISI density superimposed (solid curve). *Bottom* Spike amplitude *versus* ISI for of neuron index `large.on.3` with its model exponential relaxation superimposed.

how both parts of the data generation model (discharge statistics and amplitude relaxation) fit the data coming from individual neurons. For that purpose, function `display.model` displays two plots (figure 12): On the first plot, the ISI histogram of the chosen neuron, as it is isolated by the *MCMC* algorithm, and its model log-normal curves, as they are specified in the object `mcmc.model`, are superimposed. On the second plot, the spike amplitudes *versus* ISIs of this neuron and its model relaxation exponential curve, as it is specified in `mcmc.model`, are superimposed.

```
> display.model(spike.list = analyze, model.list = mcmc.model,
+     neuron.index = large.on.3, how.long = 10, nbins = 100)
```

Besides the `analyze` object and the model `mcmc.model`, that we have just computed, we need to specify the index of the neuron we want to plot, as well as the maximal ISI for the amplitude plot (`how.long`, as a multiple of the relaxation time constant of this neuron) and the number of bins, `nbins`, of the ISI histogram.

7. Conclusions

We have described the package **SpikeOMatic** for performing automatic and reliable spike-sorting on multi-unit data in electrophysiology. This package enables the experimentalist to perform the whole analysis on his raw data, from spike detection to a complete spike-sorting. Two spike-sorting procedures are implemented, based on two data generation models that are relevant for a variety of multi-unit data.

The next major addition will be the implementation of Ogata's "thermodynamic integration" (Ogata (1990)) to estimate the number of neurons in the framework of our *MCMC* algorithm.

Acknowledgments

This work was supported in part by a grant from the Ministère de la Recherche (ACI Neurosciences Intégratives et Computationnelles, pré-projet, 2001-2003) and by a grant inter EPST (Bioinformatique). Matthieu Delescluse was supported by a fellowship from the Ministère de l'Education Nationale et de la Recherche.

References

Camproux A, Saunier F, Chouvet G, Thalabard J, Thomas G (1996). "A Hidden Markov Model Approach to Neuron Firing Patterns." *Biophys J*, **71**, 2404–2412.

Delescluse M, Pouzat C (????). "Efficient spike-sorting for multi-states neurons." *submitted*.

Dempster A, Laird N, Rubin D (1977). "Maximun likelihood from incomplete data via the EM algorithm." *J R Stat Soc B*, **39**, 1–38.

Fee M, Mitra P, Kleinfeld D (1996). "Variability of extracellular spike waveforms of cortical neurons." *J Neurophysiology*, **76**, 3823–3833.

Fishman G (1996). *Monte Carlo Concepts, Algorithms and Applications*. Springer-Verlag, new-york edition.

Geman S, Geman D (1984). "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images." *IEE Trans Pattern Anal Mach Intell*, **6**, 721–741.

Hukushima K, Nemoto K (1996). "Exchange Monte Carlo and application to spin glass simulations." *J Pys Soc Jpn*, **65**, 1604–1608.

Liu J (2001). *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, new-york edition.

Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953). "Equations of state calculations by fast computing machines." *J Chem Phys*, **21**, 1087–1092.

Mitsutake A, Sugita Y, Okamoto Y (2001). "Generalized-ensemble algorithms for molecular simulations of biopolymers." *Biopolymers*, **60**, 96–1023.

Ogata Y (1990). "A Monte Carlo Method for an Objective Bayesian Procedure." *Ann Inst Statist Math*, **42**, 403–433.

Pouzat C (2005). *Methods and Models in Neurophysics*, pp. 729–786. Springer-Verlag, Berlin, j dalibard et al. edition.

Pouzat C, Delescluse M, Viot P, Diebolt J (2004). "Improved Spike-Sorting by Modeling Firing Statistics and Burst-Dependent Spike Amplitude Attenuation: a Markov Chain Monte Carlo Approach." *J Neurophys*, **91**, 2910–2928.

Pouzat C, Mazor O, Laurent G (2002). "Using Noise Signature to Optimize Spike-Sorting and to Assess Neuronal Classification Quality." *J Neurosci Methods*, **122**, 43–57.

Robert C, Casella G (1999). *Monte Carlo Statistical Methods*. Springer-Verlag, new-york edition.

Schwarz G (1978). "Estimating the dimension of a model." *Annals of Statistics*, **6**, 461–464.

Affiliation:

Christophe Pouzat
CNRS - UMR 8118
Université René Descartes
45 rue des Saints-Pères
75006 Paris, France
E-mail: christophe.pouzat@univ-paris5.fr
URL: http://www.biomedicale.univ-paris5.fr/physcerv

5.4 Quatrième article : absence de corrélations entre décharges spontanées de cellules de Purkinje

Le quatrième article présente la nouvelle démarche de publication scientifique que nous avons mise en oeuvre dans le deuxième article et qui passe par la soumission d'un *compendium* permettant à tout lecteur de *reproduire entièrement l'analyse présentée par un article de neurophysiologie*. Afin d'illustrer cette démarche sur une étude concrète d'électrophysiologie, ce quatrième article présente l'analyse des corrélations d'activité spontanée des PCs dans les tranches.

Cette analyse est menée en trois temps. Tout d'abord, nous présentons l'analyse des trains de PAs spontanés de PCs enregistrées individuellement en configuration cellule-attachée lâche, dans les conditions contrôle (solution saline BBS) et bicuculline ($25\,\mu m$, appliquée dans le bain), afin de caractériser l'effet de la bicuculline sur la décharge des PCs : le blocage de la transmission synaptique $GABA_A$ provoque une augmentation de plus de 30% de la fréquence de décharge, accompagnée d'une diminution de plus de 35% de la SD des ISIs (compte non tenu des pauses de plus de 1 seconde qui peuvent avoir lieu dans les deux conditions).

Dans un deuxième temps, nous montrons que les corrélations PA à PA, telles qu'elles apparaissent dans les histogrammes de corrélation croisée (CCs) et les histogrammes des temps du précédent (TTFHs), ne sont pas modifiées par le blocage de l'inhibition $GABA_A$: les CCs de toutes les paires de PCs, comme leurs TTHFs, restent plats, dans les limites des intervalles de confiance d'indépendance des trains. Il est possible d'observer, dans les deux conditions, une légère translation de tout le CC (sur $[-100, +100\,ms]$) et du TTHF légèrement au-dessus de la valeur attendue sous l'hypothèse nulle d'indépendance, lorsque ceux-ci sont calculés sur des périodes de temps où les trains ne sont pas stationnaires : c'est la marque d'une évolution approximativement parallèle de la fréquence instantanée de décharge des deux cellules durant l'enregistrement.

Ces résultats sont inattendus si l'on considère ceux présentés dans le chapitre 2. Ils le deviennent moins lorsque l'on étudie de plus près la sensibilité des CCs et des TTHFs à une inhibition ou une excitation *commune* à deux cellules et non *directe*, de l'une sur l'autre. La troisième partie de notre analyse montre ainsi, sur des trains de PAs simulés, que les CCs détectent bien l'*excitation directe* d'une cellule présynaptique sur une cellule postsynaptique ; ils détectent moins bien l'*inhibition directe* (les simulations d'Aersten et Gerstein, 1985, sont sans équivoque sur ce point) ; et ils sont incapables de détecter des entrées excitatrices ou inhibitrices communes à deux cellules postsynaptiques, du moins pour des valeurs des paramètres de simulation reproduisant les trains d'interneurones et de PCs. Les résultats sont semblables pour les TTHFs.

Par conséquent, les résultats négatifs de l'analyse des corrélations de trains multiples de PCs ne signifient pas que l'inhibition de plusieurs PCs par un même interneurone n'existe pas ; ils signifient seulement que *les corrélations PA à PA qu'elle induit sont trop faibles pour être visibles sur les CCs et les TTHFs*. L'analyse des corrélations PA à PA nécessite la mise au point de méthodes différentes pour mettre en évidence des entrées synaptiques communes à deux cellules. En revanche, on peut affirmer que dans notre préparation, à température ambiante et en activité spontanée, les interneurones

ne sont pas capables de synchroniser les PAs des PCs.

Cette analyse de trains de PAs est donnée dans l'appendice de l'article ; elle constitue un papier "de démonstration". En effet, l'objet de ce quatrième article est d'introduire, dans la communauté des neurophysiologistes, l'idée de *reproductibilité des analyses de données*, ainsi que les outils qui existent pour la mettre en oeuvre. Nous avons choisi de le faire au moyen de l'exemple particulier d'analyse décrit ci-dessus. Le corps de cet article expose donc ce qu'est un compendium. Fondamentalement, il s'agit d'une entité distribuable regroupant les données de l'article, les codes utilisés pour les analyser, ainsi qu'un document (un "méta-fichier") exécutable comprenant aussi bien la séquence des commandes produisant l'analyse présentée que le texte descriptif de l'analyse. Nous décrivons les outils disponibles dans R pour réaliser un tel compendium, et en particulier ce méta-fichier dont l'exécution permet au lecteur de recréer sur sa propre machine l'analyse qui lui est présentée. En particulier, nous donnons, dans l'article lui-même, des *fragments du texte du méta-fichier qui a produit le papier de démonstration de l'appendice*, afin d'expliquer dans le détail la construction de ce méta-fichier. Enfin, nous exposons en quoi le compendium est, à notre avis, un outil extrêmement utile à la communication scientifique en général, et en quoi il se révèle être un support très précieux pour stocker, réutiliser, transmettre une analyse complète particulière au sein d'un laboratoire.

Evidemment, nous accompagnons la soumission de cet article du compendium comprenant les données du papier de démonstration, les routines particulières utilisées, ainsi que le méta-fichier produisant le papier de démonstration et dont nous donnons des extraits dans le corps de l'article. Ce compendium est disponible sur le site *web* du laboratoire[2]. Ce quatrième article illustre ainsi la démarche proposée, en même tant qu'il donne tous les détails de sa réalisation pratique.

Les programmes d'analyse de trains de PAs font l'objet d'un deuxième logiciel mis à la disposition de tous sur le site du laboratoire, sous licence GPL : *SpikeTrainAnalysiswithR*, ou *STAR*[3] (voir chapitre 4, section 4.4). Comme pour le premier logiciel, il s'agit d'un paquetage R regroupant l'ensemble des routines indispensables à l'étude de trains de PAs ainsi que leurs documentations respectives détaillées .

[2] `http ://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Compendium.html`
[3] `http ://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/STAR.html`

Reproducible Neurophysiological Data Analysis

Matthieu Delescluse and Christophe Pouzat

August 24, 2005

Abstract

Reproducible data analysis consists in having the complete analysis presented in a paper, from loading the data into an analysis environment to producing summary figures and tables, repeatable by anyone at anytime. The idea emerged in the last ten years in geophysics, signal processing and biostatistics and is exposed here to a neurophysiological audience. Following Gentleman and Temple Lang (Statistical Analyses and Reproducible Research, Bioconductor Project Working Papers. Working Paper 2. 2004) it is implemented by distributing a compressed file arborization containing the raw data used in the paper, the computer codes developed specifically to analyze them and a meta-file where a textual description of the analysis is mixed with the actual commands used to produce it. This file arborization is called a compendium and provides everything necessary to automatically re-generate the analysis together with a fully explicit description of it. A detailed implementation is presented using a study of correlations among Purkinje cells spike trains as an example. These Purkinje cells were recorded from rat cerebellar slices and the study was motivated by the presence of synchronous inhibitory inputs in neighboring Purkinje cells due to common pre-synaptic interneurons. This compendium implementation requires only open-source and freely available software running on every computer / operating system combination presently available in neurophysiology labs. The compendium concept and its associated tools offer an unparalleled precision in the communication and explanation of what is done with data to produce results. It looks therefore as an ideal support for supplementary materials accompanying published papers allowing authors to greatly strengthen their research claims.

1 Introduction

As readers and/or referees of neuroscience papers we probably all ran through questions about what we read like:

- What would happen to the analysis results if a given parameter had another value?
- What would be the effect of applying my pre-treatment to the data instead of the one used by the authors?
- What would a given figure look like with a log scale ordinate instead of the linear scale use by the authors?

We can all think of a dozen of similar questions. The problem is to find a way to address them. The classical journal article format cannot be entirely satisfying from this view point. Editors cannot publish two or more versions of each figure to satisfy different readers. Many intricate analysis and modeling methods would require a too long description to fit the usual bounds of the printed paper, etc. Moreover, some, not to say most, readers are still satisfied with the papers in their present format. This is perfectly reasonable since we all have a lot of different things to do and we cannot afford to look at every piece of work as thoroughly as just described. Nevertheless, many people feel uncomfortable with the present way of diffusing scientific information as a canonical (printed) journal article, even if now those can be "enhanced" with supplementary web material. What is needed is a more systematic and more explicit way to describe how the analysis (or modeling) was done while letting the freedom to the reader/referee to dive into the details or to stay at the level of the canonical article.

This "problem" has already been discussed in other scientific fields like geophysics [37, Schwab et al., 2000], signal processing [5, Buckheit and Donoho, 1995], statistics [34, 35, Rossini, 2001; Rossini and Leisch, 2003] and biostatistics [12, Gentleman and Temple-Lang, 2004]. The solutions proposed consist all in a mechanism allowing authors to distribute as a single compressed file a folders plus files arborization containing:

- The raw data analyzed in the paper.
- The codes developed by the authors specifically to analyze the data.
- A "meta-file" where a textual description of what is done with the data is *mixed* with the actual commands used to perform the analysis; these commands being expressed in a high level language like Matlab or R scripts.

Such a package is called a `compendium` by Gentleman and Temple-Lang [12] and the general approach is commonly dubbed `reproducible research` following Buckheit and Donoho [5] and Gentleman [13]. The key idea is that the meta-file when properly processed allows any reader/referee to *regenerate* the complete analysis presented in the paper, from loading the raw data into the analysis environment to producing the final figures and tables. Because the meta-file is *fully explicit* about what is done with the data the reader/referee can for instance specifically modify the parameters and see the consequence on the paper's results.

Tools are clearly needed in order to go from the compendium concept to its realization. Although several practical implementations have been developed we will focus here on the most recent one which is the easiest to learn and use by non-programing experts. It has the advantage that the tools required to generate a compendium and process the meta-file are part of a general data analysis environment. This environment, R[1] [32], although not very common (yet) in neuroscience labs, has a command line syntax similar to Matlab, is open source and freely available and

[1] http://www.r-project.org

runs on all computer / operating system combinations found in neurophysiology labs. Following the R nomenclature we will from now on use the term `vignette` to refer to the meta-file.

In order to keep our reader in a familiar context we will illustrate the compendium concept with a simple implementation which will constitute "a demonstration paper" making the core of the Appendix. The biological question is motivated by the following results:

1. We know that, in young rats (P11-15), GABAergic inhibitory interneurons of the molecular layer of the cerebellar cortex, stellate and basket cells, often evoke powerful (chloride) conductance in their post-synaptic targets which are mainly Purkinje cells (PCs) [40, 30].

2. The axons of these interneurons go 200 to 300 μm in the sagittal plane along the PC layer making inhibitory synapses on several neighboring PCs [28] on their way. Vincent and Marty [39] showed moreover that several PCs do receive simultaneous IPSCs from the same pre-synaptic interneuron.

3. Interneurons → PC paired recordings have shown that a pre-synaptic spike in an interneuron can *delay* spikes in a post-synaptic PC *already engaged in a tonic discharge* [14].

We therefore ask: *Can these simultaneous and potentially large GABAergic conductances in several PCs lead to observable correlations of their spike trains*? This question has been tackled experimentally by recording extracellularly groups of PCs in control condition and in the absence of any $GABA_A$ inhibition. The latter condition was obtained by adding bicuculline to the control saline solution, thereby blocking any fast interneuron → PC inhibition. For the reasons explained above, we expected to see correlations between PC spike trains in control condition due to GABAergic inhibition from common interneurons. These correlations should have disappeared or have had another form upon inhibition suppression by bicuculline.

2 Methods

2.1 R basics

We do not assume that the reader is familiar with R [32] but we do assume she has some familiarity with other data analysis environments like Matlab or Igor. So R is an open source, freely available implementation of the S programming language [7, 6, Chambers, 1998 & 1999]. In fact most of what is presented in this paper also applies to the commercial implementation of S, S-Plus. Like Igor and Matlab, R is a *functional* language, meaning that if `my.isi` is the name of the working space variable containing the inter-spike intervals (ISIs) of a neuron, entering the following command:

```
> mean(my.isi)
```

returns the mean ISI value[2]. `mean` is the function name, whose arguments are, clearly enough, given between "`()`". Functions can be:

- R built-in functions.
- Part of an additional function bundle provided by a user, that is, a `package` in the R jargon.
- A function defined directly by the user.

Most R built-in and package functions have default argument values. If the user wants to use the default values she does not need to specify them in the function call. The documentation which is necessarily provided with R built-in and package functions is of course explicit about these default values. There are two ways to specify variable assignment in R:

[2]Here and in the sequel, ">", represents the R command-line prompt.

```
> my.isi.mean = mean(my.isi)
> my.isi.mean <- mean(my.isi)
```

That is, both the classical "=" and "<-" can be used. We will use the latter in this paper[3].

2.2 The Compendium: Building an R package

Following [12, Gentleman and Temple Lang, 2004] and [13, Gentleman, 2004] we will exploit the R built-in functionality to create and distribute packages in order to create and distribute our compendium. Originally R packages were designed to allow (and encourage) individual users to distribute easily specific software libraries with their associated documentation and demo data sets. It just turned out that this package framework was very well suited for an implementation of the compendium concept. What follows is just a minimal outline of an R package and its management, detailed documentation can be found in [33]. An R package is a parent directory that minimally contains 2 sub-directories called `R` and `man`, as well as two files, "DESCRIPTION" and "README". The `R` directory contains all the R functions of the package (`.R` files). `man` contains the help files, with extension `.Rd`, that document the functions put in the `R` directory. Extra source codes in foreign languages (C, C++, Fortran, ...) can be placed in another directory called `src`. There can be an additional directory called `data` where data files are put. One can also have a directory `inst` containing a sub-directory `doc` in which the vignette itself, with `.Rnw` extension, is placed.

The function `package.skeleton` automatically creates the structure and the arborization of an R package: it creates the directories described above and saves functions and data to the appropriate places. It also creates a skeleton for the help files in `man`. The use of this function is simple: its arguments are the name of the package (which will be the parent directory) and a list of R objects that have to be placed in the package (functions and data). The following code thus creates a package `myRPackage` containing functions `isi.histograms` and `cross.correlograms`. The help files `isi.histograms.Rd` and `cross.correlograms.Rd` are created and have to be edited and completed.

```
> functions.list <- list("isi.histograms", "cross.correlograms")
> package.skeleton(list=functions.list, name="myRPackage", path= "myHome")
```

The parent directory `myRPackage` is placed in the path specified by the argument `path` (if not specified, this is the present R working directory). If no list of R objects is given, the function `package.skeleton` creates a package with all the objects present in the working environment. These objects are put in the directories `R` and `data` according to their class (function or simple variables).

Any supplementary R function can be added to the package latter on in the sub-directory `R` of the package and *must* be documented. To create the skeleton of a `.Rd` file corresponding to an added standard R function, say `ttf.histograms`, and place it in the `man` sub-directory, one can use `prompt` with the function to be documented as argument, like:

```
> prompt(ttf.histograms, filename="myRPackage/man/ttf.histograms.Rd")
```

The argument `filename` specifies the path of the created file. If it is not given, the file `ttf.histograms.Rd` is created in the directory where R has been started.

Other "prompt" functions generate other types of `.Rd` skeleton for specific types of object like: `promptClass` for a `.Rd` describing a class, `promptMethods` for a `.Rd` describing a method related to a particular class, or `promptData` for a `.Rd` describing a data file.

[3] The older among the readers perhaps remember that the programming language `Pascal` also had a way to distinguish between the assignment operation and the mathematical equality statement, using, `:=`, for the former.

2.3 The vignette

The fundamental feature of a meta-file or vignette is that it is a document containing both text and command segments or `chunks`. The text chunks describe what the commands do as well as their outputs and the possible figures they produce. In that respect a vignette is a *literate program* [18, 35], *i.e* an executable document combining text and commands and whose content is dynamically generated. A specific program processes the vignette by doing two things:

- The text chunks are extracted and copied verbatim into a new output file.
- The commands chunks are extracted and executed sequentially. Their results are copied into the new output file together with or, optionally, in place of the corresponding command chunk of the original vignette. If figures and/or tables are generated they are also included into the new output file.

By mixing textual description and *explicit* commands, the vignette gives access to every detail of an analysis making the latter fully reproducible by anyone *and* allowing anyone to explore the consequences of analysis parameters modifications.

In the R analysis environment vignettes are processed by the `Sweave`[4] function developed by Friedrich Leisch [20, 19, 22, 21]. The vignette itself is a pure ASCII file where text chunks are written with LaTeX and command chunks are normal R commands. In other words, writing a vignette can be seen as writing a LaTeX file in which, instead of inserting prefabricated graphs and tables, the author puts the *R commands* necessary to obtain them. When this vignette is processed within R by `Sweave`, the commands are executed and all commands outputs, including figures, are inserted into a final LaTeX file. This file can be further processed into a PDF or any other desirable format.

Most of the time the vignette requires supporting software for the proper execution of its command chunks. This software can be a *general purpose software* available within the R environment. It can also be an *auxiliary software* written specifically by the authors for the analysis they are presenting. If no auxiliary software is used with the vignette, the reader can run it as such in her R environment. Most often though, codes written by the authors for the analysis must be provided together with the vignette in the compendium. These codes can of course be a simple accompanying R script file (usually with a `.R` extension) containing all the functions used by the vignette.

Finally it is important to mention that if the raw data files are too heavy or too numerous to be all included in the distributed compendium, they can be left on a server from where they can be loaded into R.

2.4 The Bare Minimum of LaTeX

As was made clear in the previous section, writing an Sweave document requires a minimal knowledge of LaTeX. That will probably mean nothing for most of the readers. LaTeX is a typesetting *language*. Its design is drastically different from the one of the most common word processors like Open Office Writer[5], meaning that some learning is required to use it properly, but good introductions are freely available [26, 11, Oetiker et al, 2005; Flynn, 2005]. LaTeX source files, like HTML source files and of course vignettes, are ASCII files. That implies that all the usual features of classical papers like font variations, symbols, etc, cannot appear as such in the file. They are instead *marked* as such with a specific convention (set by the LaTeX language syntax). For instance the symbol "μ" is obtained by writing `$\mu$` in the source file[6]. "This is normal

[4] http://www.ci.tuwien.ac.at/~leisch/Sweave

[5] http://www.openoffice.org/.

[6] Here the first "$" signals the beginning of the equation mode and the second the end. the "\mu" is interpreted, within the equation mode as the Greek symbol μ.

text$_{this\ is\ subscript}$" is entered as: "This is normal text$_{this\,is\,subscript }$". A clear advantage is that one immediately sees when a subscript starts and when it ends. The logical structure of the text: title, section, subsection, etc, is also marked as such in the source file, meaning that the writer does not have to bother with the font and font size of the titles, the section numbering, etc. The program which processes the source files to produce, say, the PDF output, will manage all that. Now a major advantage of having an ASCII source file is experienced daily by the reader when he browses web pages with (mainly) textual content: it is fast and reliable. The latter meaning here that the display does not depend neither on the computer nor on the operating system types. We can therefore write papers with collaborators across the world and exchange manuscript versions by directly pasting them in e-mail messages without having to bother with translation of binary formats between, say, Mac and PC. We can also fit a whole book on an old fashioned floppy. Of course much more can be done as detailed by [25, Mittelbach et al, 2004].

2.5 Handling an Sweave file

Writing an Sweave file

Using a text editor

Writing an Sweave file is pretty much the same as writing a LaTeX[7] file and inserting R code chunks in it. The latter are just separated from the surrounding LaTeX text by:

```
<<>>=
# R code
@
```

The *Emacs*[8] editor is particularly suited to edit Sweave files if used with the additional package *Emacs Speaks Statistics* (ESS)[9] [36]. ESS knows the syntax of R language (and that of other statistical analysis packages) so that it provides consistent display and editing features.

Using LyX

If you do not want to bother about writing LaTeX codes you will appreciate the use of LyX[10]. It is a practical application that provides an easy to use Graphical User Interface allowing to write and process a LaTeX document without typing the LaTeX commands: you type your plain text, and save it as a `.lyx` file that can be exported into a `.tex` file. In other words LyX generates the LaTeX code for you. It is thus possible to write an Sweave file with LyX: it suffices to insert the code chunks in your text as shown previously, with TeX insets (Menu Insert, TeX).

Compiling an Sweave file

Here are the three steps to follow:

1. Write an Sweave file `mySweaveFile.Rnw` as described above.
2. Run this Sweave file in R with `Sweave` function:

   ```
   > Sweave("mySweaveFile.Rnw")
   ```

   This creates a `.tex` file.
3. Compile this `.tex` file with the tool of your choice to get a DVI or a PDF file.

[7] http://www.latex-project.org/
[8] http://www.gnu.org/software/emacs/emacs.html.
[9] http://ess.r-project.org/
[10] http://www.lyx.org

If you use LyX, following Leisch[11], the compilation process goes like:

1. Write your LyX file `mySweaveFile.lyx` with code chunks as described above.
2. Export your LyX file to LaTeX (`mySweaveFile.tex`).
3. Rename this LaTeX file to Rnw (`mySweaveFile.Rnw`).
4. Run this Sweave file in R as shown above. This creates a new `mySweaveFile.tex`.
5. Compile this `.tex` file with the tool of your choice to get a DVI or a PDF file.

2.6 Availability of the compendium

The compendium re-generating the demonstration paper displayed in the Appendix is an R package available on our web site: http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Compendium.html. The analysis performed in this compendium resorts to our add-on R package *STAR* that needs to be installed beforehand; this package is also available on our website: http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/STAR.html.

3 Results

The Appendix contains the complete demonstration paper we will be referring to in this section. This demonstration paper has been generated by processing a vignette, as described in the Methods section. We will focus here on *how this Sweave file has been built*. To our mind the best way to do that is to clearly and precisely relate specific segments of the Sweave file to their corresponding outputs in the final document provided in the Appendix.

We have therefore extracted 5 parts in our demonstration paper. Each of these parts is located in between two lines of "*". In the following 5 insets we display the texts of the corresponding parts of our Sweave source file. In other words each inset shows the specific Sweave lines that generate the related selected part of our demonstration paper.

3.1 A basic example of a complete *Sweave* file

It is necessary to first give an outline of what the architecture of an Sweave file is. To do that, we skip the Methods section of our demonstration paper (which is, like the Discussion section, a simple LaTeX text) and we consider the 1st extract as a complete, stand-alone document. This document is generated by processing a short Sweave source file whose text is integrally displayed in the following inset:

```
1  \documentclass[10pt,english,a4paper]{article}
2  \usepackage[pagebackref=true,colorlinks=true]{hyperref}
3  \usepackage{geometry}
4  \geometry{tmargin=1.5cm,bmargin=1.5cm,lmargin=2.25cm,rmargin=2.25cm,headheight=0.5cm,
5    headsep=0.5cm,footskip=0.5cm}
6  \pagestyle{empty}
7
8  \begin{document}
9
10 \subsubsection*{Single cell recordings}
11
```

[11] http://www.ci.tuwien.ac.at/~leisch/Sweave/LyX/

```
12 We start by loading our analysis library, as well as the single cell
13 data of our compendium into our analysis environment:
14
15 <<results=hide>>=
16 library(STAR)
17 library(DelesclusePouzatDemoPaper)
18 data(single.spike.train)
19 @
20
21 Then a call to function \texttt{summary} gives us a short reminder
22 of what is in the data:
23
24 <<>>=
25 summary(single.spike.train)
26 @
27
28 \paragraph*{Rate evolution}
29 We see that a Purkinje cell (PC) was recorded during two
30 epochs (control and bicuculline) of 5 minutes (300 s) each. We can then
31 get a bird-eye view on the data by calling function
32 \texttt{rateEvolution} which generates estimates of instantaneous
33 neuronal discharge frequency of this neuron by convolving the spike
34 train with a Gaussian kernel whose default width is 1 s:
35
36 <<results=hide>>=
37 rateEvolution(single.spike.train)
38 @
39
40 <<echo=FALSE>>=
41 dede<-dev.print(device=postscript,"rateEvolSingle.eps")
42 ede<-dev.print(device=pdf,"rateEvolSingle.pdf")
43 @
44
45 In Fig.~\ref{fig:rateSingle} we see immediately the net increase of
46 the average firing rate upon application of  bicuculline. On this
47 figure we also identify periods of stationarity in both
48 conditions: 150-300 s in control condition and 350-450 s in
49 bicuculline condition.
50
51 \begin{figure}
52 \begin{center}
53 \includegraphics[width=0.5\textwidth]{rateEvolSingle}
54 \caption{\label{fig:rateSingle} Rate evolution of a PC in control conditions and with $25 \mu M$
55   bath applied bicuculline.}
56 \end{center}
57 \end{figure}
58
59 \end{document}
```

As described in the Methods section this file can be seen as a LaTeX file with inserted R commands. It therefore begins with a usual LaTeX heading (l.1 to 7) which sets global parameters for the whole document. This heading is followed by the LaTeX `begin` command that opens the document (l.8). The content of the latter therefore starts in l.10 with a section title and two lines

of plain text explaining what the following code chunk is going to do. This code chunk (l.16 to 18) is separated from the preceding LaTeX text by `<<results=hide>>=` (l.15). The symbol `<< >>=` introduces the R code environment in the Sweave file and the argument, `results=hide`, suppresses the inclusion of the messages written by R as a result of the three function evaluations into the output file. We return to the LaTeX environment with "@", placed at the end of the code chunk (l.19). The following lines are therefore plain LaTeX text again (l.21-22) until l.24 that re-introduces a new code chunk, and so on until the final LaTeX `end` command of l.59 that closes the document.

The first code chunk (l.16-18) is meant to load the required functions and data to perform the analysis. We first load (l.16) the R package *STAR* that has the functions that will be used (see Methods). We also load the compendium (l.17) which contains the data to be analyzed, as well as specific R functions specifically designed for this particular analysis. Finally we load the single neuron spike train to analyze (l.18).

The second code chunk (l.25) perfectly illustrates the key point of the Sweave file. When this file is processed within R the command `summary(single.spike.train)` (l.25) is executed; it generates an output and this output is inserted in the final document (the argument `results=hide` was *not* used here). You can read this output in the 1st Sweave extract of our demonstration paper following the call to function `summary`.

In the same vein the output of function `rateEvolution` (l.37) generates a plot. This plot can be saved as a `.pdf` and/or a `.eps` file. This is the purpose of code lines 41 and 42. This figure is then inserted into the final document with the appropriate LaTeX commands (l.51 to 57).

Following each code chunk, few lines of plain LaTeX text comment the output obtained and introduce the next code chunk (l.26-31 and l.42-45).

This short and simple Sweave file contains one more detail that is worth noticing: two code chunks are introduced by a non empty symbol `<< >>=`, in l.15 and l.40. We have already explained the meaning of `results=hide` in l.15. In l.40 the the argument `echo=FALSE` prevents the following R commands themselves to be inserted into the output file. That is why the function calls of this code chunk (l.41 and l.42) do not appear in the final document (1st Sweave extract of our demonstration paper in the Appendix). When nothing is specified within the symbol `<< >>=`, the R commands and their outputs are by default automatically inserted in the final document. We thus control very simply what we see from the R chunks and their outputs in the final document.

3.2 Commenting out parts of the *Sweave* file

Now that the principal features of a Sweave file have been introduced on the 1st extract, we will use each subsequent extract of our demonstration paper to bring up a new particular asset of the vignette.

Let us consider the 2nd extract. The following inset displays the part that generates it in the Sweave source file of this paper.

```
1  \paragraph*{Statistical independence of successive ISIs in the
2    control condition}
3  We test the statistical independence of successive ISIs of this PC in control condition. For
4  that test we select a period of data that we previously identified as stationary
5  (150-300 s in control condition). This data selection appears
6  explicitly in argument \texttt{time.window}.
7
8  <<>>=
9  renewalTest(single.spike.train,epoch="con",time.window = c(150,300))
10 @
11
12 These Q-Q plots show that, over the selected period, the ISIs
```

```
13 conditioned to the previous ISI duration have the same distribution as the
14 whole sample of ISIs (Fig.~\ref{fig:renewalControl}). This indicates
15 that successive ISIs are \emph{statistically independent} over this
16 period. It allows us to assume that this spike train is a \emph{renewal process} that
17 can be completely described by its ISI distribution.
18
19 <<echo=FALSE>>=
20 dede <- dev.print(device=postscript, "renewalControl.eps")
21 dede <- dev.print(device=pdf, "renewalControl.pdf")
22 @
23
24 \begin{figure}
25 \begin{center}
26 \includegraphics[width=0.75\textwidth]{renewalControl}
27 \caption{\label{fig:renewalControl} Q-Q plots of ISIs conditioned on the
28   previous ISI duration \emph{vs} the whole ISI distribution (control condition).}
29 \end{center}
30 \end{figure}
31
32 %\paragraph*{Statistical independence of successive ISIs in the
33 %  bicuculline condition}
34 %We draw the same plots for the bicuculline condition
35 %(Fig.~\ref{fig:renewalBicuculline}). The stationary
36 %period retained here is 350-450 s.
37
38 %<<>>=
39 %renewalTest(single.spike.train,epoch="bic",time.window = c(350,450))
40 %@
41
42 %<<FALSE>>=
43 %dede<-dev.print(device=postscript,"renewalBicuculline.eps")
44 %dede<-dev.print(device=pdf,"renewalBicuculline.pdf")
45 %@
46
47 %\begin{figure}
48 %\begin{center}
49 %\includegraphics[width=0.75\textwidth]{renewalBicuculline}
50 %\caption{\label{fig:renewalBicuculline} Q-Q plots of ISIs conditioned on the
51 %  previous ISI duration\emph{vs} the whole ISI distribution (bicuculline condition).}
52 %\end{center}
53 %\end{figure}
54
55 %One sees in Fig.~\ref{fig:renewalBicuculline} that in this condition
56 %also the successive ISIs of the selected stationary
57 %spike train are independent. This spike train also can be considered to be a
58 %renewal process.
```

This inset introduces another simple but very practical feature of Sweave files: one can write down comments like in every LaTeX document. One just has to put a "%" at the begining of every comment line. These lines are simply ignored when the Sweave file is processed. This provides a means to temporarily suppress parts of the Sweave file. To run these parts again, it suffices to remove the "%".

That is why the output of lines 32 to 58 do not appear in the 2nd extract. This part of

the Sweave file repeats the analysis done in the control condition (l.1 to 30) for the bicuculline condition. The output of this analysis, namely the generated Q-Q plots, are similar in both conditions. To obtain a more concise final document, we simply comment out this part of the Sweave file, as we did here. The related results are then said to be "not shown". We thus avoid displaying two similar figures. However, it is straightforward to uncomment this part in order to re-generate this "not shown" analysis on one's own machine.

This is a means to keep available to anyone the complete analysis without charging to much the final, static paper whose space is naturally limited. As a rule then, every stated but "not shown" result can be obtained by running the vignette provided with the paper. This rule naturally holds for the whole demonstration paper displayed in the Appendix. The reader is therefore strongly invited to generate by herself all the not shown figures she would like to see.

3.3 Making explicit every analysis parameter

The following inset displays the few lines of our Sweave source file that produce the 3rd extract:

```
1 \paragraph*{Fit of the ISI distribution in the control condition}
2 We then fit the ISI distribution of the selected period of the control
3 spike train with 4 different models (Maximum Likelihood Estimator, MLE):
4
5 <<results=hide>>=
6 ctlModel <- isiMle(single.spike.train,epoch="con",
7                    time.window = c(150,300),
8                    isi.density = c("lnorm","invGauss","weibull"))
9 @
```

With this inset we make another crucial point of such a vignette. Consider the call to function `isiMle` shown in the preceding inset (l.6). The argument `time.window` of this function selects the period $[150 - 300s]$ in the spike train. This period corresponds to the stationary period we want to analyze. We thus make *explicit* which part of the data has been selected to be fitted by the four chosen densities. More generally, *all the parameters used to perform an analysis do appear* in the Sweave file and in its final output. Therefore everyone knows them and, even more *can change them*. The latter point will be developed in section 3.5.

3.4 Using functions specific to this analysis

Until now we have exclusively used functions that are found in an *external* R package, namely *STAR*, that is loaded at the beginning of the analysis. The compendium designed to reproduce the whole analysis has also been loaded to get the data to analyze. In addition to the data the compendium contains some specific functions written for this particular analysis. To generate the 4th extract of our demonstration paper we resort to these functions. Therefore in the corresponding part of the Sweave source file, whose text follows, loading the compendium package as we did in the first Sweave extract is necessary to get access to the data *and* to these auxiliary functions.

```
1 In the compendium, there is a set of 10 single PC spike trains recorded in ``old'' rats
2 aged between 21 and 25 days post-natal (including the PC studied so far)
3 and a set of 7 single PC spike trains recorded in ``young'' rats aged
4 between 11 and 15 days post-natal. These sets are named ``PC'' and ``jPC'' respectively.
5
6 We load the first data set:
7 <<results=hide>>=
```

```
load.data("PC")
@

We then compute and plot the mean, standard deviation (SD) and coefficient of
variation (CV) of the 10 single PC ISIs from the old rats data set, in
control and bicuculline conditions (Fig.~\ref{fig:isiStats}):

<<>>=
stats.list <- isi.stats("PC")
display.isi.stats(stats.list)
@

<<echo=FALSE>>=
dede<-dev.print(device=postscript,"isiStats.eps")
ede<-dev.print(device=pdf,"isiStats.pdf")
@

\begin{figure}
\begin{center}
\includegraphics[width=0.7\textwidth]{isiStats}
\caption{\label{fig:isiStats} 21-25 days old rats. \emph{Top}: mean ISI values with SDs for
10 PCs in control (filled circles) and bicuculline (open circles)
conditions. The difference between both values is significant for all
cells (see text). Ordinate axis in ms. \emph{Bottom}: coefficient of
variation (CV) for the same 10 PCs in control (filled circles) and
bicuculline (open circles) conditions.
}
\end{center}
\end{figure}

We see in Fig.~\ref{fig:isiStats} that, for each PC, the ISI mean is
smaller in bicuculline than in control condition. We also see that the
ISI SD decreases in bicuculline with respect to the control condition.

To find out whether these observed changes are significant or not, we
perform a one-sided t-test and a one-sided F-test respectively.  The ISIs of PCs are not
normally distributed but the ISI samples are large, so that a t-test
and an F-test are still relevant.

<<>>=
PC.t.test("PC")
PC.var.test("PC")
@

All \emph{p}-values are almost null, so that the null hypothesis of
equal means for the control and bicuculline ISI distribution is
rejected. For all PCs, the ISI mean in bicuculline is significantly smaller
than in control. This also holds for the ISI variance which
significantly decreases in bicuculline, except for the 8th PC.

Similar results are obtained with the data set of ‘‘young’’ PCs (not shown).
```

These auxiliary functions are called in lines 8, 16, 17, 48 and 49. Once again, the results obtained with the other data set are not shown here to avoid charging to much the paper (l.58), but they can be generated by editing the Sweave file and removing the "%" to uncomment the part which follows this line 58.

3.5 Changing the analysis parameters

This section demonstrates another key point of the vignette. As described in section 3.3 the analysis parameters used by the author are explicitly displayed in the Sweave file. More than that the reader can change herself these parameters at her convenience so as to produce any figure she would like to see with these data. The following part of our analysis perfectly illustrates this invaluable asset.

```
1  To understand these rather unexpected negative results, we need to
2  explore quantitatively the sensitivity of the tools we used to analyze
3  temporal cross-correlations between spike trains. For that purpose we
4  simulate interacting spike trains.
5
6  \subsubsection*{Common inhibition}
7  We first simulate an interneuron train with log-normal scale and shape
8  parameters equal to $0.3$ s and $0.05$ respectively:
9
10 <<results=hide>>=
11 para.pre<-list(c(0.3,0.05))
12 pre.train<-simul.one.epoch(para.pre,"lognormal","epoch",0,120)
13
14 @
15 We then simulate two PC trains that are post-synaptic to this interneuron
16 train. The log-normal scale and shape parameters of these PCs are
17 respectively $(0.06 s, 0.3)$ and $(0.08 s, 0.3)$. We make the pre-synaptic
18 spike prevents the post-synaptic cells from firing ($\delta = -1$),
19 with a time constant $\tau = 40 ms$ for the exponential relaxation.
20
21 <<results=hide>>=
22 para.post<-list(c(0.06,0.3), c(0.08,0.3))
23 post.trains.in<-simul.postsynaptic.trains(parameters=para.post,
24                                           presynaptic.train=pre.train,
25                                           post.type=c("lognormal", "lognormal"),
26                                           delta=-1, tau=0.04, epoch.name="simul_inhibition")
27 @
28
29 We plot the CCs of the pre-synaptic and post-synaptic trains (Fig.~\ref{fig:ccSim1}).
30
31 <<results=hide>>=
32 cross.correlograms(post.trains.in, epochs="simul_inhibition", bin.width=0.005, max.lag=0.1)
33 @
```

The output of this Sweave file section is the 5th extract of our demonstration paper: we simulate interacting spike trains and study their temporal correlations. Three spike trains with log-normal inter-spike interval (ISI) densities are simulated; one train is "pre-synaptic" and inhibits the other two ones. These simulations require several parameters to be specified (see Methods):

1. The 3 pairs of log-normal parameters (scale and shape) for the 3 neurons (l.11 and l.22).

2. The 2 parameters (δ and τ) of the synaptic effect exerted by the pre-synaptic neuron onto the firing of the post-synaptic ones (in the call to function `post.train.in` in l.23).

In the analysis shown in the preceding inset we set these parameters to values that simulate the experimentally observed spike trains in our preparation: the pre-synaptic (interneuron) frequency is approximately 3 to 4 times less than the post-synaptic (PC) frequencies. The chosen value of δ simulates the strongest inhibition possible, namely a complete silencing of the post-synaptic cells, with and exponential relaxation whose time constant is certainly longer than usual durations for synaptic effects.

By simply editing this file, the reader can try by herself different values for these parameters. She can for instance simulate pre- and post-synaptic spike trains with similar frequencies to see that it does not fundamentally change the cross-correlogram between the post-synaptic neurons. She can also try other values of δ and τ. She will convince herself that common inhibitory inputs do not appear on CCs. This possibility to explore other parameter values is undoubtedly a crucial added value of the vignette.

4 Discussion

We have presented in this paper the idea of *reproducible data analysis* together with an *actual implementation* of it on a simple study of (absence of) correlations among Purkinje cell spike trains. Reproducible data analysis simply consists in having a complete analysis, from loading the data into an analysis environment to producing summary figures and tables, repeatable by anyone at anytime. The analysis can be the "routine analysis" of the last experiment performed in the lab as well as the "whole analysis" of a set of experiments making a paper. "Anyone" can be, depending on the context, the original author of the analysis, someone of the same lab, a collaborator, a referee, the reader of a paper. Achieving reproducible analysis certainly requires a minimal discipline on the side of the analysis author as well as *proper tools*. We have tried to make clear in this paper that these tools are now available for all computer / operating system combinations presently available in neurophysiology labs. Users have to make the effort of learning these tools, namely the R analysis environment and the LaTeX typesetting language. This effort will nevertheless not be spent only at learning how to make a compendium. The R analysis environment is a very powerful data analysis software on its own right and we would personally keep using it even if it did not make compendium management so easy. The same holds for LaTeX which is such an efficient typesetting language that any scientific author should benefit from an at least rudimentary knowledge of it.

Our actual implementation was presented with the referee / reader of a scientific paper in mind. We have tried to demonstrate that in such a context the compendium constitutes a fully explicit presentation of the analysis performed on the data. This presentation is moreover *dynamic* because the referee / reader has access to the different parameters used for the analysis, *can change them and repeat the analysis with the new parameters*. In our experience, writing things in the most explicit manner is the best way to find our own mistakes. In the not so infrequent cases where we do not succeed in finding them, it is still the best way for others to detect and correct our mistakes. The bottom line is that we think we all do a lot of mistakes and the compendium should help us as a community to produce more reliable knowledge on a shorter time span.

The compendium paradigm clearly shifts power towards the reader because the latter gains more freedom to "choose" what is interesting in the paper. If a specific analysis method or display technique looks interesting to us in a paper, we can easily extract it from the compendium and use it in our own work. The same holds for the data which become accessible for analysis by other people with other methods. We think that such an openness should also pay off for the original authors because it should result in more citations for the paper / compendium.

Leaving the published or submitted paper context it is probably worth pointing out that a compendium can also ideally be used as a lab book. Everyone who has been active for a few years

in research is very likely to have experienced the following situation. We start working on a project, do experiments, analyze the data regularly and after one or two months we have to stop because we have to write a grant application, reply to the referees comments about our last submitted manuscript, prepare the new course we are teaching this year, take care of the new post-doc who just joined the lab. Four months have gone before we are done with all that and when we come back to our initial project the nightmare starts: we have forgotten the detection threshold we used, the precise way to get the initial guesses for our non-linear model fitting routine, etc. We can even sometimes shamefully realize that we are unable to reproduce our former analyzes. The solution is clear, we have to also use a compendium for our "daily" data analysis. The benefits of the approach will quickly show up. For instance if after we performed the long analysis of 10 experiments we decide to modify the analysis procedure slightly by applying, say, a new test, we will typically have to modify a few lines of our vignettes thereby becoming able to perform a modified analysis in a *uniform* way on our whole experiments set. In addition, in such a situation, the vignettes of the different experiments will very often be the same except for the names of the input data files.

If it can be irksome to reproduce our own work, what to say about reproducing the work of someone else, like the work of a predecessor in a laboratory. The compendium turns out to be a precious analysis support when someone has to take over the project of someone else. In particular compendia enable a member of the laboratory who is leaving to easily transmit his/her codes, data and analysis sequences to the person who is going to continue his/her work. It is thus a very welcome aid to transferring knowledge and know-how within a laboratory.

Applying the compendium paradigm requires some efforts to learn new tools and some discipline in the routine analysis practice. But the reward is an unparalleled precision in the communication and explanation of what is done with data to produce results. We think it is formidable tool to strengthen research claims outside the laboratory and to keep consistent analysis practices within the laboratory. We can only hope that our readers will consider the idea and adopt it.

5 Appendix

Methods

Experimental procedure

The experimental methods, that is, slice preparation and recording procedures, are the same as in [9, Delescluse and Pouzat, 2005] where they are described in detail.

Briefly, we performed loose cell-attached recordings of Purkinje cells (PCs) in rats cerebellar (sagittal) slices (P11-P25), at room temperature, to get spontaneous spike trains of these cells. These recordings were performed in normal saline condition (control condition) and with 25μM bath applied bicuculline (bicuculline condition); they provide the single cell data analyzed in this paper.

We also performed multi-site extracellular recordings of neighbouring PCs in the same preparation (P8-P25) in the 2 same conditions, in order to get simultaneously recorded, spontaneous PC spike trains; these recordings provide the multiple cells data analyzed in this paper.

Data analysis

For a given data set, this analysis has two stages: spike train reconstruction (*i.e* spike detection and, for multi-unit data, spike-sorting) followed by the spike train analysis *per se*.

Spike detection and classification Spikes are detected on the raw traces of the first four recording sites of the electrode. Their waveforms are extracted, reduced (12 amplitude samples,

i.e 3 *per* site, are kept for each spike) and whitened, as described in [31]. Then, spike-sorting is performed on the reduced, whitened spikes using the software *SpikeOMatic*[12].

This software implements two different algorithms, whose use depends on the analyzed data. The first and simplest algorithm is an Expectation-Maximization (EM) algorithm which maximizes the likelihood function assuming a multivariate Gaussian model [31]. To be efficient, this method requires the spike waveforms of the recorded neuron to be stationary. Such was the case in the present data set. We therefore resorted to this algorithm to process these data. When spike waveforms of the recorded neurons are non-stationary and/or when spike waveforms of different neurons get confused because of background noise, the second algorithm performs much better [9]. This Markov Chain Monte Carlo (MCMC) algorithm enables its user to work with a more realistic data generation model that accounts for non-stationary spike waveforms and neuronal discharge statistics. The mathematical and algorithmic foundations of this second method have been described [29], as well as its performances on real bursty data [9].

Spike train analysis: outline of the analysis After data collection and spike-sorting we are left with multi-spike trains which will be taken as the "raw data" of the present study. The analysis presented aims at uncovering potential correlations in PC firing induced by simultaneous inhibitory post-synaptic conductances evoked by common pre-synaptic interneurons. In order to keep this report short we will target discharge regimes where these potential correlations are easier to detect and model. That is, situation where a reasonably realistic, but non trivial, null hypothesis can be defined. That practically means that we will study *stationary* discharge regimes where individual PC firing can be well modeled by a *renewal* process [15, 4, Johnson, 1996; Brown, 2005]. Because the whole discharge of PCs cannot be so described we will have to select part of the data. That will be the occasion to illustrate a key feature of the compendium. Namely, although the whole selection procedure would not appear in the "printed manuscript", it will be explicitly present in the vignette *and therefore open to scrutiny and critics*. Once such data sections have been identified and test of the discharge model adequacy performed, correlations can be measured with classical methods: cross-correlograms, "time-to-neighbor" histograms [16, Johnson and Kiang, 1976], etc. These observed statistics can then be compared with the expected ones under the null hypothesis of no correlation. But more can also be done by estimating the *sensitivity* of these statistics. That is, data can be simulated with realistic - even stronger than realistic - correlated inputs in the observed neurons, the strength of these inputs can be modulated and their consequence on the cross-correlogram can be studied.

The routines used for this analysis are available as a R package: *SpikeTrainAnalysiswithR*, *STAR*[13].

Single spike train analysis Estimates of the instantaneous firing frequency of a neuron are obtained by convolving its spike train with a Gaussian kernel of adjustable width (whose default value in STAR is 1 s). Stationary periods are then selected, that is, periods during which the instantaneous firing rate remains constant. These stationary periods are kept for the rest of the analysis.

We test the statistical independence of successive inter-spike intervals (ISIs) in a stationary spike train by comparing the distribution of ISIs conditioned on the previous ISI length with the distribution of all the ISIs (of the selected stationary period). This comparison is done with a Quantile-Quantile (Q-Q) plot that shows the quantiles of one distribution against the ones of the other. If two samples are drawn from the same distribution, this graph should lie along the first bissectrice. Confidence boundaries are obtained with a Monte Carlo method as described by [38, Venables and Ripley, 2002, pp 86-89].

[12]This software is freely available, along with tutorials, on: http://www.biomedicale.univ-paris5.fr/C_Pouzat/SOM.html

[13]http://www.biomedicale.univ-paris5.fr/C_Pouzat/STAR.html

If a spike train passes this test, it is compatible with a *renewal process* that can be entirely described by its ISI distribution [15, 4]. An estimate of the latter is obtained by constructing the ISI histogram. Several models are moreover fitted to the ISI sample by the method of maximum likelihood. Fitted models are then compared with the Akaike Information Criterion (AIC) [2, 3]. Finally Q-Q plots of the empirical ISI quantiles against the fitted model quantiles allow a precise evaluation of the goodness-of-fit of these models.

Multiple spike trains analysis Temporal correlations between spike trains are studied in a pairwise manner with cross-correlograms (CCs) and time-to-former histograms (TTFHs) [16].

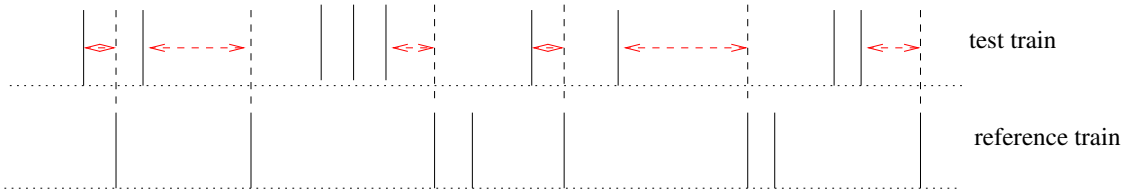

Figure 1: Time-to-former definition. Time intervals considered to build a time-to-fromer histogram are shown with dashed double arrows.

A time-to-former histogram $\mathcal{H}$ is a histogram of the time intervals between the spikes of a reference train and the spikes of a test train immediately preceding them (Fig. 1). Under the null hypothesis of independence of the two spike trains, the expected value of this histogram is given by [16]:

$$\bar{\mathcal{H}}(t) = \nu_{test} \cdot \int_t^\infty p_{test}(\tau)\, d\tau \tag{1}$$

where ν_{test} is the frequency of the test train and $p_{test}(\tau)$ is its ISI density. Its variance is given by:

$$VAR(\mathcal{H}(t)) \approx \frac{\bar{\mathcal{H}}(t)}{N.\delta} \tag{2}$$

where N is the total number of ISIs used to construct the histogram and δ is its binwidth [15].

We compute the expected value of the histogram under the null hypothesis using the ISI histogram of the test train as an estimate of its ISI density. We then compare this expected value to the real value of the histogram computed with the reference and test trains.

Besides the TTHF we plot the Q-Q plot opposing the distribution of the time-to-former intervals of the two spike trains considered against this expected distribution expected under the null hypothesis to decide if they are identical or not. Confidence intervals are computed with the bootstrap method.

CCs and TTFHs are two ways of detecting spike-to-spike correlations between two spike trains. By comparing them in both conditions - with and without interneurons inhibition (control and bicuculline conditions) - we find out the influence of interneurons on spike-to-spike correlations between PCs discharges.

Simulation of interacting spike trains

All the spike trains generated are renewal processes whose underlying ISI distribution are log-normal, possibly modified to simulate interactions between spike trains (see below). The parameters of these log-normal densities are chosen to reproduce the empirical ISI histograms of the two types of cells that we record from (PC and interneuron).

To simulate interacting spike trains we generate a first train that we call “pre-synaptic”, which stands for an interneuron. This train is built from a sequence of ISIs drawn from a log-normal density.

We then simulate a post-synaptic PC spike train from its conditional intensity with a thinning algorithm [27, 4]. This intensity $\lambda(t)$ is computed as follows. It is first derived from the ISI log-normal density $p_{lnorm}(t)$ with [8]:

$$\lambda_{lnorm}(t) = \frac{p_{lnorm}(t)}{1 - \int_{u_{N(t)}}^{t} p_{lnorm}(u)\, du} \tag{3}$$

for $t > u_{N(t)}$, where $u_{N(t)}$ is the time of the last spike prior to t. Following Kass and Ventura [17] the effect of presynaptic spike onto the postsynaptic neuron discharge is modeled by multiplying $\lambda_{lnorm}(t)$ by a factor $m(t)$ of the form:

$$m(t) = 1 + \delta \cdot \exp(-\frac{t - t_{last\_pre}}{\tau}) \tag{4}$$

where t_{last_pre} is the time of the last pre-synaptic spike preceding time t, $\delta \geq -1$ stands for the synaptic strength and τ is a time constant.

The final intensity is then:

$$\lambda(t) = \lambda_{lnorm}(t) \cdot m(t) \tag{5}$$

This simulates the effect of the pre-synaptic neuron spike on the probability discharge of the post-synaptic neurons. A negative value of δ mimics an inhibitory synapse, whereas a positive one mimics an excitatory synapse. In particular, when a pre-synaptic spike prevents the post-synaptic cell from spiking, δ is set to -1. The case $\delta = 0$ simulates independence between the pre- and the post-synaptic train.

This intensity is bounded and we note R_{max} its maximum. The thinning algorithm simulating a train of bounded intensity $\lambda(t)$ proceeds in two stages. First a Poisson train with constant rate R_{max} is generated. Then each spike i of this Poisson train is kept with probability $q_i = \lambda_i / R_{max}$, where λ_i is the value of the intensity $\lambda(t)$ at the time of spike i.

Two post-synaptic trains can be generated with the same pre-synaptic train to simulate the inhibition of two different PCs by a single interneuron.

The routines implementing these methods of simulation are also part of our R package STAR.

Results

********** Here starts the 1st Sweave extract **********

Single cell recordings

We start by loading our analysis library, as well as the single cell data of our compendium into our analysis environment:

```
> library(STAR)
> library(DelesclusePouzatDemoPaper)
> data(single.spike.train)
```

Then a call to function `summary` gives us a short reminder of what is in the data:

```
> summary(single.spike.train)
```

```
------------------------
Epoch control :
------------------------

  Begin: 0 s
```

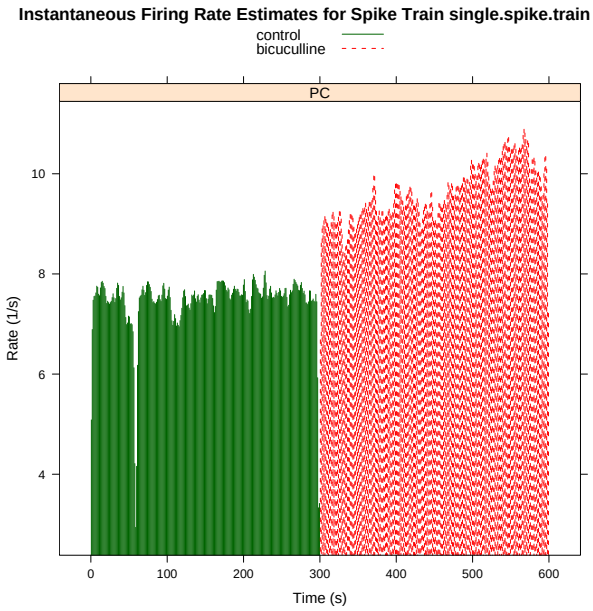

Figure 2: Rate evolution of a PC in control conditions and with $25\mu M$ bath applied bicuculline.

```
  End: 300 s

   Nb spikes Mean rate (Hz)
PC      2232           7.44


------------------------
Epoch bicuculline :
------------------------

  Begin: 300 s
  End: 600 s

   Nb spikes Mean rate (Hz)
PC      2888       9.626667
```

Rate evolution We see that a Purkinje cell (PC) was recorded during two epochs (control and bicuculline) of 5 minutes (300 s) each. We can then get a bird-eye view on the data by calling function `rateEvolution` which generates estimates of instantaneous neuronal discharge frequency of this neuron by convolving the spike train with a Gaussian kernel whose default width is 1 s:

```
> rateEvolution(single.spike.train)
```

In Fig. 2 we see immediately the net increase of the average firing rate upon application of bicuculline. On this figure we also identify periods of stationarity in both conditions: 150-300 s in control condition and 350-450 s in bicuculline condition.

********** Here ends the 1st Sweave extract **********

********** Here starts the 2nd Sweave extract **********

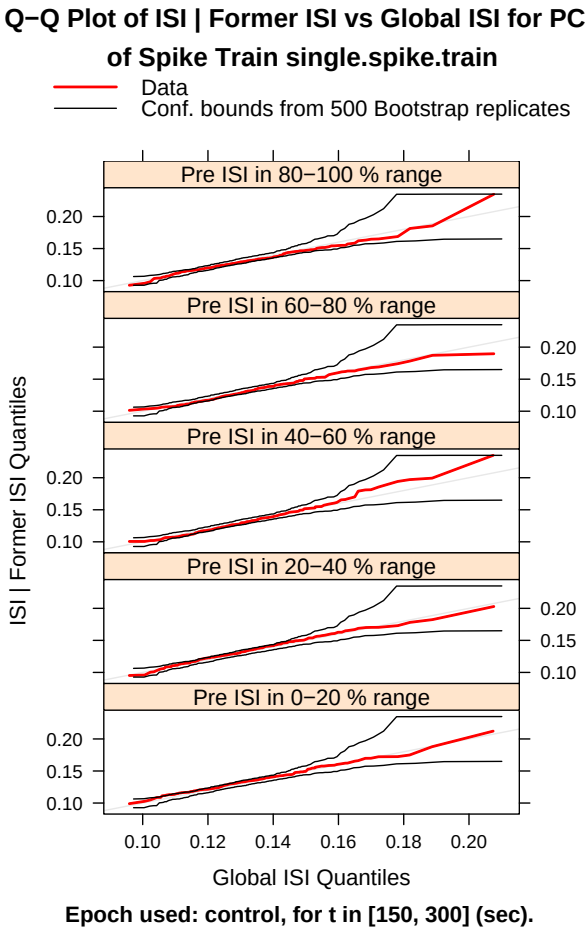

Figure 3: Q-Q plots of ISIs conditioned on the previous ISI duration *vs* the whole ISI distribution (control condition).

Statistical independence of successive ISIs in the control condition We test the statistical independence of successive ISIs of this PC in control condition. For that test we select a period of data that we previously identified as stationary (150-300 s in control condition). This data selection appears explicitly in argument `time.window`.

```
> renewalTest(single.spike.train, epoch = "con", time.window = c(150,
+     300))
```

These Q-Q plots show that, over the selected period, the ISIs conditioned to the previous ISI duration have the same distribution as the whole sample of ISIs (Fig. 3). This indicates that successive ISIs are *statistically independent* over this period. It allows us to assume that this spike train is a *renewal process* that can be completely described by its ISI distribution.

********** Here ends the 2nd Sweave extract **********

********** Here starts the 3rd Sweave extract **********

Fit of the ISI distribution in the control condition We then fit the ISI distribution of the selected period of the control spike train with 4 different models (Maximum Likelihood Estimator, MLE):

```
> ctlModel <- isiMle(single.spike.train, epoch = "con", time.window = c(150,
+     300), isi.density = c("lnorm", "invGauss", "weibull"))
```

********** HERE ENDS THE 3RD SWEAVE EXTRACT **********

The object `ctlModel` contains the fitted parameters, the values of the maximal log-likelihood and the Akaike Information Criterion (AIC) values for all models:

```
> summary(ctlModel)

Maximum likelihood estimation

Analysis of Neuron, PC, of SpikeTrain, single.spike.train, during epoch(s), control.

In the following time window:
  [150,300]

Model Comparison:
           logLik df       AIC
lnorm    3072.869  2 -6141.738
invGauss 3070.020  2 -6136.040
weibull  2835.761  2 -5667.522

Coefficients for the best model:
          Estimate  Std. Error
meanlog -2.0392243 0.003624978
sdlog    0.1216935 0.002562684
```

According to the AIC, the best model is the log-normal density but the inverse Gaussian is almost as good. We can test the goodness-of-fit of this model with a Q-Q plot opposing the fitted log-normal density to the actual distribution of ISIs (Fig. 4).

```
> qqTest(ctlModel, "lnorm")
```

One sees in Fig. 4 that the Q-Q plot remains within the confidence interval, except for a couple of "too long" ISIs, which means that the log-normal density fits well the empirical ISI distribution of the stationary period selected in the control condition.

The mode of the fitted log-normal density is (in s):

```
> as.numeric(exp(coef(ctlModel@mle.list$lnorm)[1])/exp((coef(ctlModel@mle.list$lnorm)[2])^2))

0.1282167
```

Fit of the ISI distribution in the bicuculline condition The same analysis for the bicuculline spike train leads to the same conclusions: this train can be considered as a renewal process whose ISI distribution can very well be modeled with a log-normal density (not shown). Like in the control condition, both inverse Gaussian and log-normal fit almost equally well the ISI distribution, the inverse Gaussian having this time a slightly smaller AIC value. The value of the mode of the fitted log-normal is now (in s):

```
> as.numeric(exp(coef(bicuModel@mle.list$lnorm)[1])/exp((coef(bicuModel@mle.list$lnorm)[2])^2))

0.1046261
```

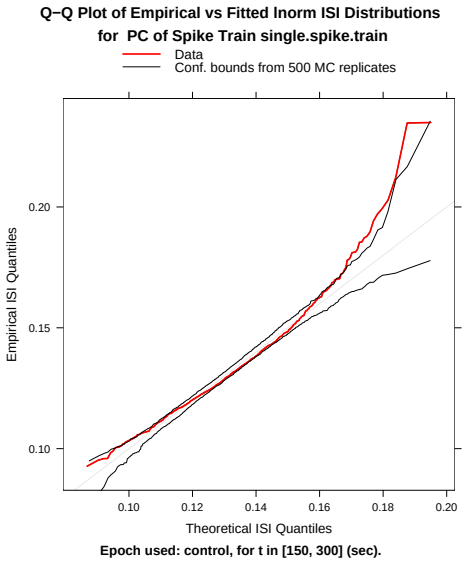

Figure 4: Q-Q plot of the fitted log-normal density against the ISI distribution (control condition).

One finds that the ISI distribution in the bicuculline spike train is shifted to the left (smaller mode value) with respect to the ISI distribution of the control spike train. This reflects the already observed increase in firing frequency of the PC when bicuculline blocks $GABA_A$ inhibition.

Plotting the ISI histograms of the PC spike train in both conditions allows to quickly appreciate this shift (Fig. 5):

```
> isi.histograms(single.spike.train, nb.bins = 100, xsup = 0.25,
+     time.window = c(150, 300, 350, 450))
```

Analysis of other PCs We now determine if this shift of the ISI distribution is significant in 17 PCs. We use here functions specifically written for this compendium. These functions summarize ISI statistics of several PCs in both conditions. We have already loaded the compendium (1st Sweave extract) to get the data. The compendium is now useful for the specific functions it contains.

********** Here starts the 4th Sweave extract **********

In the compendium, there is a set of 10 single PC spike trains recorded in "old" rats aged between 21 and 25 days post-natal (including the PC studied so far) and a set of 7 single PC spike trains recorded in "young" rats aged between 11 and 15 days post-natal. These sets are named "PC" and "jPC" respectively.

We load the first data set:

```
> load.data("PC")
```

We then compute and plot the mean, standard deviation (SD) and coefficient of variation (CV) of the 10 single PC ISIs from the old rats data set, in control and bicuculline conditions (Fig. 6):

```
> stats.list <- isi.stats("PC")
> display.isi.stats(stats.list)
```

We see in Fig. 6 that, for each PC, the ISI mean is smaller in bicuculline than incontrol condition. We also see that the ISI SD decreases in bicuculline with respect to the control condition.

To find out whether these observed changes are significant or not, we perform a one-sided t-test and a one-sided F-test respectively. The ISIs of PCs are not normally distributed but the ISI samples are large, so that a t-test and an F-test are still relevant.

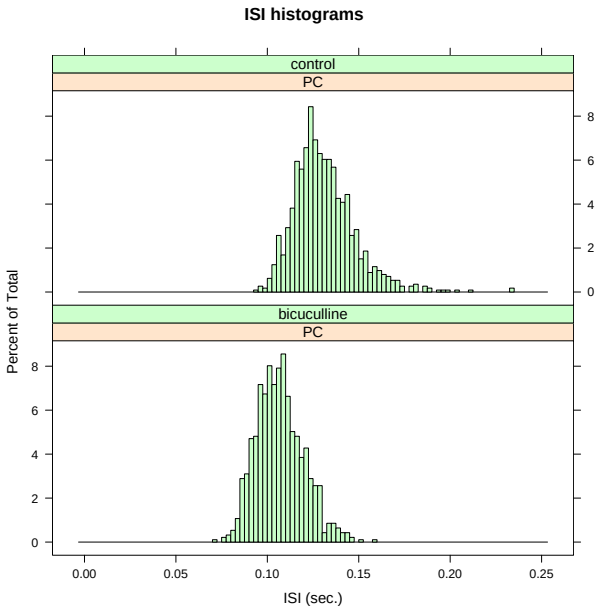

Figure 5: ISI histogram of the PC spike train in control (*top*) and bicuculline (*bottom*) conditions. Note the shift of the ISIs towards smaller values with bath applied bicuculline.

```
> PC.t.test("PC")
```

```
The null hypothesis is that ISI means do not differ in both conditions
Here are the p-values for the 10 PCs of this data set:
0.0004614775  0  7.819276e-50  5.341818e-204  0  0  5.962936e-267  0  0  0
```

```
> PC.var.test("PC")
```

```
The null hypothesis is that the ISI variance is greater in bicuculline condition
Here are the p-values for the 10 PCs of this data set:
0.02422295  0  0  0  0  0  0  0.999938  0  0
```

All p-values are almost null, so that the null hypothesis of equal means for the control and bicuculline ISI distribution is rejected. For all PCs, the ISI mean in bicuculline is significantly smaller than in control. This also holds for the ISI variance which significantly decreases in bicuculline, except for the 8th PC.

Similar results are obtained with the data set of "young" PCs (not shown).

********** Here ends the 4th Sweave extract **********

We compute the average relative shift of the ISI mean over all the PCs of the first data set:

```
> m.difference <- stats.list$Cish.m - stats.list$Bish.m
> m.difference <- m.difference/stats.list$Cish.m
> mean(m.difference)
```

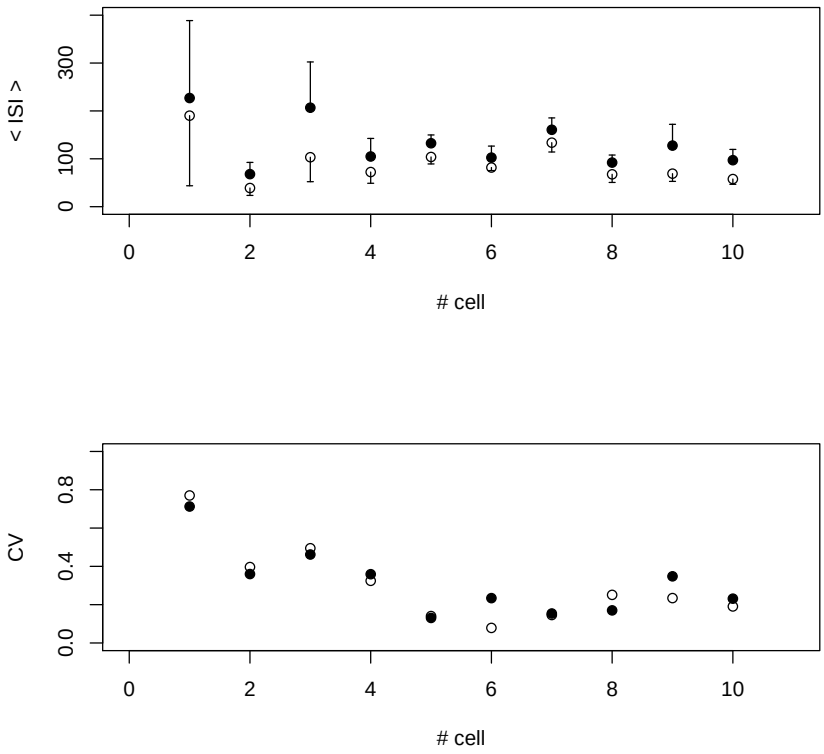

Figure 6: 21-25 days old rats. *Top*: mean ISI values with SDs for 10 PCs in control (filled circles) and bicuculline (open circles) conditions. The difference between both values is significant for all cells (see text). Ordinate axis in ms. *Bottom*: coefficient of variation (CV) for the same 10 PCs in control (filled circles) and bicuculline (open circles) conditions.

```
[1] 0.3115408
```

```
> sd(m.difference)/sqrt(length(m.difference))
```

```
[1] 0.04022511
```

The average decrease in ISI mean with bicuculline therefore amounts to 31 (+/-4) % ($n = 10$) for this data set.

We also compute the average relative decrease of the ISI SD over this data set:

```
> sd.difference <- stats.list$Cish.sd - stats.list$Bish.sd
> sd.difference <- sd.difference/stats.list$Cish.sd
> mean(sd.difference)
```

```
[1] 0.3466072
```

```
> sd(sd.difference)/sqrt(length(sd.difference))
```

```
[1] 0.08004766
```

The average decrease in ISI SD in bicuculline amounts to 35 (+/-8) % ($n = 10$) for this data set.

As a result of similar decreases of the mean and SD of the ISI distribution in bicuculline, there is no systematic effect on the CV values, as seen in the preceding figure: 4 CVs are smaller in bicuculline, 4 are larger and 2 do not change.

In the other data set, the average decrease in ISI mean with bicuculline amounts to 36 (+/-7) % ($n = 7$) and the average decrease in ISI SD amounts to 45 (+/-12) % ($n = 7$) (not shown).

Multiple cells analysis

Multiple PC spike trains

We now tackle the analysis of multiple PC spike trains simultaneously recorded. We load the multiple cells data of our compendium into our analysis environment:

```
> data(multi.spike.train)
```

These data contain 8 PCs that were simultaneously recorded during two epochs (control and bicuculline) of 5 minutes (300 s) each, similarly to the single PC data of the previous section. We call function `rateEvolution` to see the evolution of instantaneous firing frequencies of these neurons (Fig. 7).

```
> rateEvolution(multi.spike.train)
```

In Fig. 7 we see immediately the net increase of the average firing rate of all neurons, except neuron 7, upon application of bicuculline, as well as a very clear slow temporal pattern change. This ensures that the known bicuculline effect on PC spike trains was there during this multi-unit recording.

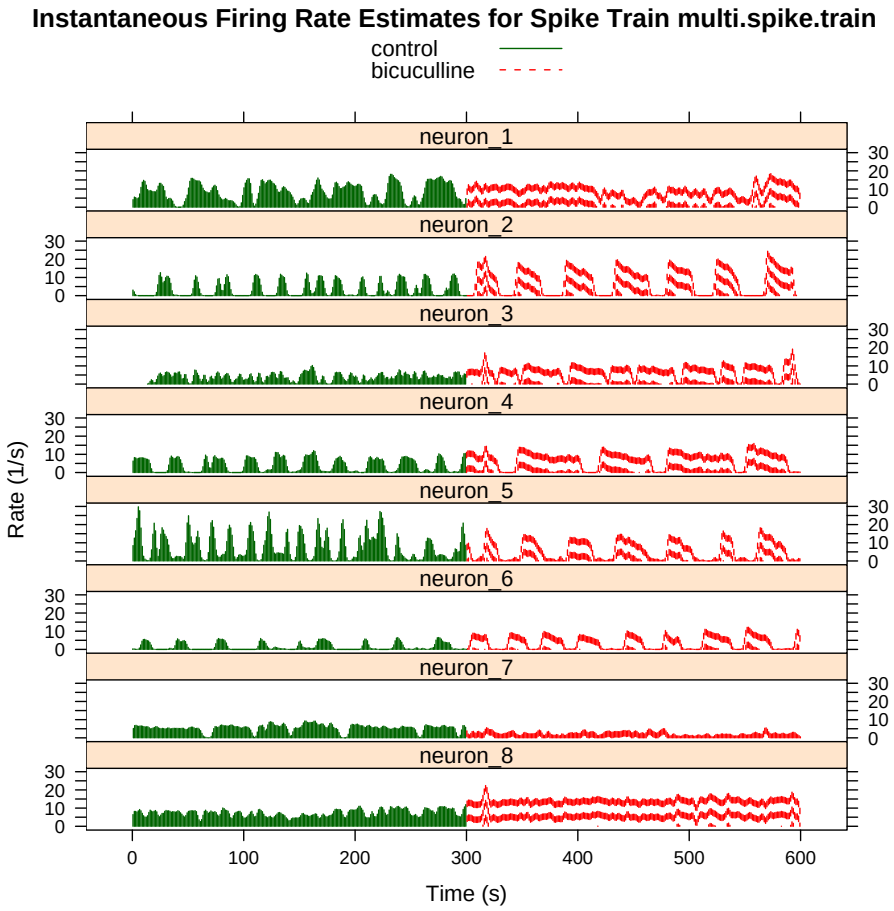

Figure 7: Rate evolution of 8 simultaneously recorded PCs in control conditions and with $25\mu M$ bath applied bicuculline.

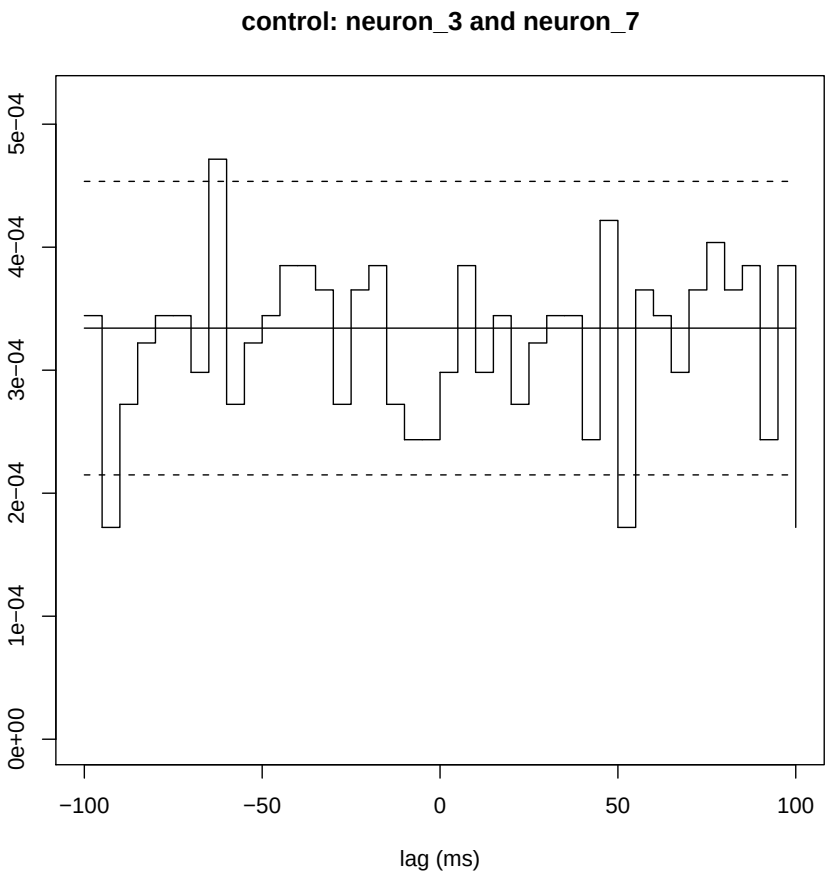

Figure 8: Cross-correlogram of a pair of neurons (control condition).

Cross-correlations in control condition

Cross-correlograms We now plot the CCs of some of these spike trains in the control condition (Fig. 8). To plot CCs in the control condition we call function `cross.correlograms`, specifying the neurons whose spike trains are going to be cross-correlated, as well as the name of the epoch ("control"). Based on Fig. 7 we select a period of time where the firing of these neurons is stationary. This selection is done through argument `time.window`:

```
> cross.correlograms(multi.spike.train, epochs = "control", neurons = c(3,
+     7), time.window = c(195, 255), bin.width = 0.005, max.lag = 0.1)
```

Contrarily to what we expected, the CCs show no obvious spike-to-spike correlation (Fig. 8). They remain flat and within the confidence interval around the expected value in case of independence. The reader can plot other CCs for other pairs of neurons. This necessitates to select periods of times where both spike trains are stationary. The reader will find that all CCs are flat.

When CCs are computed over periods where spike trains are not stationary, one sometimes obtain CCs that remain slightly and uniformly above the expected value in case of independence of the two spike trains, but within the confidence interval. And so does the TTHF (the corresponding Q-Q plot remains next to the lower confidence bound for all quantiles). This indicates a slight positive correlation between the evolutions of their firing rates. In Fig. 7 indeed one sees that some neurons, like neurons 2 and 3 in control condition, have some loosely concomitant frequency increases and decreases. This phenomenon can also be observed in bicuculline (with neurons 2 and 4 for instance) and therefore *is not due to common inhibition of these PCs*.

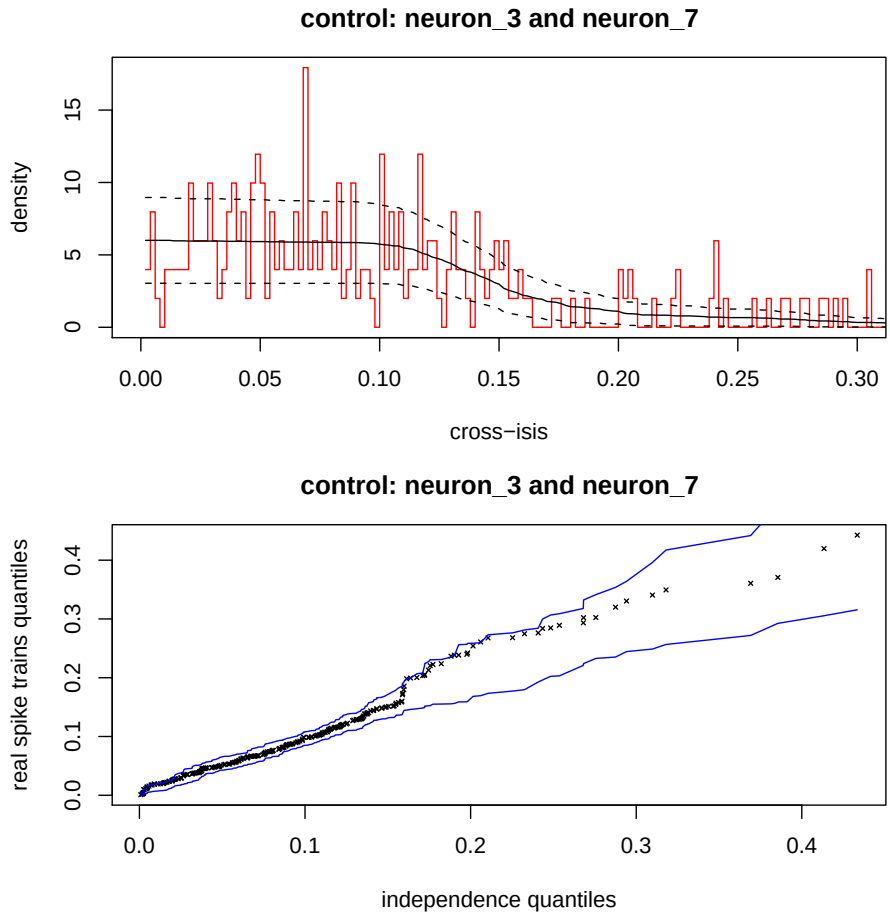

Figure 9: Time-to-former histogram of a pair of neurons and corresponding Q-Q plot (control condition).

Time-to-former histograms We plot the TTHFs of some of the neurons over a selected period of stationarity in the control condition (Fig. 9). Besides each histogram we display the Q-Q plot of the distribution of the real time-to-former intervals against the one expected under the null hypothesis of independence.

```
> ttf.histograms(multi.spike.train, epochs = "control", neurons = c(3,
+     7), time.window = c(195, 255), bin.width = 0.002, xsup = 0.3,
+     nb.shuffles = 100)
```

Like the CCs in this condition (Fig. 8), the TTFHs show no obvious spike-to-spike correlation (Fig. 9). The Q-Q plots remain along the first bissectrice over all the quantiles, confirming that the the time-to-former intervals of the pairs of neurons can be drawn from the distribution computed under the null hypothesis of independence.

Cross-correlations in bicuculline condition

Cross-correlograms We plot the CC of a pair of neurons over a common period of stationarity in the bicuculline condition (Fig. 10):

```
> cross.correlograms(multi.spike.train, epochs = "bicuculline",
+     neurons = c(3, 4), time.window = c(360, 400), bin.width = 0.005,
+     max.lag = 0.1)
```

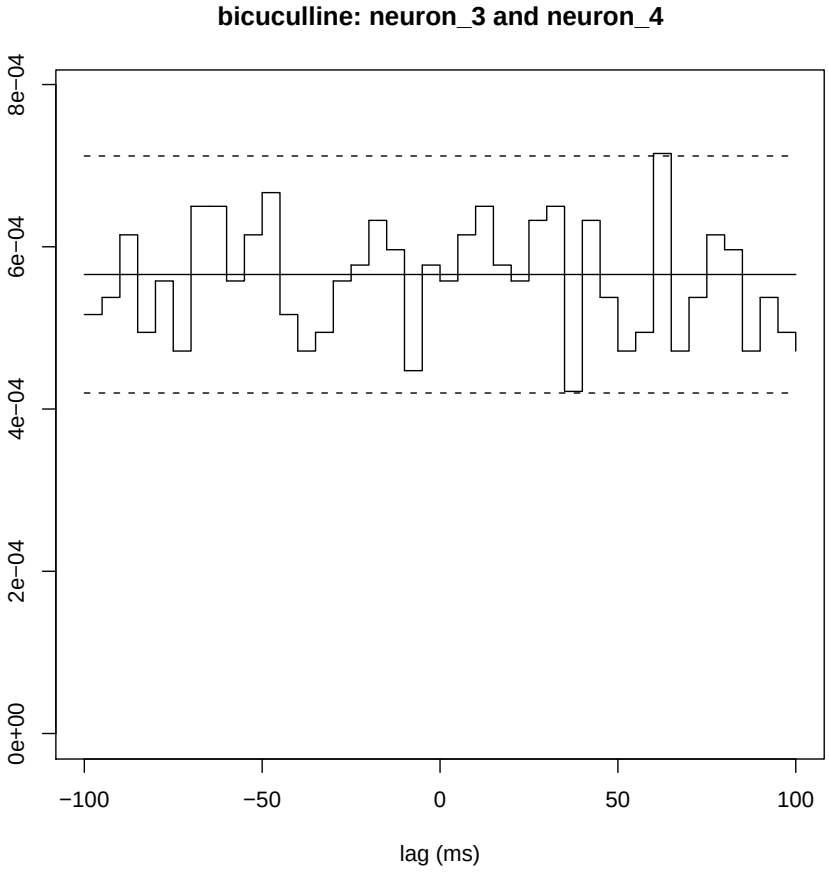

Figure 10: Cross-correlogram of a pair of neurons (bicuculline condition).

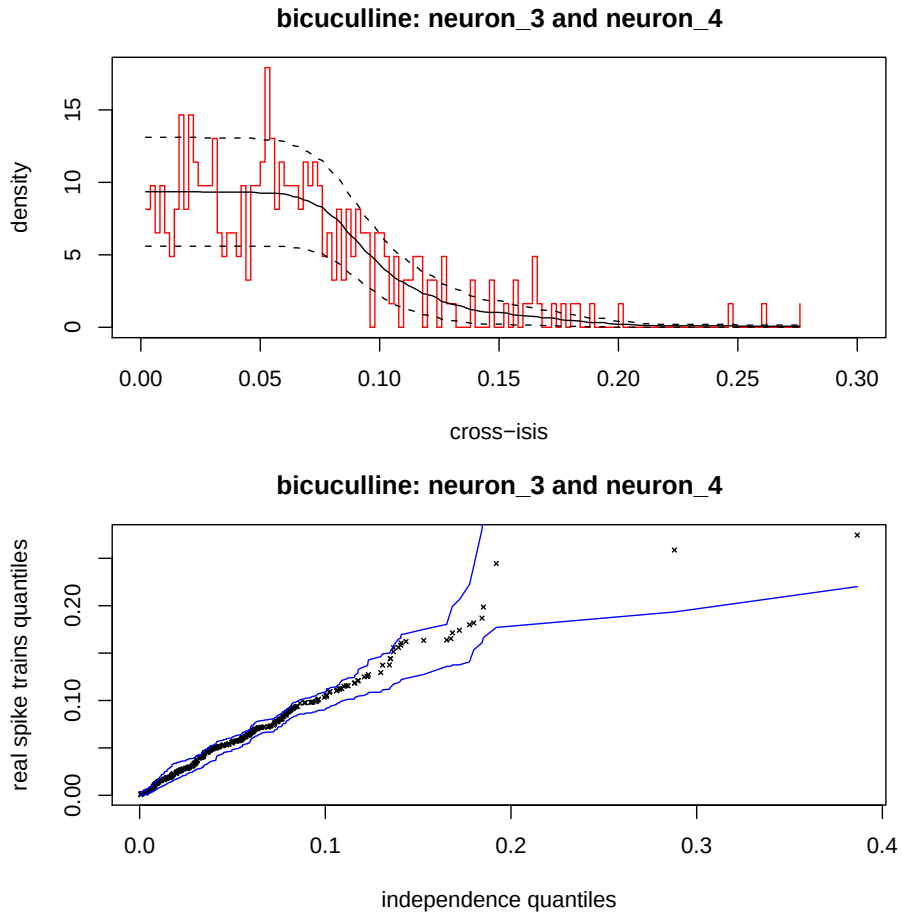

Figure 11: Time-to-former histogram of a pair of neurons and corresponding Q-Q plot (bicuculline condition).

Similarly to the control condition the CCs in bicuculline show no obvious spike-to-spike correlation. This is not surprising as the only possible source of correlation in this preparation, namely GABAergic inhibition by interneurons (and possibly other PCs), has been blocked by the $GABA_A$ receptor antagonist.

The remark made in the control condition about possible loosely concomittant firing frequency changes still holds in bicuculline, when CCs are computed over periods of non-stationarity.

Time-to-former histograms We plot the TTHF of a pair of neurons over a common selected period of stationarity in the bicuculline condition (Fig. 11):

```
> ttf.histograms(multi.spike.train, epochs = "bicuculline", neurons = c(3,
+     4), time.window = c(360, 400), bin.width = 0.002, xsup = 0.3,
+     nb.shuffles = 100)
```

Like the CCs in this condition (Fig. 10), the TTFHs show no obvious spike-to-spike correlation (Fig. 11). The Q-Q plots remain along the first bissectrice over all the quantiles, confirming that the the time-to-former intervals of the pairs of neurons can be drawn from the distribution computed under the null hypothesis of independence.

Conclusion

The effect of bicuculline on individual PCs has been described in the first section "Single Cell recordings". This effect is still present in our multi-unit recordings, as assessed by the net increase in frequency for all recorded neurons (Fig. 7). However, no change in their cross-correlations can be observed. None of the CCs and TTHFs shows significant spike-to-spike correlation between any spike trains neither in control condition, in accord with a previous report [10], nor in the presence of bicuculline. Similar results were obtained in 12 other experiments in rats aged between 8 and 25 post-natal days. Therefore, we are unable to detect any influence of interneuron inhibition on PC spike-to-spike correlations.

Simulations

********** Here starts the 5th Sweave extract **********

To understand these rather unexpected negative results, we need to explore quantitatively the sensitivity of the tools we used to analyse temporal cross-correlations between spike trains. For that purpose we simulate interacting spike trains.

Common inhibition

We first simulate an interneuron train with log-normal scale and shape parameters equal to 0.3 s and 0.05 respectively:

```
> para.pre <- list(c(0.3, 0.05))
> pre.train <- simul.one.epoch(para.pre, "lognormal", "epoch",
+     0, 120)
```

We then simulate two PC trains that are post-synaptic to this interneuron train. The log-normal scale and shape parameters of these PCs are respectively $(0.06s, 0.3)$ and $(0.08s, 0.3)$. We make the pre-synaptic spike prevents the post-synaptic cells from firing ($\delta = -1$), with a time constant $\tau = 40ms$ for the exponential relaxation.

```
> para.post <- list(c(0.06, 0.3), c(0.08, 0.3))
> post.trains.in <- simul.postsynaptic.trains(parameters = para.post,
+     presynaptic.train = pre.train, post.type = c("lognormal",
+         "lognormal"), delta = -1, tau = 0.04, epoch.name = "simul_inhibition")
```

We plot the CCs of the pre-synaptic and post-synaptic trains (Fig. 12).

```
> cross.correlograms(post.trains.in, epochs = "simul_inhibition",
+     bin.width = 0.005, max.lag = 0.1)
```

********** Here ends the 5th Sweave extract **********

The direct inhibitions of the two post-synaptic trains by the pre-synaptic one are seen on their respective CCs (first two CCs in Fig. 12). In this simulation, each pre-synaptic spike has a very strong effect on the post-synaptic trains since it immediately sets the intensities of the latter to zero, with an exponential relaxation whose time constant equals $40ms$. This value is obviously an upper bound of the duration of an IPSC effect on the post-synaptic spiking.

Despite this strong common inhibition the CC of the two post-synaptic trains remains flat (third CC in Fig. 12): *the common inhibitory inputs are not detected at all with this technique.* Likewise the TTFH of both post-synaptic cells does not show any significant deviation from independence, whereas a clear deviation is seen for the TTHFs computed with each post-synaptic train against the pre-synaptic one (not shown).

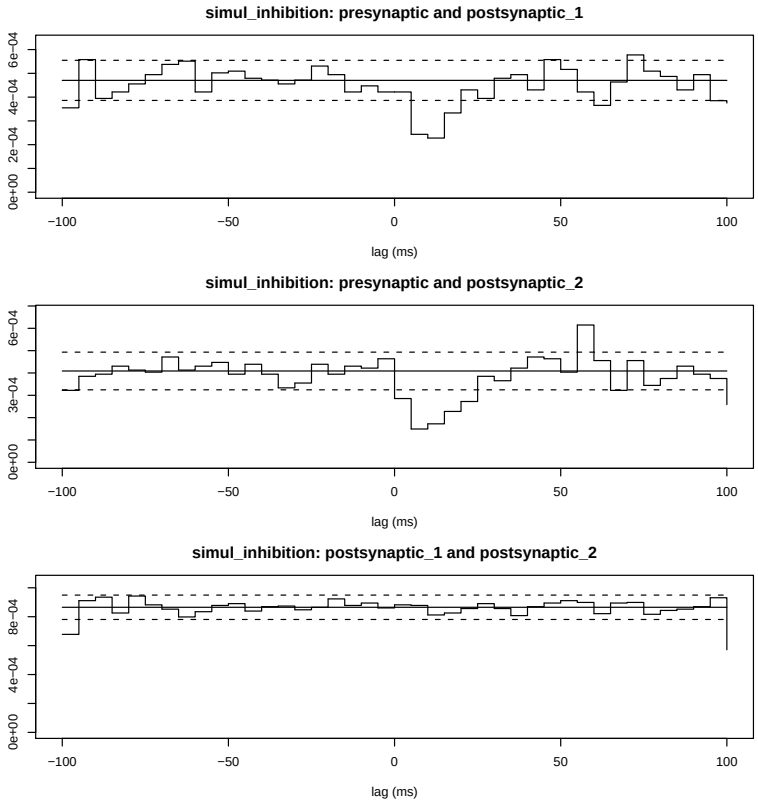

Figure 12: Cross-correlograms of the pre-synaptic train with each post-synaptic train (*top* and *middle*) in the case of simulated inhibition. *bottom*: cross-correlogram of the two post-synaptic cells.

Common excitation

Now we consider the synapses between the pre- and post-synaptic trains to be excitatory. We simulate two PC trains with the same log-normal parameters as in the inhibition simulation but with a positive value for δ. Choosing a small value for the time constant τ ($e.g$ $10ms$) makes the common excitation window narrow, increasing the efficacy of the synchronization of the post-synaptic spike trains. In the frame of our interneuron-PC system, this simulates a possible post-inhibitory excitation generating rebound spikes in the post-synaptic PCs at the end of their simultaneous inhibitory inputs from their common interneuron.

```
> post.trains.ex <- simul.postsynaptic.trains(parameters = para.post,
+     presynaptic.train = pre.train, post.type = c("lognormal",
+         "lognormal"), delta = 5, tau = 0.04, epoch.name = "simul_excitation")
```

We plot the CCs of the pre-synaptic and post-synaptic trains (Fig.13).

```
> cross.correlograms(post.trains.ex, epochs = "simul_excitation",
+     bin.width = 0.005, max.lag = 0.1)
```

The same comments as in the case of common inhibition hold in this case. The direct excitations of the post-synaptic trains by the pre-synaptic one are clearly seen on their respective CCs (first two CCs in Fig. 13). In this simulation, each pre-synaptic spike has a very strong effect on the post-synaptic trains, since it multiplies their intensities by 6, with an exponential relaxation whose time constant equals $10ms$ which makes the two post-synaptic cells fire simultaneously in a narrow time window.

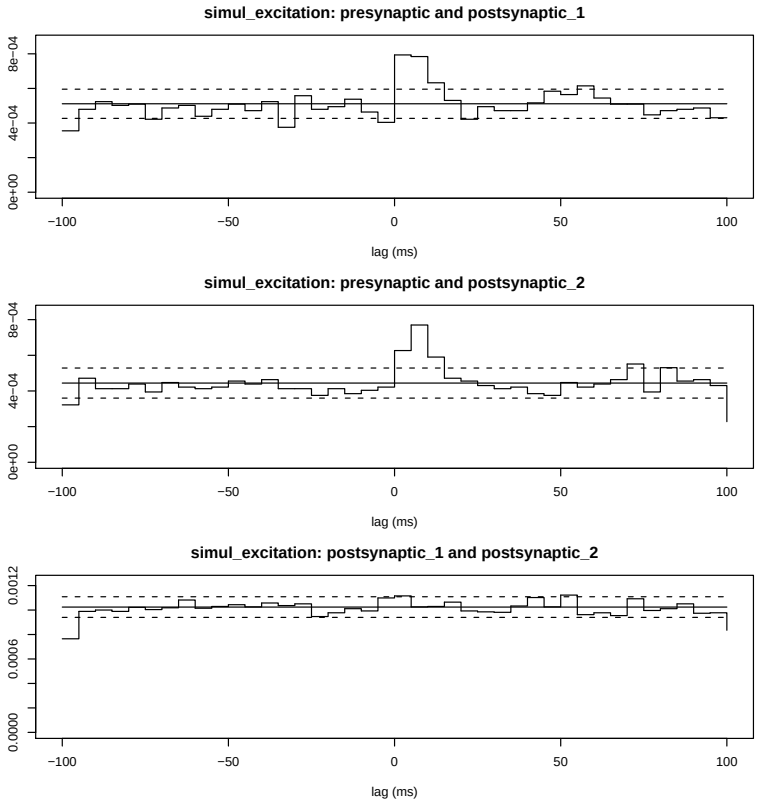

Figure 13: Cross-correlograms of the pre-synaptic train with each post-synaptic train (*top* and *middle*) in the case of simulated excitation. *bottom*: cross-correlogram of the two post-synaptic cells.

Once again, despite this strong common excitation, the CC of the two post-synaptic trains remains flat (third CC in Fig. 13): the common excitatory inputs are not detected at all at that level with this technique. Likewise the TTFH of the post-synaptic trains does not show any significant deviation from independence, whereas a clear deviation is seen for the TTHFs computed with each post-synaptic train against the pre-synaptic one (not shown).

Discussion

PCs and interneurons are spontaneously active in cerebellar slices [24] [23]. At room temperature they respectively fire at about 10 and $3Hz$. In this paper we studied the spontaneous activity of groups of neighbouring PCs in this preparation by means of extracellular multisite electrodes. In the Introduction section, we summarized several results published in the literature indicating that these activities should exhibit temporal correlations due to *common inhibition* by pre-synaptic interneurons.

Indeed the cited papers together lead to the conclusion that several neighbouring PCs could have *delayed action potentials simultaneously* during spontaneous activity. This contributes to induce temporal correlations between their discharges. However no temporal correlation is to be seen between spontaneous spike trains of PCs in CCs and TTHFs, whether in presence or in absence of GABAergic inhibition. In both conditions these plots do not significantly differ from the ones obtained with independent spike trains. These techniques therefore fail to detect any particular temporal relation that would be caused by common inhibitory inputs.

These results do not imply that these expected correlations do not exist. The simulations we performed clearly show that *common inputs, whether inhibitory or excitatory, do usually not appear in CCs*

and TTHFs, unless unreasonably shaped and strong. The results of these simulations run counter the usual idea that common inputs are easily observed in CCs. They show that there is no way CCs can detect any correlation induced by common inhibitory or excitatory inputs when the pre-synaptic cell is firing at a third of the firing rates of the post-synaptic cells (as it is the case with interneurons and PCs in our preparation), at least for the recording duration considered. One has to design other tests to make common inputs stand out.

In fact, CCs are globally worse at detecting *common* than *direct* inputs in two cells. This observation is obvious in Fig. 12 and Fig. 13: on each of these figures, direct inputs are clearly seen on the first 2 CCs, whereas common inputs do not on the third CC. This is due to the fact that in the case of direct inputs, *the events of the pre-synaptic point processe directly inflence the temporal positions of the events of the post-synaptic point process*, whereas this does not hold in the case of common inputs only.

Aertsen and Gerstein (1985) [1] studied the sensitivity of CCs in detecting *direct* connections between neurons. They showed that the CCs were much less sensitive (an order of magnitude) to direct inhibition than they were to direct excitation. Their simulations indeed demonstrate the very bad visibility of direct inhibition in CCs: if CCs are good at representing an excess of events of a train relative to the events of the other train, they are obviously less good at detecting a lack of them. The authors also quantify this difference in sensitiviy in terms of connection strength, relative pre and post firing rates, recording duration and in terms of CC bin width. They show that direct inhibition is seen in CCs only when all pre-synaptic spikes silence the post-synaptic cells, firing frequencies being comparable in pre- and post-synaptic cells. If only 80% of them prevent the post-synaptic the cell from firing, the total number of spikes recorded has to be very much increased (that is in fact the "recording" time has to be increased from $256s$ to $4096s$ in their simulation) to make the trough appear in the CC. The authors relate the bias in favor of excitatory connections found with cross-correlations techniques in the literature to these results. Our experience supports Aertsen and Gerstein's observations, although our purpose here was not to quantify this difference in sensitivity. Repeated simulations do show that in general CCs of simulated pre-synaptic and post-synaptic trains with direct inhibition have relatively weak troughs (often above the lower confidence limit) compared to the peaks observed in the case of direct excitation.

References

[1] AMH Aertsen and GL Gerstein. Evaluation of neuronal connectivity: Sensitivity of cross-correlation. *Brain Research*, 340:341–354, 1985.

[2] H Akaike. A new look at the statistical identification model. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[3] Riccardo Barbieri, Michael C. Quirk, Loren M. Frank, Matthew A. Wilson, and Emery N. Brown. Construction and analysis of non-Poisson stimulus-response models of neural spiking activity. *Journal of Neuroscience Methods*, 105(1):25–37, 2001.

[4] Emery N. Brown. *Methods and Models in Neurophysics*, chapter Theory of Point Processes for Neural Systems, pages 691–726. Elsevier, 2005.

[5] J Buckheit and DL Donoho. *Wavelets and Statistics*, chapter Wavelab and reproducible research, pages 55–81. Springer-Verlag, New-York, a antoniadis and g oppenheim edition, 1995.

[6] John Chambers. Computing with Data: Concepts and Challenges. *The American Statistician*, 53(1):73–84, 1999.

[7] John M. Chambers. *Programming with Data: A Guide to the S Language*. Springer, 1998.

[8] D Daley and D Vere-Jones. *An introduction to the theory of point process*. Springer-Verlag, New-York, 1988.

[9] M Delescluse and C Pouzat. Efficient spike-sorting of multi-state neurons using inter-spike intervals information. *J Neurosci Methods*, In press, 2005.

[10] U Egert, D Heck, and A Aertsen. Two-dimensional monitoring of spiking networks in acute brain slices. *Exp Brain Res*, 142:268–274, 2002.

[11] Peter Flynn. Formatting Information. A beginner's introduction to typesetting with Latex. http://www.tug.org/tex-archive/info/beginlatex/, 4 April 2005.

[12] R Gentleman and D Temple Lang. Statistical analyses and reproducible research. *Bioconductor Project Working Papers*, Working Paper 2:http://www.bepress.com/bioconductor/paper2, 2004.

[13] Robert Gentleman. Reproducible Research: A Bioinformatics Case Study. Working Paper 3, Bioconductor Project Working Papers, 20 May 2004.

[14] M Haeusser and BA Clark. Tonic synaptic inhibition modulates neuronal output pattern and spatiotemporal synaptic integration. *Neuron*, 19:665–678, 1997.

[15] DH Johnson. Point process models of single-neuron discharges. *J Comput Neurosci*, 3:275–299, 1996.

[16] DH Johnson and NYS Kiang. Analysis of discharges recorded simultaneously from pairs of auditory nerve fibers. *Biophys J*, 16:719–734, 1976.

[17] R. E. Kass and V. Ventura. A spike-train probability model. *Neural Comput.*, 13(8):1713–1720, 2001.

[18] D Knuth. Literate programming. Center for the Study of Language and Information, Stanford, California 1992.

[19] Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physica Verlag, Heidelberg, 2002. ISBN 3-7908-1517-9.

[20] Friedrich Leisch. Sweave, part I: Mixing R and LaTeX. *R News*, 2(3):28–31, December 2002.

[21] Friedrich Leisch. Sweave and beyond: Computations on text documents. In Kurt Hornik, Friedrich Leisch, and Achim Zeileis, editors, *Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria*, 2003. ISSN 1609-395X.

[22] Friedrich Leisch. Sweave, part II: Package vignettes. *R News*, 3(2):21–24, October 2003.

[23] I Llano and HM Gerschenfeld. Inhibitory synaptic currents in stellate cells of rat cerebellar slices. *J Physiol*, 468:177–200, 1993.

[24] R Llinas and M Sugimori. Electrophysiological properties of in vitro purkinje cell dendrites in mammalian cerebellar slices. *J Physiol (Lond)*, 305:197–213, 1980.

[25] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion, 2nd Edition*. Addison-Wesley, 2nd edition, 22 April 2004.

[26] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. The Not So Short Introduction to LATEX2e. http://www.ctan.org/tex-archive/info/lshort/english/, 8 May 2005.

[27] Y Ogata. On lewis' simulation method for point processes. *IEEE Transactions on Information Theory*, IT27:23–31, 1981.

[28] SL Palay and V Chan-Palay. *Cerebellar Cortex, Cytology and Organization*. Springer-Verlag, Berlin, 1974.

[29] C Pouzat, M Delescluse, P Viot, and J Diebolt. Improved spike-sorting by modeling firing statistics and burst-dependent spike amplitude attenuation: a markov chain monte carlo approach. *J Neurophys*, 91:2910–2928, 2004.

[30] C Pouzat and S Hestrin. Developmental regulation of basket/stellate cell->purkinje cell synapses in the cerebellum. *J Neurosci*, 17:9104–9112, 1997.

[31] C Pouzat, O Mazor, and G Laurent. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. *J Neurosci Methods*, 122:43–57, 2002.

[32] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-07-0.

[33] R Development Core Team. *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-11-9.

[34] A. J. Rossini. Literate statistical analysis. In Kurt Hornik and Friedrich Leisch, editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing, Vienna, Austria*, 2001. ISSN 1609-395X.

[35] Anthony Rossini and Friedrich Leisch. Literate Statistical Practice. UW Biostatistics Working Paper Series 194, University of Washington, 2003.

[36] Anthony Rossini, Martin Maechler, Kurt Hornik, Richard M. Heiberger, and Rodney Sparapani. Emacs Speaks Statistics: A Universal Interface for Statistical Analysis. Technical Report 173, University of Washington, 17 July 2001. http://www.bepress.com/uwbiostat/paper173.

[37] M Schwab, M Karrenbach, and J Claerbout. Making scientific computations reproducible. *Computing in Science & Engineering*, 2:61–67, 2000.

[38] William N. Venables and Brian D. Ripley. *Modern Applied Statistics with S. Fourth Edition*. Springer, 2002. ISBN 0-387-95457-0.

[39] P Vincent and A Marty. Neighboring cerebellar purkinje cells communicate via retrograde inhibition of common presynaptic interneurons. *Neuron*, 11:885–893, 1993.

[40] P Vincent and A Marty. Fluctuations of inhibitory postsynaptic currents in purkinje cells from rat cerebellar slices. *J Physiol (Lond)*, 494:183–199, 1996.

Chapitre 6

Discussion et perspectives

6.1 L'approche MCMC du *spike-sorting* : bilan et perspectives

6.1.1 Un nouveau cadre de travail

Le premier article a montré qu'il était possible de faire l'analogie entre le problème du *spike-sorting* des électrophysiologistes et celui de la restauration d'image. Plus généralement, ces deux problèmes sont des cas particuliers d'un modèle de Potts formalisé en Physique Statistique. Cette analogie a permis d'adapter au *spike-sorting* les solutions développées depuis plusieurs décennies pour étudier ces modèles ou pour résoudre le problème de la restauration d'image. Ces solutions font appel à une classe de méthodes qui sont l'objet d'une intense activité de recherche depuis vingt ans au sein de la communauté des statisticiens : les méthodes de Monte Carlo par chaînes de Markov (MCMC). Cette approche du *spike-sorting* est radicalement nouvelle et offre un cadre de travail général cohérent et fécond.

L'application de la méthodologie MCMC que nous proposons permet de considérer un modèle de génération de données plus réaliste et plus complexe que ceux considérés traditionnellement. En particulier, notre algorithme permet de prendre en compte, dans la classification, l'*information temporelle* contenue dans les trains de potentiels d'action (PAs), et non plus uniquement l'information fournie par la *forme* des PAs. Pour ce faire, l'un des volets du modèle sur lequel est réalisée l'inférence porte sur la statistique de décharge des neurones enregistrés. Il s'agit de la première méthode automatique de *spike-sorting* capable d'utiliser l'information des temps des PAs pour réaliser sa classification. En ce sens, notre méthode fait un saut qualitatif important par rapport aux méthodes existantes (voir le chapitre 3 pour une revue de ces méthodes). Le franchissement de cette étape majeure dans le domaine est d'ailleurs attendu depuis longtemps et permet de lever certaines difficultés récurrentes du *spike-sorting* et décrites dans le chapitre 3 (section 3.8).

La première de ces difficultés est la *non-stationnarité des formes de PAs d'un neurone donné*. En se fondant exclusivement sur la forme d'un PA pour en déterminer le neurone d'origine, les méthodes existantes s'interdisent d'attribuer le même neurone à

deux PAs de formes différentes. La prise en compte, par notre algorithme, des temps écoulés entre PAs permet de modéliser la dépendance de leur amplitude vis-à-vis de ces temps : la modélisation de cette dépendance est l'objet du second volet de notre modèle de génération de données.

En second lieu, l'information fournie par les temps des PAs se révèle précieuse lorsque ceux-ci ont, malgré leurs origines neuronales différentes, des formes proches. Les méthodes de *spike-sorting* utilisées ont généralement recours à la délimitation - automatique ou manuelle - de nuages de points dans l'espace de représentation des PAs choisi au préalable (voir chapitre 3). La similarité des formes de PAs de deux neurones différents se traduit par un *recouvrement* des nuages correspondants. Il est dès lors vain de vouloir séparer correctement les PAs des deux neurones dans ces régions de recouvrement si l'on ignore l'information que fournissent leurs temps d'émission. Le test de notre algorithme sur les données simulées du premier article montre que la prise en compte de cette information temporelle permet de lever l'incertitude de la classification, même lorsque les recouvrements de nuages sont importants. Il est à noter que l'utilisation d'électrodes multisites dans les expériences limite ce problème : en fournissant plusieurs points de vue pour un même signal, ces électrodes permettent une meilleure différenciation des formes des signaux. Il est cependant impossible d'éviter systématiquement tout recouvrement de nuages et le problème resurgit avec plus ou mois d'acuité suivant les données.

Par ailleurs, notre méthode permet d'obtenir une distribution *a posteriori* des paramètres du modèle de génération de données, ainsi que des labels (neurones d'origine) pour chaque PA. En général, pour de nombreux PAs, l'un des labels a une probabilité égale à, ou voisine de, 1. Et c'est heureux. Cependant, les PAs de certains neurones peuvent être difficiles à séparer (recouvrement de nuages, voir ci-dessus) ; en toute rigueur, il faut tenir compte des probabilités d'attribution à différents neurones de ces PAs dans l'étude des corrélations entre trains, afin d'éviter le biais que constitue une classification définitive et rigide.

6.1.2 Un logiciel libre

Dans de très nombreuses préparations, les électrophysiologistes enregistrent des neurones émettant des *bursts* de PAs d'amplitude décroissante. Ce type de données n'a jamais pu être analysé automatiquement jusqu'à présent, pour les raisons que j'ai rappelées plus haut. Nous avons montré, dans le deuxième article, que notre méthode pouvait réaliser une classification automatique très fiable de telles données. Pour cela, nous avons sophistiqué le modèle de la statistique de décharge des neurones par rapport au premier article et sommes passé d'une unique densité log-normale à un modèle de Markov (HMM) caché à 3 états (3 log-normales), pour modéliser les densités empiriques des intervalles entre PAs (ISIs) de ces neurones. Cet article montre donc la pertinence d'un HMM pour modéliser la décharge de cellules émettant des *bursts*, et il valide expérimentalement notre méthode de *spike-sorting*. Ce faisant, il montre la généralité et la flexibilité de cette dernière : il est possible d'avoir recours à des modèles de génération de données différents si les données d'enregistrements de cellules uniques l'exigent ; la méthode reste la même et l'algorithme ne doit pas être entièrement réécrit.

Tous les algorithmes que nous utilisons pour réaliser l'ensemble des étapes nécessaires au *spike-sorting*, de la détection des événements sur la trace brute à la représentation des résultats du *spike-sorting*, sont disponibles gratuitement sous la forme d'un logiciel R, *SpikeOMatic*, accompagné d'un manuel d'utilisateur qui fait l'objet du troisième article. L'algorithme MCMC à proprement parler est distribué sous la forme d'une librairie de fonctions C intégrée au logiciel. Le logiciel implémente également un algorithme d'*Expectation-Maximization* (EM) classique, dont les performances en matière de *spike-sorting* peuvent être satisfaisantes pour certaines données, notamment si les PAs de tous les neurones sont stationnaires et si les recouvrements de nuages ne sont pas trop importants. Le manuel d'utilisateur, l'aide en ligne dans R et l'interface graphique dans cet environnement, doivent rendre l'utilisation de ce logiciel accessible à d'autres personnes que ses auteurs. Il n'est en particulier pas nécessaire de connaître le détail des méthodes qui y sont implémentées.

6.1.3 Les développements en cours

La *méthode d'échange de répliques* (REM) implémentée dans notre algorithme pour résoudre le problème de sa relaxation lente suppose la simulation de plusieurs chaînes de Markov à différentes températures. Ceci rend notre méthode très exigeante computationnellement. Cependant, il est possible de réaliser ces chaînes de Markov en parallèle : la parallélisation du code à l'aide de l'interface MPI (*Message Passing Interface*) constitue la prochaine étape du développement.

Par ailleurs, sous sa forme actuelle, l'algorithme MCMC ne détermine pas le nombre de neurones présents dans les données. Ce nombre doit être donné en entrée par l'utilisateur. Le choix du nombre de neurones le plus pertinent s'effectue en comparant les résultats du *spike-sorting* obtenus avec les différentes valeurs essayées. Cependant, dans le cadre de notre méthode, il est possible de comparer quantitativement les modèles avec différents nombres de neurones : cette comparaison peut être réalisée par la méthode d'Ogata-Tanemura dite de l'*intégration thermodynamique* (Ogata, 1989). Elle nécessite la simulation de plusieurs chaînes de Markov à différentes températures. On a évidemment ici recours aux chaînes simulées dans le cadre du REM, qui se révèle donc utile à double titre : échapper à la relaxation lente et réaliser l'intégration thermodynamique. Nous avons implémenté cette dernière sur des modèles simples comportant un mélange simulé de plusieurs densités log-normales ; ces premiers résultats, qui n'ont pas été exposés ici, nous montrent qu'il est possible de retrouver le bon nombre de densités avec cette méthode. Son incorporation dans notre algorithme n'est pas encore terminée. Une fois qu'elle le sera, nous disposerons d'une méthode de *spike-sorting* générale, cohérente et complète. Il faut noter ici qu'aucune méthode, à ma connaissance, ne détermine actuellement le nombre de neurones présents dans les données automatiquement de façon satisfaisante. L'expérience que nous avons faite du critère d'information bayésienne (BIC) dans le cadre des modèles de mélanges gaussiens nous a montré qu'il surestime systématiquement, souvent largement, le nombre de nuages présents. C'est donc en définitive l'expérimentateur qui doit décider de ce nombre. En ce sens, si l'intégration thermodynamique se révèle satisfaisante pour l'estimation du nombre de neurones, notre méthode de *spike-sorting* aura levé cette difficulté récurrente supplémentaire.

Le logiciel *SpikeOMatic* est également destiné à s'intégrer dans un ensemble d'outils complet et cohérent, constitué d'une matrice de microélectrodes, d'un système d'acquisition et d'un logiciel d'analyse des données multiunitaires. *SpikeOMatic* sera au coeur de ce dernier. Ce projet ambitieux est en cours de développement[1].

6.2 L'absence de corrélations entre activités spontanées des cellules de Purkinje

Ces développements méthodologiques ont eu pour support expérimental le système des cellules de Purkinje (PCs) dans les tranches de cervelet de rats. C'est en particulier sur ce système que notre méthode MCMC a été validée expérimentalement.

Parallèlement à ces développements, l'analyse des trains de PAs des PCs obtenus a été réalisée afin d'étudier précisément les corrélations qui peuvent apparaître durant l'activité spontanée de ces cellules. Celles-ci sont soumises à l'activité inhibitrice spontanée des interneurones de la couche moléculaire et des autres PCs. Des travaux antérieurs montrent que les interneurones ont une influence directe et visible sur les décharges de leurs PCs postsynaptiques ; d'autres travaux ont démontré que deux PCs partagent une inhibition commune d'un (ou plusieurs) interneurone(s) présynaptique(s) ; d'autres enfin ont mis en évidence anatomiquement et fonctionnellement l'existence d'une inhibition directe PC-PC (voir le chapitre 2 pour une revue de tous ces travaux). Dans le chapitre 2, nous avons conclu que les corrélations entre décharges spontanées de deux PCs, induites par l'inhibition qu'elles reçoivent, pouvaient *a priori* prendre trois formes différentes (section 2.2.5) : (i) la synchronisation des PAs en rebond après un courant postsynaptique inhibiteur (IPSC) commun, (ii) la présence d'ISIs prolongés concomitants, (iii) l'inhibition directe d'une PC par une collatérale récurrente de l'autre PC. L'hypothèse (i) d'une synchronisation des PCs par les interneurones, en particulier, apparaît dans de très nombreuses introductions et discussions de la littérature.

Le quatrième article montre l'effet notable du blocage de la transmission gabaergique inhibitrice par la bicuculline sur les décharges individuelles spontanées des PCs : leur fréquence et leur régularité augmente en l'absence d'inhibition par les interneurones. Cependant, aucun effet n'est visible sur les *relations temporelles* entres les trains de PAs des PCs. Les histogrammes de corrélation croisée (CCs) restent plats avec ou sans inhibition ; l'analyse des évolutions de fréquences des trains au cours du temps ne permet de dégager de véritables groupes de cellules coopératives dans aucune condition.

L'absence de corrélation constatée entre les trains de PAs spontanés de PCs en présence d'inhibition peut avoir deux causes différentes : (i) ces corrélations n'existent pas ; cette hypothèse est difficile à soutenir compte tenu des travaux cités dans le chapitre 2 et des conclusions qui en ont été tirées, (ii) nos méthodes d'analyse ne permettent pas de détecter cette inhibition commune.

La première explication vaut certainement pour une partie des données analysées. En effet, la probabilité d'enregistrer deux PCs subissant une inhibition commune d'un

[1]Quatre laboratoires portent actuellement ce projet, avec les équipes de MM. Meyrand, Yvert, Boraud et Pouzat.

interneurone ayant une activité spontanée soutenue et provoquant des courants synaptiques importants dans les deux cellules postsynaptiques est sans doute moins élevée qu'on pourrait l'imaginer. Il est très vraisemblable que dans nombre de données analysées, il n'y ait effectivement pas de PCs qui reçoivent une inhibition commune ou qui soient directement connectées, auquel cas aucune corrélation d'activité n'est attendue.

Cependant, il est difficile de concevoir qu'aucune des expériences menées n'ait permis d'enregistrer l'activité de deux PCs recevant une inhibition commune d'un même interneurone ou d'interneurones électriquement connectés. Et c'est évidemment la deuxième explication qui doit avoir notre faveur. Les simulations de trains de PAs réalisées dans le quatrième article reproduisent tout à la fois la statistique de décharge spontanée des PCs et l'inhibition qu'elles peuvent recevoir de la part des interneurones, telle qu'elle est établie et décrite dans la littérature. Il apparaît que cette inhibition commune simulée est trop faible pour être détectée sur un CC ou sur un histogramme des temps du précédent. Plus précisément, *la fréquence des IPSCs communs à deux PCs est trop faible au regard de la fréquence spontanée de décharge des PCs* pour apparaître sur ces graphiques. Ainsi, notre analyse n'est pas en mesure de détecter cette inhibition commune, même si elle est effectivement présente dans les données.

Il est nécessaire de décomposer cette conclusion globale selon les trois types de corrélations attendus et rappelés au début de cette section. Aertsen et Gerstein (1985) ont montré que les CCs étaient *considérablement moins sensibles à la présence d'une connexion inhibitrice directe entre deux cellules* qu'ils ne le sont pour une connexion excitatrice directe : les CCs font plus facilement apparaître un excès qu'un manque de PAs dans une cellule après les PAs de l'autre cellule. Sans entrer dans les détails quantitatifs qui y sont donnés, cet article démontre qu'il faut une beaucoup plus grande quantité de données et une connexion synaptique beaucoup plus forte entre les deux cellules pour constater un creux significatif dans leur CC que pour y constater un pic. Ces auteurs indiquent que ce fait frappant est sans aucun doute l'une des sources du biais en faveur des connexions excitatrices effectivement constaté dans la littérature.

Cette constatation peut expliquer pourquoi les CCs n'ont pas fait apparaître de connexion inhibitrice directe entre PCs (forme (iii) des corrélations attendues). Compte tenu de la fréquence typique d'une PC dans notre préparation ($10\,Hz$) et de la fréquence typique des IPSCs spontanés d'une PC ($10\,Hz$), si la synapse PC-PC était efficace, tous les IPSCs d'une PC seraient dus à une seule PC présynaptique. Or, la fréquence de décharge spontanée typique d'un interneurone est de $2\,Hz$ et le nombre de cellules en panier présynaptiques à une PC est de l'ordre de 5ou 6 (voir chapitre 2). On constate donc qu'en première approximation tous les IPSCs spontanés d'une PC peuvent être expliqués par l'activité des cellules en panier présynaptiques. Dans une PC, les IPSCs spontanés dus à une PC présynaptique sont donc vraisemblablement peu fréquents et les CCs entre PCs ne sont pas assez sensibles pour faire apparaître cette inhibition directe.

Sur cette question de l'inhibition PC-PC, j'ai réalisé des expériences supplémentaires sur des souris avec David Orduz au laboratoire. Leur analyse n'a pas été reproduite ici. Brièvement, nous avons enregistré extracellulairement les activités spontanées de PCs avec l'électrode multisite, selon le même protocole que celui décrit dans la section 4.1. Simultanément, nous réalisions un enregistrement cellule-entière d'une PC possiblement

postsynaptique aux PCs vues par l'électrode multisite, en mode *voltage-clamp*. Aucune corrélation entre l'un des trains "présynaptiques" enregistrés extracellulairement et les IPSCs de 6 PCs enregistrées intracellulairement au cours de 6 expériences différentes n'a pu être mise en évidence. Avec une moyenne de 6 à 8 PCs enregistrées extracellulairement pour 1 PC intracellulaire, nous avons couvert au moins 36 paires de PCs potentiellement connectées. Ces expériences confirment donc que l'inhibition directe PC-PC est peu fréquente.

Quant au phénomène d'*inhibition commune* noté (ii) ci-dessus, il est encore plus subtil et difficile à voir sur un CC que l'*inhibition directe* au sein d'une paire : il ne s'agit pas de constater un manque significatif de PAs d'un neurone relativement à ceux de l'autre neurone. Il s'agit de détecter la présence d'ISIs simultanément prolongés par des IPSCs communs (pauses communes). Ce type de corrélation, dans la mesure où il ne se manifeste pas par une *une relation temporelle étroite entre les deux processus ponctuels croisé*s, est encore moins visible sur un CC que ne l'est l'inhibition directe des événements d'un processus par ceux de l'autre processus. Les simulations du quatrième article démontrent que les ISIs concomitants prolongés par des IPSCs communs à deux PCs n'apparaissent pas dans les CCs. Seule la répétition de pauses longues et simultanées devient visible sur un CC.

Dans la troisième forme de corrélation que l'on pouvait attendre, et notée (i) ci-dessus, l'inhibition commune se traduit en quelque sorte par une excitation retardée des deux cellules postsynaptiques. Dans cette hypothèse, l'*excitation commune* finale contribue à synchroniser les PAs de la paire de PCs. Les CCs sont performants pour détecter toute synchronisation, ils ont été conçus pour cela. Il n'en demeure pas moins que la synchronisation doit atteindre un niveau suffisant pour être nettement visible sur un CC (Aertsen et Gerstein, 1985). Les simulations du quatrième article montrent que, bien qu'ils détectent rapidement une *excitation directe*, les CCs ne détectent pas aisément une *excitation commune*, et ce, contrairement à une idée intuitive répandue. En dépit des pics marqués dans les CCs entre le train présynaptique et chacun des trains postsynaptiques (excitation directe), le CC entre ces deux derniers reste plat (excitation commune). Ce qui différencie fondamentalement la détection par un CC d'une connexion directe d'une connexion commune, c'est le fait que dans le premier cas, contrairement au deuxième, *ce sont les événements de l'un des deux processus croisés qui ont une influence sur ceux du second*. Or, un CC représente les positions temporelles des événements d'un processus relativement à celles des événements de l'autre processus : il rend donc avant tout visible l'influence directe que peuvent avoir les événements d'un processus sur les positions temporelles des événements de l'autre.

Nos simulations montrent donc clairement que l'inhibition reçue par les PCs, telle qu'elle est connue, ne peut être mise en évidence par les CCs. Cela ne signifie pas qu'elle est totalement absente dans notre préparation. Ce qui est certain, c'est que *son influence sur les relations temporelles entre trains de PCs est faible* et difficilement visible par les techniques classiques d'analyse de paires de trains. En particulier, dans nos conditions expérimentales et en régimes spontanés, les interneurones ne sont pas capables de sychroniser efficacement les PAs de PCs.

Enfin, ces analyses ont été conduites sur des données obtenues dans d'autres conditions pharmacologiques avec le même résultat ; elles n'ont pas été reproduites ici. En

particulier, les cellules de Lugaro sont des neurones gabaergiques situés juste sous la couche des PCs. Elles sont actives en présence de sérotonine (Dieudonné et Dumoulin, 2000) et inhibent les PCs (Dean et al., 2003). La sérotonine n'a pas induit de corrélations détectables par nos méthodes d'analyse. La présence de noradrénaline dans la solution extracellulaire, en activant les récepteurs β-adrénergiques, provoque une nette augmentation de la fréquence des IPSCs dans les PCs (Llano et Gerschenfeld, 1993b). La noradrénaline n'a pas induit de corrélations entre les décharges spontanées de PCs. Pour finir, aucune corrélation n'est apparue entre les démarrages de *bursts* des PCs en présence de l'agoniste des récepteurs glutamate métabotropiques DHPG. Dans cette dernière condition pharmacologique, chaque PC semble donc émettre ses *bursts* selon sa propre horloge, indépendamment de ses voisines.

6.3 La reproductibilité des analyses

Le quatrième article a également pour objet un point méthodologique important, à savoir la mise à disposition, pour les lecteurs intéressés, des données, des programmes d'analyse et de l'analyse elle-même, qui ont mené aux résultats présentés dans une publication de neurophysiologie. Les outils qui permettent cette mise à disposition existent et ont été présentés dans cet article. Cette mise à disposition prend la forme d'un paquetage R, appelé *compendium*, contenant un élément-clé particulier, à savoir un métafichier exécutable, ou *vignette*, dont l'exécution reproduit la succession des analyses et des figures produites dans l'article. Plus encore, ce fichier comprend non seulement l'ensemble des commandes d'une analyse, mais aussi les textes, commentaires et descriptions, nécessaires à sa compréhension. Le produit final de l'exécution de ce fichier mêle les sorties des codes exécutés, notamment les figures, au texte traditionnel de description qui doivent les accompagner. Nous proposons d'accompagner d'un tel compendium la soumission d'un article de neurophysiologie sous sa forme finie et statique traditionnelle. Nous proposons que ce compendium soit mis en ligne par le journal, au même titre que l'article. Cette démarche améliore considérablement la communication scientifique pour plusieurs raisons. Le lecteur intéressé peut en effet :

- reproduire lui-même toute l'analyse présentée dans un article, ainsi que ses figures "non montrées", à partir des données de cet article.
- changer un ou plusieurs paramètres de l'analyse sur les données des auteurs pour en comprendre l'influence sur les résultats obtenus.
- effectuer l'analyse des auteurs sur ses propres données.
- effectuer sa propre analyse, avec ses propres codes, sur les données des auteurs.

Il est évident que les premiers des "lecteurs intéressés" devraient être les *reviewers* qui ont à juger de la qualité et de la fiabilité des travaux qui leur sont communiqués. La démarche proposée facilite considérablement la procédure d'examen des articles soumis. Cet article, tout comme le second, est donc accompagné des données et d'un compendium permettant de régénérer le papier de démonstration qu'il contient entièrement.

Les autres lecteurs intéressés sont naturellement ceux qui travaillent dans le champ de recherche exploré par l'article. Ce sont par exemple les lecteurs qui travaillent avec des données de même nature et qui voudraient utiliser les outils d'analyse décrits dans ledit article. Evidemment, nombreux sont ceux qui se contenteront de la lecture de l'article seul, mais au moins ceux qui le souhaitent auront les moyens de mieux exploiter les résultats d'un travail publié.

Par ailleurs, au sein d'un laboratoire, le compendium est un moyen efficace de préserver l'ensemble des séquences d'analyse utilisées par l'un de ses membres sur certains types de données. Cette préservation des séquences d'analyse est précieuse pour au moins trois raisons.

Tout d'abord, c'est un moyen pour le chercheur de reproduire sans effort la même analyse complète, qui peut être longue et fastidieuse, sur différents ensembles de données. La seule exécution de la vignette assure celle de tous les programmes successivement impliqués dans cette analyse. Seuls changent les noms des fichiers de données sur lesquels la vignette est exécutée. Il s'agit là d'un gain de temps et d'énergie considérable. D'autres que lui au laboratoire peuvent utiliser les mêmes analyses, éventuellement légèrement modifiées, et avoir recours au même compendium de base.

D'autre part, il est fréquent qu'un chercheur ait à revenir sur l'analyse de certaines données après plusieurs mois, voire plusieurs années, consacrés à d'autres travaux. Il est alors très appréciable de retrouver l'ensemble de l'analyse qui était alors effectuée sous la forme d'un unique document, le compendium. Ce dernier permet en ce sens une reprise facile et sans faille d'une analyse interrompue pendant quelque temps.

Enfin, la préservation des séquences d'analyse est précieuse lorsque celle-ci doit être transmise d'un membre à l'autre d'un laboratoire. En particulier, lorsqu'un doctorant ou un post-doctorant quitte le laboratoire, il est souvent fastidieux et compliqué de transmettre les outils d'analyse à celui/celle/ceux qui le sui(ven)t et doi(ven)t prendre la suite du travail. La construction d'un ou plusieurs compendium(a) permet d'organiser les différentes séquences d'analyses élaborées et/ou utilisées par celui qui part, en un ou plusieurs fichiers exécutables, assortis des commentaires et descriptions nécessaires à la compréhension. La continuité de l'analyse d'une personne à l'autre se fait plus aisément.

En dernier lieu, la vignette, tel qu'il est présenté dans le quatrième article, se prête particulièrement bien à la rédaction de manuels d'utilisateurs ou tutoriels pour un programme ou un logiciel donné. Les appels de fonction, leurs sorties et leurs descriptions y sont naturellement mêlés. C'est sous cette forme qu'a été écris le manuel de *SpikeOMatic*, objet du troisième article.

6.4 Les défis de l'analyse de trains multiples

Le travail méthodologique présenté dans cette thèse permet d'améliorer la séparation des activités de neurones voisins, dans de petits volumes de tissu, en particulier lorsque ceux-ci émettent des bursts et/ou des PAs de formes peu différentes. Cependant, les difficultés de l'analyse de trains multiples de PAs ne s'arrêtent pas à celles du

spike-sorting, comme en témoigne la partie expérimentale du présent travail et sa discussion dans la section 6.2. Une fois les différentes décharges neuronales reconstruites, l'expérimentateur se trouve face à un deuxième problème : *l'analyse multivariée des multiples trains de PAs* dont il dispose. Là encore, de nombreux travaux et développements méthodologiques sont nécessaires pour construire des outils d'analyse pertinents en la matière (Brown et al., 2004). A l'heure actuelle, l'analyse de trains multiples de PAs est presqu'exclusivement effectuée paire par paire, à l'aide des histogrammes de corrélation croisée (CCs) notamment. Cette situation n'est évidemment pas satisfaisante car les interactions entre neurones d'une population donnée ne peuvent se résumer en une collection de CCs entre toutes les paires possibles de ladite population. Il s'agit donc de développer des techniques d'analyse d'une assemblée de neurones considérée en tant que telle et non en tant qu'un ensemble de paires. Les principales questions que l'on souhaite résoudre à ce niveau sont :

- Les neurones interagissent-ils ? Si oui, comment ?
- Peut-on comprendre l'activité d'une population de neurones à partir de nos connaissances de leurs propriétés individuelles (membranaires, synaptiques) ?
- Comment les neurones individuels et les populations de neurones représentent-ils les stimuli sensoriels, les commandes motrices ?
- Les corrélations entre neurones contribuent-elles ou non à la représentation d'un stimulus ?

6.4.1 L'analyse par paires de trains

Dans le domaine temporel, l'analyse des corrélations entre les trains de PAs de deux neurones est généralement conduite comme nous l'avons fait dans ce travail (voir le chapitre 4) : CC (Perkel et al., 1967), histogramme des temps du précédent (et/ou du suivant) (Perkel et al., 1967 ; Johnson et Kiang, 1976), évolution des fréquences de décharge. Ces trois types d'analyse sont présentés dans le chapitre 4, section 4.3. Les deux premiers supposent la stationnarité des trains. Les plus utilisés, les CCs, donnent la probabilité d'émission d'un PA d'un neurone cible à différents temps relativement aux PAs d'un neurone de référence. Ils détectent donc une coïncidence (avec un possible délai) entre les PAs émis par deux neurones.

Lorsque les trains de PAs sont analysés relativement à un stimulus, on peut avoir recours à un diagramme joint des temps post-stimulus (*joint peri-stimulus time scatter diagram*) (Gerstein et Perkel, 1972), ou à sa version en histogramme normalisé (*joint peri-stimulus time histogram*, JPSTH, Aertsen et al., 1989). Il s'agit d'une extension à deux neurones de l'histogramme des temps post-stimulus (PSTH) : alors que ce dernier compte le nombre de PAs d'un neurone par unité de temps aux temps t après le stimulus, le JPSTH est un histogramme à deux dimensions qui compte les PAs des deux neurones aux temps u et v après le stimulus pour les neurones 1 et 2 respectivement. La diagonale principale du JPSTH donne, à chaque temps t après le stimulus, les PAs coïncidents (à la largeur de fenêtre près) des deux neurones. De façon générale, la sommation sur chaque diagonale du JPSTH, normalisée par la longueur de cette diagonale, est une estimation du CC de la paire.

L'analyse des corrélations au sein d'une paire peut être également réalisée dans le *domaine fréquentiel* : les transformées de Fourier des deux trains de PAs sont utilisées pour calculer les spectres de ces trains, ainsi que leur spectre croisé (ou *cohérence*). Sous l'hypothèse de stationnarité des trains, la cohérence met en évidence une possible association des deux trains pour certaines fréquences. Ce type d'analyse a été effectué au début de ce travail. Il n'a pas non plus fourni de résultats positifs et n'a pas été reproduit ici.

Dans toutes ces analyses, il est nécessaire de comparer les résultats obtenus pour les paires de trains enregistrées à ceux attendus sous l'hypothèse nulle d'indépendance des trains de cette paire. On peut calculer les intervalles de confiance sous l'hypothèse asymptotique des grands échantillons, ou encore en ayant recours à des procédures non paramétriques comme le *bootstrap* (voir section 4.3.3).

6.4.2 Les autres méthodes d'analyse de trains multiples

L'analyse des trains multiples de PAs ne peut se limiter à celle des paires de trains. Il existe potentiellement des interactions entre neurones qui ne se réduisent pas aux interactions de paires. L'intérêt des enregistrements de plusieurs décharges neuronales simultanées est justement de permettre l'étude de leur organisation temporelle d'ensemble, et en particulier les temps relatifs des PAs de tous les neurones enregistrés.

Dans cette perspective, certains auteurs ont développé des algorithmes pour détecter des patrons précis de PAs (*precise firing patterns*) au sein d'un ensemble de décharges neuronales (Abeles et Gerstein, 1988 ; Abeles et Gat, 2001). Il s'agit de mettre en évidence des *séquences particulières de PAs* qui se répètent significativement plus souvent que le hasard ne le prévoit sous l'hypothèse nulle d'indépendance des trains. Ces séquences sont constituées de plusieurs PAs (en général de 2 à 6), émis par plusieurs neurones, et séparés par des intervalles de temps précis. Selon les auteurs qui développent ces algorithmes, la reproductibilité de telles séquences pourrait être la signature d'une assemblée cellulaire particulière en activité (Hebb, 1949). Une assemblée de neurones se définit donc ici par la structure temporelle de son activité d'ensemble.

Toujours dans cette perspective ont été développées des méthodes de détection de coïncidences de PAs sur plusieurs décharges neuronales (Grün et al., 1999, 2002). Lorsque la coïncidence de PAs de plusieurs neurones se répète plus souvent que ce que le hasard prédit, ces auteurs parlent d' "événements unitaires" (*unitary events*). Il s'agit ici de détecter les synchronisations de plus de deux neurones et leur relation à des événements comportementaux.

Il est possible d'adopter une tout autre démarche : au lieu de s'intéresser aux séquences temporelles précises des PAs dans un ensemble de trains, on peut focaliser son attention sur l'évolution des fréquences de décharges de tous les neurones, comme on l'a fait pour des paires dans le chapitre 4 (section 4.3.3). Dans cette analyse, les fréquences de décharge des N neurones enregistrés sont calculées sur des fenêtres temporelles successives (de quelques dizaines à quelques centaines de ms de largeur). A chaque fenêtre temporelle correspond donc un vecteur de N fréquences. On peut alors considérer la séquence de ces vecteurs de fréquences comme la réalisation d'une chaîne de Markov ca-

chée : dans ce modèle, chaque état est défini par un vecteur de fréquences et l'ensemble des neurones enregistrés passe d'un état de la chaîne à l'autre, avec des probabilités données par des paramètres de transition (Abeles et al., 1995 ; Seidemann et al., 1996). Tous les paramètres de ce modèle de Markov caché sont ajustés avec un algorithme de Baum-Welch (Baum, 1970) maximisant la vraisemblance. Cet algorithme attribue également l'un des états du modèle de Markov caché à chaque vecteur de fréquences des données, reconstituant ainsi la séquence des états visités par les N neurones. *In vivo*, ce type d'analyse cherche à mettre en évidence des modulations dynamiques et cohérentes des décharges au sein de groupes neuronaux, qui ne sont pas nécessairement en relation avec des événements extérieurs (Seidemann et al., 1996).

Une autre approche a été proposée il y a une vingtaine d'années et est connue sous le nom de *gravitational clustering* (Gerstein et al., 1985). L'idée fondamentale de cette analyse est de traduire l'activité des N neurones enregistrés en mouvements de particules dans un espace euclidien de dimension N. Chaque neurone est représenté par une particule ponctuelle chargée dans cet espace. A chaque émission d'un PA par un neurone, la charge associée à la particule représentant ce neurone est incrémentée d'une quantité définie ; entre deux PAs cette charge décroît. La force qu'exerce une particule sur tout autre particule est proportionnelle au produit de leurs charges respectives ; cette force peut, en outre, dépendre de la distance euclidienne qui sépare les particules. La force résultante qui s'exerce sur une particule affecte directement sa vitesse. Ces forces tendent à aggréger en un nuage (*cluster*) les particules qui correspondent à des neurones qui déchargent ensemble. Différents nuages représentent différents groupes de neurones "coopératifs".

Ces différents types d'analyse ont le mérite de traiter les N neurones enregistrés comme un tout constitué et non comme une simple somme de paires. Cependant, leur utilisation par les expérimentateurs reste modeste, sans doute parce qu'elles n'ont pas permis, jusqu'à présent, d'obtenir de résultat biologique majeur. Aucune, à mon sens, n'a vraiment pu faire la démonstration de sa réelle pertinence dans les analyses de données qui y ont eu recours.

6.4.3 Le décodage de trains de PAs

L'analyse de décodage des trains de PAs procède en deux étapes : une étape d'encodage et une étape de décodage. L'étape d'encodage caractérise l'activité neuronale multiple comme fonction de stimuli extérieurs. L'étape de décodage réalise l'opération inverse : le signal est estimé à partir de l'activité neuronale. Plusieurs agorithmes de décodage ont été développés (vecteur de population, corrélation inverse, décodage bayésien) qui permettent d'étudier comment les patrons des trains de PAs d'un ensemble de neurones peuvent représenter des stimuli ou des signaux biologiques (Brown et al., 1998). Ces algorithmes ont par exemple été utilisés pour caractériser de quelle façon des groupes de neurones peuvent représenter des commandes motrices, ou la position de l'animal dans un environnement donné. Ils sont utilisés en particulier dans la réalisation et le contrôle de prothèses neurales et d'interfaces cerveau-machine (Warland et al., 1997 ; Wessberg et al., 2000 ; Serruya et al., 2002).

6.4.4 Perspectives

Les méthodes et outils d'analyse de trains de PAs multivariés sont donc encore, dans une large mesure, à construire. Le cadre conceptuel et mathématique des *processus ponctuels* offre les perspectives les plus prometteuses (Brown et al., 2004). Les trains de PAs multiples sont en effet des processus ponctuels multivariés dont on peut construire des modèles pertinents. En particulier, on peut représenter un processus pontuel univarié ou multivarié par une fonction d'intensité conditionnelle. On peut donc chercher à en construire des formes paramétriques que l'on ajuste sur les trains de PAs enregistrés à l'aide de méthodes d'estimation de la vraisemblance. La difficulté est ici de définir des modèles de processus ponctuels multivariés qui représentent précisément les activités conjointes des ensembles enregistrés. Ces modèles doivent, en particulier, permettre l'introduction d'interactions entre les processus ponctuels. La seconde difficulté est de construire des algorithmes qui permettent un ajustement efficace de ces modèles sur les données. Ce type de développement a débuté récemment en neurophysiologie (Barbieri et al., 2001 ; Truccolo et al., 2005) et mérite d'être massivement poursuivi. Les travaux effectués en séismologie, où les modèles de processus ponctuels interagissant sont abondamment utilisés (Ogata, 1988), seront d'une aide précieuse à cet égard.

Cette approche est intéressante pour une deuxième raison : elle doit également permettre d'étendre l'utilisation de la *théorie de l'information* à l'analyse de trains de PAs multivariés. La théorie de l'information est actuellement utilisée pour quantifier la variabilité d'un train de PAs, ainsi que l'information mutuelle qui mesure l'association entre deux trains de PAs ou entre un train de PA et un stimulus (Borst et Theunissen, 1999). Dans ce cadre d'analyse, l'expérimentateur quantifie l'information qu'un unique train de PAs transmet à propos d'un stimulus donné. Parce qu'elle prend en compte les temps précis de tous les PAs, la théorie de l'information peut révéler quelle(s) échelle(s) de temps contien(nen)t de l'information dans le codage neuronal. Si l'on est capable de modéliser explicitement la densité de probabilité jointe de l'activité neuronale d'ensemble et du stimulus, on peut estimer l'information mutuelle et tout autre fonction de cette densité de probabilité. L'utilisation de la théorie de l'information est alors possible dans l'analyse de trains multiples et pas uniquement dans celle de trains individuels.

6.5 Conclusion

Les enregistrements simultanés de multiples trains de PAs issus de différents neurones doivent permettre de mieux comprendre comment les neurones agissent de concert pour assurer diverses fonctions cérébrales. Ils doivent également permettre d'établir un lien entre nos connaissances des cellules individuelles (propriétés membranaires et synaptiques) et leur comportement en réseau. Cependant, l'exploitation satisfaisante de ces enregistrements se heurte à deux difficultés majeures : la reconstruction des trains de PAs à partir des données multiunitaires recueillies (*spike-sorting*) d'une part, l'analyse des multiples trains de PAs reconstruits d'autre part. Ce travail de thèse est une contribution au développement d'une approche radicalement nouvelle du premier problème. Le support expérimental de ce développement est le système des PCs dans les tranches de cervelet de rat. Cette préparation a permis une validation expérimentale

de la méthode élaborée. Par ailleurs, l'étude des décharges des PCs n'a pas révélé de corrélations particulières entre ces cellules en régime spontané dans cette préparation, en dépit de l'inhibition exercée par les interneurones de la couche moléculaire.

Chapitre 7

Appendice

Jan Evangelista Purkyne, Purkinje (1787-1869)

Jan Evangelista Purkyne (Purkinje) est né en Bohème (République Tchèque). C'est à Prague qu'il effectue ses études de médecine et sa thèse intitulée "Beobachtungen und Versuche zur Physiologie der Sinne" ("Observations and Experiments Investigating the Physiology of Senses"). Son travail doctoral porte en particulier sur la physiologie de la vision. Il devient professeur de physiologie à l'université de Prague, avant de créer, en 1839, le premier département de physiologie du monde, à l'université de Breslau (Prusse, aujourd'hui Wroclaw en Pologne). Il y crée également le premier institut de recherche en physiologie, en 1842. Il découvre les cellules qui portent son nom en 1837, ainsi que ses fameuses fibres (situées dans les ventricules du coeur) en 1839. Il est le premier à utiliser un microtome pour couper des tranches de tissu. Il est l'un des pionniers de la physiologie expérimentale.

Résumé Pour être réellement exploitables, les données d'enregistrements extracellulaires multiunitaires doivent faire l'objet d'un traitement préalable visant à isoler les activités neuronales individuelles qui les constituent : le *spike-sorting*. Ce travail de thèse est une contribution au développement et à la réalisation d'une méthode automatique de *spike-sorting* implémentant un algorithme de *Monte Carlo par Chaînes de Markov* (MCMC). La méthode proposée permet de tenir compte, en plus de la forme des potentiels d'action (PAs), de l'information fournie par leurs temps d'émission pour réaliser la classification. Cette utilisation de l'information temporelle rend possible l'identification automatique de neurones émettant des PAs de formes non stationnaires. Elle améliore aussi grandement la séparation de neurones aux PAs de formes similaires. Ce travail méthodologique à débouché sur la création d'un logiciel libre accompagné de son manuel d'utilisateur.

Cette méthode de *spike-sorting* a fait l'objet d'une validation expérimentale sur des populations de cellules de Purkinje (PCs), dans les tranches de cervelet de rat. Par ailleurs, l'étude des trains de PAs de ces cellules fournis par le *spike-sorting*, n'a pas révélé de corrélations temporelles significatives en régime spontané, en dépit de l'existence d'une inhibition commune par les interneurones de la couche moléculaire et d'une inhibition directe de PC à PC. Des simulations ont montré que l'influence de ces inhibitions sur les relations temporelles entre les trains de PCs était trop faible pour pouvoir être détectée par nos méthodes d'analyse de corrélations. Les codes élaborés pour l'analyse des trains de PAs sont également disponibles sous la forme d'un second logiciel libre.

Mots-clés MCMC, enregistrements multiples, électrodes multisites, populations neuronales, cervelet, électrophysiologie.

English title A Markov Chain Monte Carlo Approach for Spike-Sorting. Application to the Study of Temporal Correlations in Purkinje Cells' Activities.

Summary To be fully exploitable extracellular multi-unit data have to be sorted out into several single neuron spike trains : this particular data processing is called "spike-sorting". This work is a contribution to the development and the carrying out of an automatic spike-sorting method implementing a *Markov Chain Monte Carlo* (MCMC) method. The proposed method enables the experimentalist to take into account the occurrence times of spikes, in addition to the information provided by their waveforms, to perform spike-sorting. This use of temporal information makes it possible to automatically identify neurons with non-stationary spike waveforms. It also improves the separation of neurons whose spike waveforms are similar. This methodological work led to the release of a free software documented by its user guide.

This spike-sorting method has been experimentally validated on populations of Purkinje cells (PCs) in rat cerebellar slices. Besides, the spike train analysis of these multiple cells data did not reveal any significant temporal correlations between spontaneous PC spike trains, in spite of common inhibition of PCs by molecular layer interneurons and direct inhibition PC to PC. Simulations showed that the influence of these inhibitions onto temporal relations between spike trains is too weak to be detected with our correlation analysis. Codes written to analyse spike trains are also released as a second software.

Keywords MCMC, multi-unit recordings, multi-electrode, neuronal populations, cerebellum, electrophysiology.

Bibliographie

[1] Abeles M, Gerstein GL (1988) Detecting Spatiotemporal Firing Patterns Among Simultaneously Recorded Single Neurons. J Neurophysiol. 60 : 909-924.

[2] Abeles M, Bergman H, Gat I, Meilijson I, Seidemann E, Tishby N, Vaadia E (1995) Cortical activity flips among quasi-stationary states. PNAS. 92 : 8616-8620.

[3] Abeles M, Gat I (2001) Detecting precise firing sequences in experimental data. J Neurosci Methods. 107 : 141-154.

[4] Adrian ED (1935) Discharge frequencies in the cerebral and cerebellar cortex. J Physiol. (Lond) 83 : 33P-33P.

[5] Aertsen AMH, Gerstein GL (1985) Evaluation of Neuronal Connectivity : Sensitivity of Cross-Correlation. Brain Research. 340 : 341-354.

[6] Aertsen AMH, Gerstein GL, Habib MK, Palm G (1989) Dynamics of Neuronal Firing Correlation Modulation of "Effective Connectivity". J Neurophysiol. 61 : 900-917.

[7] Akaike H (1974) A New Look at the Statistical Identification Model. IEEE Transactions on Automatic Control. 19 : 716-723.

[8] Armstrong DM, Rawson JA (1979) Activity patterns of cerebellar cortical neurones and climbing fibre afferents in the awake cat. J Physiol. 289 : 425-448.

[9] Atiya AF (1992) Recognition of multiunit neural signals. IEEE Trans Biomed Eng. 39 : 723-729.

[10] Baker SN, Philbin N, Spinks R, Pinches EM, Wolpert DM, MacManus DG, Pauluis Q, Lemon RN (1999) Multiple single unit recording in the cortex of monkeys using independently moveable microelectrodes. J Neurosci Methods. 94 :5-17.

[11] Barbieri R, Quirk MC, Frank LM, Wilson MA, Brown EN (2001) Construciton and analysis of non-Poisson stimulus-respone models of neural spiking activity. J Neurosci Methods. 105 : 25-37.

[12] Baum LE (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of a Markov chain. Ann Math Stat. 41 : 164-171.

[13] Bezdek JC (1981) *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum, New-York.

[14] Bell CC, Grimm RJ (1969) Discharge Properties of Purkinje Cells Recorded on Single and Double Microelectrodes. J Neurophysiol. 32 : 1044-1055.

[15] Bell CC, Kawasaki T (1972) Relations Among Climbing Fiber Responses of Nearby Purkinje Cells. J Neurohysiol. 35 : 155-169.

[16] Bishop GA (1993) An analysis of HRP-filled basket cell axons in the cat cerebellum. I. Morphometry and configuration. Anatomy and Embryology. 188 : 287-297.

[17] Bishop CM (1995) *Neural Networks for Pattern Recognition*. Oxford University Press.

[18] Borst A, Theunissen F (1999) Information theory and neural coding. Nat Neurosci. 2 : 947-957.

[19] Bower JM, Woolston DC (1983) Congruence of spatial organization of tactile projections to granule cell and Purkinje cell layers of cerebellar hemispheres of the albino rat - vertical organization of cerebellar cortex. J Neurophysiol. 49 : 745 :766.

[20] Brillinger DR (1976) Estimation of second-order intensities of a bivariate stationary point process. J Roy Statist Soc B38 : 60-66.

[21] Brown EN, Frank LM, Tang D, Quirk MC, Wilson MA (1998) A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. J Neurosci. 18 : 7411-7425.

[22] Brown EN, Kass RE, Mitra PP (2004) Multiple neural train data analysis : state-of-the-art and future challenges. Nature Neuroscience. 7 : 456-461.

[23] Buzsaki G (2004) Large-scale recording of neuronal ensembles. Nature Neuroscience. 7 : 446-451.

[24] Camproux AC, Saunier F, Chouvet G, Thalabard JC, Thomas G (1996) A Hidden Markov Model Approach to Neuron Firing Patterns. Biophys J. 71 : 2404-2412.

[25] Cerminara NL, Rawson JA (2004) Evidence that Climbing Fibers Controls an Intrinsic Spike Generator in Cerebellar Purkinje Cells. J Neurosci. 24 : 4510-4517.

[26] Chandra R, Optican L (1997) Detection, Classification and Superposition Resolution of Action Potentials in Multiunit Single-Channel Recordings by On-Line Real-Time Neural Network. IEEE Trans Biomed Eng. 44 : 403-412.

[27] Chapin JK (2004) Using multi-neuron population recordings for neural prosthetics. Nature Neuroscience. 7 : 452-455.

[28] Cheron G, Gall D, Servais L, Dan B, Maex R , Schiffmann SN (2004) Inactivation of Calcium-Binding Protein Genes Induces 160 Hz Oscillations in the Cerebellar Cortex of Alert Mice. J Neurosci. 24 : 434-441.

[29] Cheron G, Servais L, Wagstaff J, Dan B (2005) Fast cerebellar oscillation associated with ataxia in a mouse model of angelman syndrome. Neuroscience. 130 :631-637.

[30] Cingolani LA, Gymnopoulos M, Boccaccio A, Stocker M, Pedarzani P (2002) Developmental regulation of small-conductance Ca^{2+}-activated K^+ channel expression and function in rat Purkinje neurons. J Neurosci. 22 : 4456-4467.

[31] Connors BW, Long MA (2004) Electrical Synapses in the Mammalian Brain. Annu Rev Neurosci. 27 : 393-418.

[32] Crépel F (1972). Maturation of the Cerebellar Purkinje Cells. I. Postnatal Evolution of the Purkinje Cell Spontaneous Firing in the Rat. Exp. Brain Res. 14 : 463-472.

[33] Crépel F, Penit-Soria J (1986) Inward rectification and low-threshold calcium conductance in rat cerebellar Purkinje cells. An in vitro study. J Physiol. (Lond) 372 : 1-23.

[34] Csicsvari J, Henze DA, Jamieson B, Harris KD, Sirota A, Bartho P, Wise KD, Buzsaki G (2003) Massively Parallel Recording of Unit and Local Field Potentials With Silicon-Based Electrodes. J Neurophysiol 90 :1314-1323.

[35] Dean I, Robertson SJ, Edwards F (2003) Serotonin drives a Novel GABAergic Synaptic Current Recorded in Rat Cerebellar Purkinje Cells : A Lugaro Cell to Purkinje Cell Synapse. J Neurosci. 23 : 4457-4469.

[36] Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc B. 39 : 1-38.

[37] Dieudonné S, Dumoulin A (2000) Serotonin-Driven Log-Range Inhibitory Connections in the Cerebellar Cortex. J Neurosci. 20 : 1837-1848.

[38] Dinning GJ, Sanderson AC (1983) Real time classification of multiunit neural signals using reduced feature sets. IEEE Trans Bio-Med Eng. 28 : 804-811.

[39] Drake KL, Wise KD, Farraye J, Anderson DJ, and Bement SL (1988) Performance of planar multisite microprobes in recording extracellular single-unit intracortical activity. IEEE Trans Biomed Eng 35 :719-732.

[40] Ebner TJ, Blödel JR (1981) Correlation Between Activity of Purkinje Cells and Its Modification by Natural Peripheral Stimuli. J Neurophysiol. 45 : 948-961.

[41] Eccles JC, Llinas R, Sasaki K (1966) The action of antidromic impulses on the cerebellar Purkinje cells. J Physiol. 182 : 316-345.

[42] Eccles JC, Ito M, Szentagothai J (1967) *The cerebellum as a neuronal machine*. Springer-Verlag, Berlin.

[43] Edgerton R, Reinhart PH (2003) Dsitinc contributions of small and large conductance Ca^{2+}-activated K^{+} channels to rat Purkinje neuron function. J Physiol. 548 : 53-69.

[44] Egert U, Heck D, Aertsen A (2002) Two-dimensional monitoring of spiking networks in acute brain slices. Exp Brain Res. 142 : 268-274.

[45] Eggermont JJ, Epping WJ, Aertsen AM (1983) Stimulus dependent neural correlations in the auditory midbrain of the grassfrog (*Rana Temporaria* L.). Biol Cybern. 47 : 103-117.

[46] Fee MS, Mitra PP, Kleinfeld D (1996) Variability of extracellular spike waveforms of cortical neurons. J Neurophysiol. 76 : 3823-3833.

[47] Furshan EJ, Potter DD (1959) Transmission at the giant motor synapses of the crayfish. J Physiol. 145 : 289-325.

[48] Gähwiler BH, Llano I (1989) Sodium and potassium conductances in somatic membranes of rat Purkinje cells from organotypic cerebellar cultures. J Physiol. 417 : 105-122.

[49] Garcia P, Paz Suarez C, Rodriguez J, Rodriguez M (1998) Unsupervised classification of neural spikes with a hybrid multilayer artificial network. J Neurosci Methods. 82 : 59-73.

[50] Gerstein GL, Perkel DH (1972) Mutual Temporal Relationships among Neuronal Spike Trains. Biophys J. 12 : 453-473.

[51] Gerstein GL, Perkel DH, Dayhoff JE (1985) Cooperative Firing Activity in Simultaneously Recorded Populations of Neurons : Detection and Measurement. J Neurosci. 5 : 881-889.

[52] Glaser EM, Marks WB (1968) Online separation of interleaved neuronal pulse sequences. Data Acquisition Process Biol Med. 5 : 137-156.

[53] Granit R, Philips CG (1956) Excitatory and inhibitory processes acting upon individual Purkinje cells of the cerebellum in cats. J Physiol (Lond) 133 : 520-547.

[54] Gray CM, Maldonado PE, Wilson M, McNaughton B (1995) Tetrode markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. J Neurosci Methods 63 :43-54.

[55] Grenier F, Timofeev I, Steriade M (1998) Leading role of thalamic over cortical neurons during postinhibitory rebound excitation. PNAS. 95 : 13929-13934.

[56] Grieco TM, Malhotra JD, Chen C, Isom LL, Raman IM (2005) Open-Channel Block by the Cytoplasmic Tail of Sodium Channel β_4 as a Mechanism for Resurgent Sodium Current. Neuron. 45 : 233-244.

[57] Gross GW, Rhoades BK, Reust DL, Schwalm FU (1993) Stimulation of monolayer networks in culture through thin-film indium-tin oxide recording electrodes. J Neurosci Methods 50 :131-143.

[58] Gross GW, Harsch A, Rhoades BK, Reust DL, Goepel W (1997) Odor, drug and toxin analysis with neuronal networks in vitro : extracellular array recording of network responses. Biosens Bioelectron 12 :373-393.

[59] Grün S, Diesmann M, Grammont F, Riehle A, Aertsen A (1999) Detecting unitary events without discretization of time. J Neurosci Methods. 93 : 67-79.

[60] Grün S, Diesmann M, Aertsen A (2002) Unitary Events in Multiple Single-Neuron Spiking Activity : I. Detection and Significance.

[61] Gruol DL, Franklin CL (1987) Morphological and physiological differentiation of Purkinje neurons in cultures of rat cerebellum. J. Neurosci. 7 : 1271-1293.

[62] Gruol DL, Jacquin T, Yool AJ (1991) Single-channel K^+ currents recorded from the somatic and dendritic regions of cerebellar Purkinje neurons in culture. J Neurosci. 11 : 1002-1015.

[63] Güçlü B, Bolanowski SJ (2004) Tristate Markov Model for the firing statistics of rapidly-adapting mechanoreceptive fibers. J Comput Neurosci. 17 : 107-126.

[64] Harris KD, Henze DA, Csicsvari J, Hirase H, Buzsaki G (2000) Accuracy of Tetrode Spike Separation as Determined by Simultaneous Intracellular and Extracellular Measurements. J Neuropysiol. 84 : 401-414.

[65] Hartigan JA (1975) *Clustering Algorithms*. Wiley, New-York.

[66] Hartmann MJ, Bower JM (1998) Oscillatory Activity in the Cerebellar Hemispheres of Unrestrained Rats. J Neurophysiol. 80 :1598-1604.

[67] Häusser M, Clark BA (1997) Tonic Synaptic Inhibition Modulates Neuronal Output Pattern and Spatiotemporal Synaptic Integration. Neuron 19 : 665-678.

[68] Hebb DO (1949) *The Organization of Behavior : a Neurophysiological Theory.* Wiley, New-Yorg.

[69] Hounsgaard J (1979). Pacemaker properties of mammalian Purkinje cells. Acta Physiol Scand. 106 : 91-92.

[70] Hulata E, Segev R, Ben-Jacob E (2002) A method for spike sorting and detection based on wavelet packets and Shannon mutual information. J Neurosci Methods. 117 : 1-12.

[71] Isope P, Dieudonne S, Barbour B (2002) Temporal Organisation of Activity in the Cerebellar Cortex : A Manifesto for Synchrony. Ann NY Acad Sci. 978 : 1-11.

[72] Jäger D, Bower JM (1994) Prolonged responses in rat cerebellar Purkinje cells following activation of the granule cell layer : an intracellular in vitro and in vivo study. Exp. Brain Res. 100 : 200-214.

[73] Jäger D, Bower JM (1999) Synaptic control of spiking in cerebellar Purkinje cells : dynamic current clamp based on model conductances. J Neurosci. 17 : 91-106.

[74] Janke W. Statistical Analysis of Simulations : Data Correlations and Error Estimation [online]. 2002 ; http ://www.fz-juelich.de/nic-series/volume10 [29th March 2005].

[75] Jansen RF (1990) The reconstruction of individual spike trains from extracellular multineuron recordings using a neural network emulation program. J Neurosci Methods. 35 : 203-213.

[76] Johnson DH, Kiang NYS (1976) Analysis of discharges recorded simultaneously from pairs of auditory nerve fibers. Biophys J. 16 : 719-734.

[77] Johnson DH (1996) Point Process Models of single-Neuron Discharges. J Comput Neurosci. 3 : 275-299.

[78] Kass RE, Raftery AE (1995) Bayes Factors. J Amer Statist Assoc. 90 : 773-795.

[79] Khaliq ZM, Gouwens NW, Raman IM (2003) The contribution of resurgent sodium current to high-frequency firing in Purkinje neurons : an experimental and modeling study. J. Neurosci. 23 : 4899-4912.

[80] Kim KH, Kim SJ (2000) Neural Spike Sorting Under Nearly 0-dB Signal-to-Noise Ratio Using Nonlinear Energy Operator and Artificial Neural-Network Classifier. IEEE Trans Biomed Eng. 47 : 1406-1411.

[81] King JS, Bishop GA (1982) The synaptic features of horseradish peroxidase-labelled recurrent collaterals in the ganglionic layer of the cat cerebellar cortex. J Neurocytology. 11 : 867-880.

[82] Krüger J (1983) Simultaneous Individual Recordings From Many Cerebral Neurons : Techniques and Results. Rev Physiol Biochem Pharm. 98 : 177-233.

[83] Lang EJ, Sigihara I, Welsh JP, Llinas R (1999) Patterns of Spontaneous Purkinje Cell Comple Spike Activity in the Awake Rat. J Neurosci. 19 : 2728-2739.

[84] Larramendi LMH, Lemkey-Johnston N (1970) The Distribution of Recurrent Purkinje Collateral Synapses in the Mouse Cerebellar Cortex : An Electron Microscopic Study. J Comp Neur. 138 : 451-482.

[85] Latham A, Paul DH (1971) Spontaneous activity of cerebellar Purkinje cells and their responses to impulses in climbing fibers. J. Physiol. (Lond). 213 : 135-156.

[86] Lebarbier E, Mary-Huard T (2004) Le critère BIC : fondements théoriques et interprétation. Projet SELECT. INRIA. Rapport de recherche 5315.

[87] Letelier JC, Weber PP (2000) Spike sorting based on discrete wavelet transform coefficients. J Neurosci Methods. 101 : 93-106.

[88] Lewicki MS (1994) Bayesian modeling and classification of neural signals. Neural Comput. 6 : 1005-1030.

[89] Lewicki MS (1998) A review of methods for spike-sorting : the detection and classification of neural action potentials. Network : Comput Neural Syst 9 : R53-R78.

[90] Ling L, Tohlburst DJ (1983) Recovering the parameters of finite mixtures of normal distributions from noisy record : an empirical comparison of different estimating procedures. J Neurosci Methods. 8 : 309-333.

[91] Llano I, Gerschenfeld HM (1993a) Inhibitory synaptic currents in stellate cells of rat cerebellar slices. J Physiol. 468 : 177-200.

[92] Llano I, Gerschenfeld HM (1993b) β-adrenergic enhancement of inhibitory synaptic activity in rat cerebellar stellate and Purkinje cells. J Physiol. 468 : 201-224.

[93] Llinas R, Nicholson C, Precht W (1969) Preferred centripetal conduction of dendritic spikes in alligator Purkinje cells. Science. 163 : 184-187.

[94] Llinas R, Sugimori M. (1980a). Electrophysiological properties of in vitro Purkinje cell somata in mammalian cerebellar slices. J. Physiol. (Lond) 305 : 171-195.

[95] Llinas R, Sugimori M (1980b). Electrophysiological properties of in vitro Purkinje cell dendrites in mammalian cerebellar slices. J. Physiol. (Lond) 305 : 197-213.

[96] Llinas R, Sugimori M (1992) The electrophysiology of the cerebellar Purkinje cell revisited. In : *The cerebellum revisited*. Edité par R. Llinas et C. Sotelo. Springer-Verlag, New-York.

[97] Llinas R, Yarom Y (1986) Oscillatory properties of ginea-pig inferior olivary neurones and their pharmacological modulation : an in vitro study. J pHysiol. (Lond) 376 : 163-182.

[98] Lloyd SP (1982) Least squares quantization in PCM. IEEE Transations on Information Theory. 28 : 129-137.

[99] Loewenstein Y, Mahon S, Chadderton P, Kitamura K, Sompolinsky H, Yarom Y, Haeusser M (2005) Bistability of cerebellar Purkinje cells modulated by sensory stimulation. Nature Neurosci 8 :202-211.

[100] Lu H, Hartmann MJ, Bower JM (2005) Correlations between Purkinje Cell Single Unit Activity and Simultaneously Recorded Field Potentials in the Immediately Underlying Granule Cell Layer. J Neurophysiol. In press.

[101] Maex R, De Schutter E (1998) Synchronization of Golgi and Granule Cell Firing in a Detailed Network Model of the Cerebellar Granule Cell Layer. J Neurophysiol. 80 : 2521-2537.

[102] Mann-Metzer P, Yarom Y (1999) Electrotonic Coupling Interacts with Intrisic Properties to Generate Synchronized Activity in Cerebellar Networks of Inhibitory Interneurons. J Neurosci. 19 : 3298-3306.

[103] Martina M, Lan Yao G, Bean B (2003) Properties and Functional Role of Voltage-Dependent Potassium Channels in Dendrites of Rat Cerebellar Purkinje Neuorns. J Neurosci. 23 : 5698-5707.

[104] McNaughton BL, O'Keefe J, Barnes CA (1983) The stereotrode : a new technique for simultaneous isolation of several single units in the central nervous system from multiple unit records. J Neurosci Methods. 8 : 391-397.

[105] Meister M, Pine J, Baylor DA (1994) Multi-neuronal signals from the retina : acquisition and analysis. J Neurosci Methods. 51 : 95-106.

[106] Midtgaard J (1992) Stellate cell inhibition of Purkinje cells in the turtle cerebellum *in vitro*. J Physiol. 457 : 355-367.

[107] Nam SC, Hockberger PE (1997) Analysis of Spontaneous Electrical Activity in Cerebellar Purkinje Cells Acutely Isolated from Postnatal Rats. J. Neurobiol. 33 : 18-32.

[108] Nicolelis MAL, Ghazanfar AA, Faggin BM, Votaw S, Oliveira LM (1997) Reconstructing the Engram : Simultaneous, Multisite, Many Single Neuron Recordings. Neuron 18 :529-537.

[109] Ogata Y (1988) Statistical Models for Earthquake Occurrences and Residual Analysis for Point Processes. Journal of the American Statistical association. 83 : 9-27.

[110] Ogata Y (1989) A Monte Carlo Method for High Dimensional Integration. Numerische Mathematik. 55 : 137-157.

[111] Oghalai JS, Street WN, Rhode WS (1994) A neural network-based spike discriminator. J Neurosci Methods. 54 : 9-22.

[112] Oka H, Shimono K, Ogawa R, Sugihara H, Taketani M (1999) A new planar multielectrode array for extracellular recording : application to hippocampal acute slice. J Neurosci Methods. 93 : 61-67.

[113] Palay SL, Chan-Palay V (1974) *Cerebellar Cortex, Cytology and Organization.* Springer, Berlin.

[114] Pellerin JP, Lamarre Y (1997) Local Field Potential Oscillations in Primate Cerebellar Cortex During Voluntary Movement. J Neurophysiol. 78 : 3502-3507.

[115] Perkel DH, Gerstein GL, Moore GP (1967) Neuronal Spike Trains and Stochastic Point Processes. II Simultaneous Spike Trains. Biophys J. 7 : 419-440.

[116] Pfeuty B, Mato G, Golomb D, Hansel D (2003) Electrical Synapses and Synchrony : The Role of Intrinsic Currents. J Neurosci. 23 : 6280-6294.

[117] Pfeuty B, Golomb D, Mato G, Hansel D (2005) The Combined Effects of Inhibitory and Electrical Synapses in Synchrony. Neural Computation. 17 : 633-670.

[118] Pouzat C, Hestrin S (1997) Developmental Regulation of Basket/Stellate Cell -> Purkinje Cell Synapses in the Cerebellum. J Neurosci. 17 : 9104-9112.

[119] Pouzat C (1998) Etude des synapses inhibitrices formées par les interneurones de la couche moléculaire du cervelet en développement. Thèse de doctorat (université Paris XI Orsay).

[120] Pouzat C, Mazor O, Laurent G (2002) Using noise signature to optimize spike-sorting and to asses neuronal classification quality. J Neurosci Meth 122 :43-57.

[121] Raman IM, Bean BP (1997) Resurgent Sodium Current and Action Potential Formation in Dissociated Cerebellar Purkinje Neurons. J. Neurosci. 17 : 4517-4526.

[122] Raman IM, Sprunger LK, Meisler MH, Bean B (1997) Altered Subthreshold Sodium Currents and Disrupted Firing Patterns in Purkinje Neurons of *Scn8a* Mutant Mice. Neuron. 19 : 881-891.

[123] Raman IM, Bean BP (1999). Ionic Currents Underlying Sponaneous Action Potentials in Isolated Cerebellar Purkinje Neurons. J. Neurosci. 19 : 1663-1674.

[124] Ramon y Cajal S (1911) *Histologie du système nerveux de l'Homme et des Vertébrés*, Vol. II. Paris, Maloine. Réimpression : Madrid, Consejo Superior de Investigaciones Cientificas, 1952.

[125] Robert CP, Casella G (1999) *Monte Carlo Statistical Methods*. Springer-Verlag, New-York.

[126] Quian Quiroga R, Nadasdy Z, Ben-Shaul Y (2004) Unsupervised Spike Detection and Sorting with Wavelets and Superparamagnetic Clustering. Neural Computation. 16 : 1661-1687.

[127] Salganicoff M, Sarna M, Sax L, Gerstein GL (1988) Unsupervised waveform classification for multi-neural recordings : a real-time, software based system. I. Algorithms and implementation. J Neurosci Methods. 25 : 181-187.

[128] Santamaria F, Bower JM (2005) Background Synaptic Activity Modulates the Response of a Modeled Purkinje Cell to paired Afferent Input. J Neurophysiol. 93 : 237-250.

[129] Sato Y, Miura A, Fushiki H, Kawasaki T (1992) Short-term modulation of cerebellar Purkinje cell activity after sponatneous climbing fiber input. J Neurophysiol. 68 : 2051-2062.

[130] Schmidt EM (1984a) Instruments for sorting neuroelectric data : a review. J Neurosci Methods. 12 : 1-24.

[131] Schmidt EM (1984b) Computer separation of multi-unit neuroelectric data : a review. J Neurosci Methods. 12 : 95-111.

[132] Schwarz G (1978) Estimating the Dimension of a Model. Annals of Statistics. 6 : 461-464.

[133] Seidemann E, Meilijson I, Abeles M, Bergman H, Vaadia E (1996) Simultaneous Recorded Single Units in the Frontal Cortex Go through Sequences of Discrete and Stable States in Monkeys Performing a Delayed Localization Task. J Neurosci. 16 : 752-768.

[134] Serruya MD, Hatsopoulos NG, Paninski L, Fellows MR, Donoghue JP (2002) Instant neural control of a movement signal. Nature. 416 : 141-142.

[135] Shoham S, Fellows MR, Normann RA (2003) Robust, automatic spike sorting using mixtures of multivariate t-distributions. J Neurosci Methods. 127 : 111-122.

[136] Sloper JJ (1972) Gap junctions between dendrties in the primate neocortex. Brain Res. 44 : 641-646.

[137] Sokal AD. Monte Carlo in Statistical Mechanics : Foundations and New Algorithms. Cours de Troisième Cycle de la Physique en Suisse Romande [online]. http ://citeseer.nj.nec.com/sokal96monte.html [29th March 2005]. 1989.

[138] Sotelo C, Llinas R (1972) Specialised membrane junctions between neurones in the vertebrate cerebellar cortex. J Cell Biol. 53 : 271-289.

[139] Swensen AM, Bean BP (2003) Ionic Mechanisms of Burst Firing in Dissociated Purkinje Neurons. J. Neurosci. 23 : 9650-9663.

[140] Swensen AM, Bean BP (2005) Robustness of Burst Firing in Dissociated Purkinje Neurons with Acute or Long-Term Reductions in Sodium Conductance. J Neurosci. 25 : 3509-3520.

[141] Truccolo W, Eden UT, Fellows MR, Donoghue JP, Brown EN (2005) A Point Process Framework for Relating Neural Spiking Activity to Spiking History, Neural ensemble, and extrinsic Covariate Effects. J Neurophysiol. 93 : 1074-1089.

[142] Usowicz MM, Sugimori M, Cherksey B, Llinas, R (1992) P-type calcium channels in the somata and dendrites oof adult cerebellar Purkinje cells. Neuron. 9 : 1185-1199.

[143] Vincent P, Marty A (1993) Neighboring Cerebellar Purkinje Cells Communicate via Retrograde Inhibition of Common Presynaptic Interneurons. Neuron 11 : 885-893.

[144] Vincent P, Marty A (1996) Fluctuations of inhibitory postsynaptic currents in Purkinje cells from rat cerebellar slices. J Physiol. (Lond) 494 : 183-199.

[145] Vos B, Wijnants M, Taeymans S, De Schutter E (1999) Miniature carrier with six independently moveable electrodes for recording of multiple single-units in the cerebellar cortex of awake rats. J Neurosci Methods. 94 : 19-26.

[146] Warland DK, Reinagel P, Meister M (1997) Decoding visual information from a population of retinal ganglion cells. J Neurophysiol. 78 : 2336-2350.

[147] Wessberg J, Stambaugh C, Kralik JD, Beck PD, Laubach M, Chapin JK, Kim J, Biggs SJ, Srinivasan MA, Nicolelis AL (2000) Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. Nature. 408 : 361-365.

[148] Wilson MA, McNaughton BL (1993) Dynamics of the hippocampal ensemble code for space. Science. 261 :1055-1058.

[149] Womack M, Khodakhah K (2002) Active contribution of dendrites to the tonic and trimodal patterns of activity in cerebellar Purkinje neurons. J. Neurosci. 22 : 10603-10612.

[150] Womack M, Khodakhah K (2004) Dendritic control of spontaneous bursting in cerebellar Purkinje cells. J. Neurosci. 24 : 3511-352.

[151] Yvert B, Branchereau P, Meyrand P (2004) Multiple Spontaneous Rhythmic Activity Patterns Generated by the Embryonic Mouse Spinal Cord Occur Within a Specific Developmental Time Window. J Neurophysiol. 91 : 2101-2109.