Un modèle informatique biologiquement réaliste des oscillations neuronales pathologiques observées dans la maladie de Parkinson

Nathalie Azevedo Carvalho

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Un modèle informatique biologiquement réaliste des oscillations neuronales pathologiques observées dans la maladie de Parkinson

THÈSE

présentée et soutenue publiquement le 4 mai 2022

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

AZEVEDO CARVALHO Nathalie

Composition du jury

Rapporteurs :	Benoît GIRARD Nicolas ROUGIER
Examinateurs :	Jérôme BAUFRETON (Président du jury) Pierre YGER
Encadrants :	Dominique MARTINEZ Laure BUHRY Sylvain CONTASSOT-VIVIER

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503
Inria Nancy - Grand Est

Mis en page avec la classe thesul.

Remerciements

Il y a quelques années encore, je n'aurais jamais pu imaginer arriver jusqu'ici. C'est-à-dire à la rédaction de cette thèse, et plus précisément à la rédaction de ces remerciements. Au départ faire une thèse, ce n'était pas forcément une évidence pour moi. En effet, ce n'est que pendant mon stage de fin d'études effectué au LORIA que j'ai commencé à développer un réel intérêt pour la recherche et l'envie d'aller plus loin. Le parcours n'a pas été facile, mais avec beaucoup de travail et d'acharnement j'y suis arrivé. Le soutien et les bons conseils des personnes qui m'ont entouré tout au long de cette thèse y ont beaucoup contribué, merci à eux ! Pendant cette thèse, j'ai vécu une expérience incroyable et surtout fait de bonnes rencontres tant au niveau professionnel, qu'au niveau personnel !

Tout d'abord, je tiens à remercier les membres du jury pour avoir accepté de participer à cette thèse mais aussi pour leur travail de relecture et leurs remarques pertinentes. Ces remarques m'ont permis d'améliorer considérablement la qualité de ce manuscrit mais aussi de la présentation de mon travail de thèse. Je tiens aussi à vous remercier chaleureusement d'avoir assisté en présentielle à cette soutenance de thèse malgré le contexte sanitaire, lié à la pandémie de Covid19, encore fragile à ce moment-là. Merci aussi à Jérôme de m'avoir accueil dans son équipe à Bordeaux. J'ai pu à cette occasion faire la connaissance d'une équipe très accueillante, mais aussi d'un nouveau monde pour moi, l'expérimentale. Cette collaboration m'a aussi permis d'améliorer, et de mieux comprendre, mes résultats pour cette thèse.

Merci aux membres de l'équipe NeuroSys, maintenant NeuroRhythms, pour leur accueil ! J'ai passé de bons moments avec chacun des doctarants et doctorantes, merci ! Merci aussi aux permanents, Laurent et Patrick, toujours à l'écoute des besoins de l'équipe. Vous faites toujours de votre mieux pour que tout se passe bien. Merci à Sébastien qui a toujours su être à l'écoute et est souvent de bons conseils lorsque j'en avais besoin. Cette thèse aurait été bien plus difficile moralement sans nos nombreuses discussions, merci !

Un grand merci à mes encadrants ! Jongler entre trois encadrants, ce n'est pas toujours facile néanmoins je sais que je peux compter sur vous ! Merci à vous d'avoir été disponible tout au long de ma thèse d'avoir été à mon écoute, mais surtout de m'avoir permis de découvrir le monde de la recherche, avec ces bons comme ces mauvais côtés. Merci à Laure et Dominique de m'avoir fait confiance pour cette thèse et pour m'avoir confié un projet complexe par sa pluridisciplinarité. Vous m'avez permis de travailler sur un sujet qui m'a toujours intrigué, et fasciné, les neurosciences computationnelles mais surtout la maladie de Parkinson. Merci à Sylvain pour son aide illimité, sa disponibilité et sa patience pour me faire découvrir le développement logiciel. Merci encore une fois à vous trois !

Merci à mes amies ! Sixtine et Juliette, merci pour nos nombreuses visios pendant la thèse. Et surtout merci à toi, Anna, qui tout au long de mes études, a toujours été là pour moi et à l'écoute dans les bons moments, mais aussi dans les moins bons moments.

Ce Merci vient du fond du cœur, car c'est grâce à eux que j'ai fait cette thèse, ma Famille. Ils m'ont toujours soutenu depuis le début et m'ont donné la force de faire un si long chemin vers cette thèse. Je voudrais particulièrement remercier, Filipa, tu es toujours là pour moi et c'est en

partie grâce à toi que j'ai pris la décision de faire cette thèse. Et comment ne pas remercier mes parents et mon frère, ils ont été formidables et patients, à l'écoute et toujours prêt à m'aider lorsque j'en avais besoin.

Et enfin, la personne sans laquelle, je serais devenu dingue à de nombreuses reprises pendant cette thèse. Merci Daniel, mon plus grand soutien et confident. Je ne suis pas sûr que j'aurais pu finir cette thèse saine d'esprit sans ton aide et tes conseils. Cette période de pandémie et de nombreuses autres situations, qui ont pris une grande place durant ma thèse, n'ont vraiment pas été facile à surmonter et je te remercie d'avoir été là pour moi ! Mais surtout, merci d'avoir fêté avec moi mes petites victoires, mais surtout mes grandes victoires !

Je vais y arriver !
Je vais réussir !

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Acronymes

A^-	molécule chargée négativement
Ca^{2+}	ion calcium
Ca_{vH}	canaux calciques à haut seuil d'activation, en anglais ***H**igh-**v**oltage-activated **calcium** channels*
Cl^-	ion chlore
HCN	canaux activés par l'hyperpolarisation, modulés par les nucléotides cyclides et non sélectifs aux cations, en anglais ***H**yperpolarisation-actived **C**yclic-**N**ucleotid modulated cation non-selective channels*
K^+	ion potassium
NaF	canaux transitoires rapides de sodium, en anglais ***f**ast transient sodium channels*
NaP	canaux sodiques "**p**ersistants", en anglais *persistent sodium channels*
Na^+	ion sodium
SK	canaux potassiques à faible conductance activés par le calcium, en anglais ***S**mall conductance calcium-activated **potassium** channels*
1T	mono**T**hread
AHP	hyperpolarisation, en anglais ***A**fter**H**yper**P**olarization*
C	langage de programmation **C**
C++	langage de programmation **C++**
cat	cation
CUDA	***C**ompute **U**nified **D**evice **A**rchitecture*
Cython	langage de programmation et un compilateur qui simplifient l'écriture d'extensions compilées pour Python
DBS	*stimulation cérébrale profonde*, en anglais ***D**eep **B**rain **S**timulation*,
DE	évolution différentielle, en anglais ***D**ifferential **E**volution*

EDO	**É**quation **d**ifférentielle **O**rdinaire
FI	entre le taux de décharge et le courant injecté, en anglais ***F**iring rate and **I**njected current*
FSN	**F**ast-**S**piking **N**eurons
GB	**G**anglions de la **B**ase
GeNN	**G**PU **e**nhanced **N**euronal **N**etwork
GP	**G**lobus **P**allidus
GPe	**GP e**xterne
GPe-TA	**GPe** de **T**ype **A** (ou GPeA)
GPe-TI	**GPe** de **T**ype **I** (ou GPeP)
GPeA	**GPe A**rkypallidaux
GPeP	**GPe P**rototypiques
GPi	**GP i**nterne
GPU	*processeur graphique*, en anglais ***G**raphics **P**rocessing **U**nit*,
HH	**H**odgkin-**H**uxley
IF	*Intègre-et-tire*, en anglais ***I**ntegrate-and-**F**ire*,
LFP	*potentiel de champ local*, en anglais ***L**ocal **F**ield **P**otentiel*,
LIF	*Intègre-et-tire à fuite*, en anglais ***L**eaky **I**ntegrate-and-**F**ire*,
MJE-G	**M**ise à **J**our à chaque **É**vénement avec **G**énération de connectivité
MJE-S	**M**ise à **J**our à chaque **É**vénement avec **S**tockage de connectivité
MM	**M**ulti**M**achines
MP	**M**aladie de **P**arkinson
MPI	*passage de messages entre ordinateurs distants*, en anglais ***M**essage **P**assing **I**nterface*,
MSN	**M**edium **S**piny **N**eurons
MSN-D1	**MSN** à récepteur du type **D1**
MSN-D2	**MSN** à récepteur du type **D2**
MT	**M**ulti**T**hreading
MTMM	**M**ulti**T**hreading et **M**ulti**M**achines
NEST	**NE**ural **S**imulation **T**ool
OpenMP	*interface de programmation pour le calcul parallèle*, en anglais ***Open M**ulti-**P**rocessing*,

PA	**P**otentiel d'**A**ction
PRNG	*générateur de nombres pseudo-aléatoire*, en anglais ***P**seudo**R**andom **N**umber **G**enerator*,
PT-G	à **P**as de **T**emps avec **G**énération de connectivité
PT-S	à **P**as de **T**emps avec **S**tockage de connectivité
RF	*Résonne-et-tire*, en anglais ***R**esonate-and-**F**ire*,
RK2	**R**unge-**K**utta **2**
SiReNe	***Si**mulateur de **Ré**seaux de **Ne**urones*
SM	**S**ymptôme **M**oteur
SN	**S**ubstance **N**oire
SNM	**S**ymptôme **N**on **M**oteur
STN	*noyau sous-thalamique*, en anglais ***S**ub**T**halamic **N**ucleus*,
STR	**STR**iatum

Introduction générale

Cette thèse portant sur la modélisation et la simulation de réseaux neuronaux biologiques, il me paraît important de commencer par définir les éléments fondamentaux permettant de comprendre le contexte de ce travail et les objectifs fixés.

Neurosciences computationnelles et système nerveux central

Les neurosciences consiste à étudier le système nerveux central (i.e. le cerveau et la moelle épinière) de l'échelle moléculaire à l'organe. Plus précisément, elles regroupent des techniques d'analyse du fonctionnement et de la structure du système nerveux. Il existe plusieurs champs de recherche, comme les neurosciences moléculaires, cellulaires, la neurophysiologie ou les neurosciences cognitives, mais plus récemment des nouvelles disciplines sont apparues comme par exemple les neurosciences computationnelles. Cette thèse s'inscrit dans le cadre de cette dernière.

Les neurosciences computationnelles sont vue ici comme le développement de méthodes de calcul mathématiques afin de mieux comprendre les relations complexes entre les structures et le fonctionnement du système nerveux central. Les modèles mathématiques, les statistiques et les simulations numériques permettent d'étudier ces relations et le fonctionnement du système nerveux central.

Les deux hémisphères (droite et gauche) sont divisées en plusieurs lobes ; le lobe frontal, pariétal, occipital, temporal, qui sont des lobes externes, et le lobe limbique et insula, qui sont des lobes internes. De plus, nous avons le cervelet et le tronc cérébral faisant partie des structures nerveuses distincts du cerveau. Le lobe frontal est notamment impliqué dans la pensée, la planification, la rationalité et le jugement. Le lobe pariétal traite l'information sensorielle, gère le sens de l'espace et de la navigation. Le lobe occipital est le siège du traitement de l'information visuelle. Le lobe temporal est le centre de l'audition et est impliqué dans le langage. Le lobe limbique traite les informations concernant les émotions, les affects et la mémoire. Pour finir, le lobe de l'insula traite les informations liées à la douleur, l'odeur et le goût.

Un mouvement se réalise grâce à un ensemble d'étapes. La première étape est l'initiation du mouvement par le lobe frontal et préfrontal. La deuxième étape est la planification du mouvement, à ce moment le cerveau programme le mouvement dans l'espace et la vitesse à laquelle faire le mouvement. La troisième, et dernière étape est l'exécution du mouvement réalisé par le cortex moteur primaire se trouvant au niveau du lobe frontal. L'information est ensuite transmise au tronc cérébral, la moelle épinière puis aux muscles.

Les ganglions de la base, situés à la base du cerveau antérieur et au sommet du cerveau moyen, sont responsables de l'initiation du mouvement et ont le rôle de supprimer les mouvements indésirables, d'enclencher et de maintenir le mouvement. Le cervelet planifie le mouvement et a pour rôle de faire la différence entre un mouvement voulu et le mouvement réellement effectué par l'homme.

Ganglions de la base

Les ganglions de la base sont responsables de l'activité motrice, de l'apprentissage moteur, mais aussi de la motivation et de l'émotion. Comme son nom l'indique, les ganglions de la base sont composés de plusieurs ganglions, i.e. le striatum, le noyau sous-thalamique, le globus pallidus externe et interne et pour finir la substance noire, que nous n'allons pas trop aborder dans ce manuscrit. Dans le globus pallidus externe, nous distinguons deux différentes populations de neurones, les arkypallidaux et les prototypiques. De plus, le striatum se caractérise par des neurones ayant des récepteurs dopaminergiques du type D1 et du type D2.

Dans les ganglions de la base, nous distinguons deux voies principales, la voie directe et la voie indirecte, introduites par Albin et al.[8]. La voie indirecte, impliquant les neurones dopaminergiques D2, modère l'activation de la voie principale, i.e. l'inhibition du globus pallidus externe et du noyau sous-thalamique par le striatum. La voie directe, impliquant les neurones dopaminergiques D1, passe par les neurones de la substance noire et du globus pallidus interne. Dans le striatum, les deux voies, directe et indirecte, exercent un contrôle opposé sur l'exécution des mouvements. La voie directe favorise le mouvement, alors que la voie indirecte exerce un frein sur l'activité motrice [68].

Le dysfonctionnement des ganglions de la base est lié à des maladies neurodégénératives, comme la maladie de Parkinson.

Maladie de Parkinson

La maladie de Parkinson se caractérise par des symptômes moteurs, mais aussi des symptômes non moteurs qui ne sont pas souvent évoqués [32]. Les symptômes moteurs se manifestent par une bradykinésie (ou hypokinésie), i.e. les mouvements du patient sont considérablement réduits et ralentis, des difficultés à déclencher des mouvements volontaires, une rigidité au niveau des muscles, dite plastique (ou hypertonie extrapyramide), des difficultés à parler ou à respirer [57]. Les symptômes non moteurs, récemment reconnus comme des symptômes de la maladie de Parkinson, sont, entre autres, la dépression, les troubles du sommeil paradoxal ou encore les anomalies sensorielles [61] qui peuvent apparaître plusieurs années avant la symptomatologie motrice. Les causes des symptômes non moteurs pourraient être liées à des pathologies en dehors des ganglions de la base que nous n'allons pas aborder dans ce manuscrit.

Il existe plusieurs hypothèses sur les origines physiologiques de la maladie de Parkinson. L'une d'entre elles est la destruction des neurones dopaminergiques de la substance noire, qui altère les émissions de potentiel d'action. De fortes synchronisations sont observées dans certaines structures des ganglions de la base, comme le noyau sous-thalamique, le globus pallidus externe et le striatum.

Dans les ganglions de la base, la voie directe favorise le mouvement, alors que la voie indirecte exerce un frein sur l'activité motrice. Si ces structures ne fonctionnent plus correctement, les mouvements sont affectés eux aussi. Des études ont montré que l'hyperactivité du striatum de la voie indirecte mène à un état parkinsonien chez les rongeurs [60]. Cette hyperactivité peut être liée aux renforcements synaptiques du striatum ou encore par des changements des canaux ioniques [15, 22, 34, 65].

Les boucles, entre le noyau sous-thalamique et le globus pallidus ou encore entre les arkypallidaux et les prototypiques, sont très propices aux synchronisations pathologiques dans les ganglions de la base [16, 69, 70, 82].

De nos jours, il n'existe pas de traitements pour guérir de la maladie de Parkinson. Les symptômes non moteurs [32], comme la dépression, les troubles du sommeil ou encore les ano-

malies sensorielles [61], sont très rarement traités conduisant à une mauvaise qualité de vie à long terme [24]. Les symptômes moteurs, comme le ralentissement du mouvement, la rigidité au niveau des muscles ou encore des tremblements au niveau des mains au repos, sont atténués par des traitements médicamenteux (e.g. la L-dopa) ou chirurgical (e.g. la stimulation cérébrale profonde). La L-dopa remédie au manque de dopamine en simulant l'action de la dopamine [74], malheureusement, elle provoque des effets secondaires, comme des nausées, des vomissements ou encore de la dyskinésie. La stimulation cérébrale profonde consiste en une chirurgie dans laquelle une (ou deux) électrode(s) sont implantée(s) dans le cerveau [32]. Ainsi, un courant continue est injecté dans les ganglions de la base, ce que nous appelons un système en boucle ouverte. Des effets secondaires ont été observés, comme des troubles cognitifs, comportementaux, ou la dépression [3, 93].

Objectifs de la thèse

L'objectif à long terme de ce projet est de développer un modèle de stimulation en boucle fermée. Cela consiste à stimuler les ganglions de la base en fonction du besoin du patient en temps réel, afin de diminuer au mieux les effets secondaire de la stimulation cérébrale profonde.

La finalité de cette thèse est d'élaborer un modèle suffisamment précis des ganglions de la base prenant en compte les différentes populations de neurones et leurs propriétés intrinsèques, afin d'améliorer, de tester et de valider des hypothèses ou/et des traitements. Ainsi, cette thèse a deux objectifs majeurs :

1. la réalisation d'un réseau de neurones, détaillés, de grande taille étant préalablement validé par des données biologiques
2. le développement d'un logiciel de simulation, rapide, et précis pouvant simuler des structures neuronales à taille réelle.

Le but est d'avoir une meilleur compréhension des mécanismes physiopathologiques à l'origine des synchronisation pathologiques dans la maladie de Parkinson et d'améliorer et de réaliser une première validation in silico des traitements. Pendant cette thèse, nous avons testé des hypothèses à l'origine des synchronisations pathologiques. Nous avons étudié le renforcement des circuits propices aux synchronisations pathologiques et ensuite, nous avons analysé les changements des propriétés intrinséques des neurones du striatum et du globus pallidus externe.

Plan de la thèse

Chapitre 1 : Du neurone au réseau en passant par la simulation de modèles mathématiques

Le premier chapitre est l'état de l'art des différents domaines traités pendant la thèse, afin de comprendre nos choix de modélisation et de simulation. Le chapitre est divisé en trois parties principales. La première partie, plus générale, avec les neurones et le réseau de neurones d'un point de vue biologique et mathématique appliqué aux neurosciences. La deuxième partie parle de la maladie de Parkinson, les ganglions de la base y sont détaillés, mais aussi les symptômes, l'origine physiologique et les traitements symptomatiques y sont abordés. Dans la dernière partie, la partie simulation, les méthodes numériques et les principaux simulateurs de réseau de neurones y sont décrits. Une brève comparaison des simulateurs y est donnée.

Chapitre 2 : Une nouvelle approche pour la simulation de réseaux de neurones de très grande taille sans stockage de connectivité

Le deuxième chapitre porte sur l'élaboration d'une méthode de simulation hybride, i.e. à pas de temps pour la dynamique du neurone et événementielle pour la dynamique synaptique, et où les connexions synaptiques du réseau ne sont pas stockées en mémoire, mais générées à chaque nouvel événement, afin de simuler des réseaux de neurones de très grandes tailles. À ce titre, le chapitre est divisé en trois parties. La première partie explique l'approche hybride de simulation et plus précisément les méthodes à pas de temps et événementielle. La deuxième partie porte sur la détection des événements où deux méthodes de calcul du temps du potentiel d'action y sont décrites. Dans la dernière partie est exposée la méthode utilisée pour générer la connectivité à chaque événement.

Chapitre 3 : Développement du logiciel SiReNe

Le troisième chapitre concerne le développement du logiciel **SiReNe**. Ce logiciel, développé depuis quelques années dans le laboratoire LORIA-INRIA, a été étendu à un simulateur hybride, expliqué dans le chapitre 2. Le logiciel, implémenté en C, est maintenant capable de simuler des grands réseaux neuronaux, soit avec la méthode classique à pas de temps, soit avec la méthode hybride. Nous avons aussi intégré le calcul parallèle et plus précisément, le multi-threading avec openMP et une version multi-machines avec MPI est en cours de développement. Ce chapitre se compose de trois sections principales. La première section détaille les différentes structures principales de **SiReNe**. La deuxième section décrit le fonctionnement du logiciel et la dernière section décrit sur le calcul parallèle implémenté dans **SiReNe**.

Chapitre 4 : Modélisation et simulation des neurones prototypiques et arkypallidaux de GPe

Le quatrième chapitre est sur la modélisation et la simulation des neurones prototypiques et arkypallidaux du globus pallidus. Nous avons fait le choix de distinguer les neurones prototypiques et les arkypallidaux, puisque les caractéristiques cellulaires et les spécificités de connectivité sont différentes pour ces deux populations de neurones. Ce chapitre se compose de quatre sections principales. La première section illustre les premiers résultats biologiques sur la distinction des deux populations du globus pallidus externe. La deuxième section décrit la modélisation des deux structures de neurones par le formalisme d'Hodgkin-Huxley. La troisième section porte sur l'ajustement de paramètres aux données biologiques par une optimisation avec l'algorithme d'évolution différentielle. La dernière section présente sur les résultats de simulation de ces deux populations du globus pallidus externe.

Chapitre 5 : Modélisation et simulation du réseau STN-GPeA/P-MSN

Le dernier chapitre porte sur la modélisation et la simulation du réseau des ganglions de la base. Ce chapitre se compose de trois sections principales. La première section illustre les résultats biologiques sur les connectivités entre les différentes populations de neurones des ganglions de la base et sur quelques résultats sur la maladie de Parkinson. La deuxième section décrit la modélisation du réseau complète. La troisième section est sur les résultats lorsque nous comparons notre modèle de réseau avec les résultats expérimentaux et ensuite lorsque nous simulons le réseau en conditions normales et parkinsoniennes.

1

Du neurone au réseau en passant par la simulation de modèles mathématiques

Sommaire

Les *neurones*, aussi appelés cellules nerveuses, sont un ensemble d'éléments du système nerveux qui gèrent les informations sensorielles, le mouvement musculaire et aussi le fonctionnement des organes. Ces neurones forment un *réseau complexe* qui est dense et compte plus de 10^{11}

cellules nerveuses et plusieurs kilomètres de connexions (dendrites et axones) dans le cerveau humain. Au cours d'une vie, l'homme perd plusieurs millions de neurones et, dans certains cas, ce peut être dû à des maladies neuro-dégénératives comme la *maladie de Parkinson*. Les tremblements des membres sont le principal symptôme moteur de cette maladie. La région du cerveau principalement atteinte par la maladie est celle des *ganglions de la base* où la destruction des neurones dopaminergiques est la plus importante ce qui génère des oscillations pathologiques, au niveau de cette structure. Afin d'identifier l'origine de ces rythmes pathologiques et d'améliorer des traitements, des hypothèses sont testées d'une part en expérimentation in vivo et d'autre part avec des *modèles mathématiques*. La *modélisation* permet de tester certaines hypothèses en amont des vérifications expérimentales. Ainsi les résultats sont simulés sur des *simulateurs* comme **NEURON** [50], **BRIAN** [47, 46] ou ***Si**mulateur de **Ré**seaux de **Ne**urones* (SiReNe) [13], ce dernier étant développé par notre équipe de recherche.

1.1 Neurones

1.1.1 Neurones biologiques

Anatomie du neurone

Un neurone se compose de trois parties ; les *dendrites*, le *soma* et l'*axone*. Chaque partie a une fonction différente. Les dendrites collectent les signaux d'entrée des autres neurones et transmettent les informations au soma. Le soma, le noyau du neurone, traite les signaux transmis par les dendrites. Si l'entrée totale, collectée par le soma, dépasse le seuil d'excitabilité, alors un signal de sortie est transmis à l'axone. L'axone reçoit le signal de sortie et le transmet aux autres neurones. La zone de contact entre deux neurones est appelée la *synapse* où le sens du flux d'information est de la terminaison de l'axone vers le neurone cible. Le neurone émetteur est la cellule *présynaptique* tandis que le neurone récepteur est considéré comme la cellule *post-synaptique* (voir Fig. 1.1). Un neurone peut être relié à plus de 10^4 neurones post-synaptiques qui peuvent se trouver, soit dans le voisinage direct du neurone soit dans une autre région du cerveau.

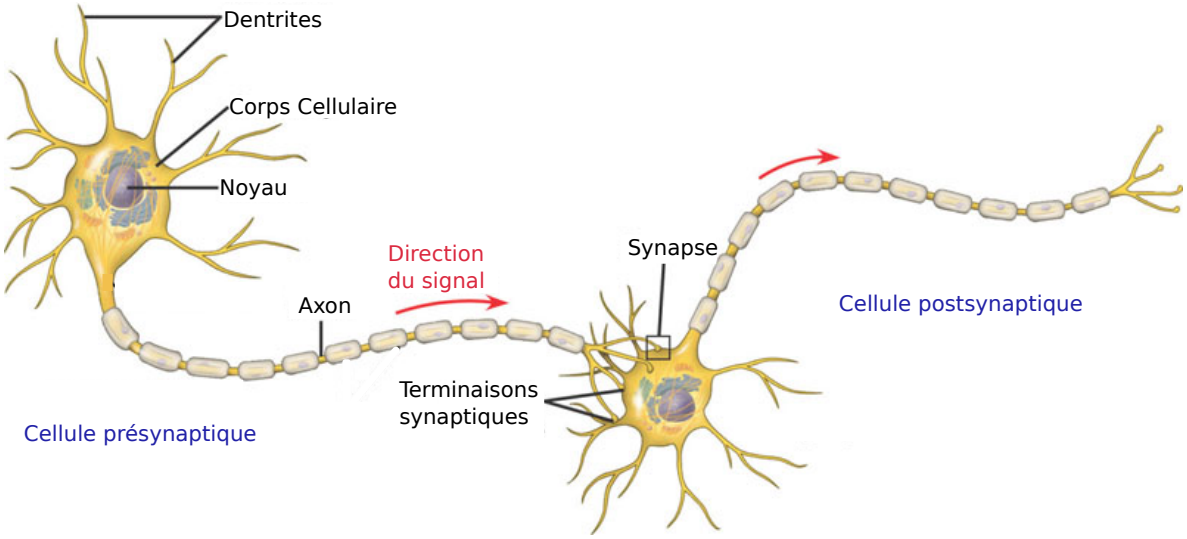

Fig. 1.1 – Structure biologique d'un neurone. Neurone pré-et post-synatique connecté par une synapse [9].

Une *membrane cellulaire* délimite le milieu *intra* et *extra-cellulaire* d'un neurone. Ces deux milieux cellulaires contiennent des particules chargées électriquement, des ions ; notamment des ions sodium ($\mathbf{Na}^+$), des ions potassium ($\mathbf{K}^+$), des ions calcium ($\mathbf{Ca}^{\mathbf{2}+}$) et aussi des ions chlore ($\mathbf{Cl}^-$) (voir Fig. 1.2).

Physiologie du neurone

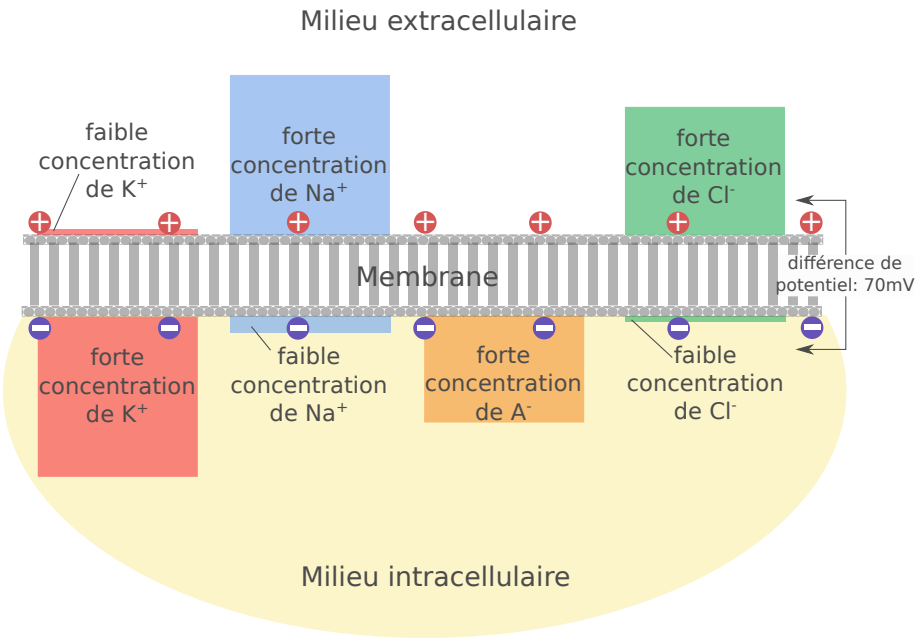

Fig. 1.2 – Concentrations ioniques d'un neurone au repos.

Les concentrations de ces ions sont différentes à l'intérieur et à l'extérieur d'un neurone, créant ainsi des *gradients électrochimiques*, à l'origine même de l'*activité neuronale*. Le milieu extracellulaire a une forte concentration de Na^+, de Cl^- et aussi de Ca^{2+}. Le milieu intracellulaire présente une forte concentration de K^+, mais aussi de molécules chargées négativement ($\mathbf{A}^-$) (voir Fig. 1.2). Au repos, le flux des ions Na^+ et Ca^2+ n'est pas important, mais le flux des ions K^+ et Cl^- l'est, ce qui conduit à une asymétrie de concentrations ioniques. Plus précisément, les A^- attirent plus de K^+ dans le neurone (puisque les ions opposés s'attirent) et repoussent plus de Cl^- hors du neurone, créant ainsi des gradients de concentration. Ces ions sont pompés dans le milieu intra-et extra-cellulaire par des *pompes ioniques*. À titre d'illustration, la pompe Na^+-K^+ pompe trois ions Na^+ pour deux ions K^+ introduits, maintenant ainsi les gradients de concentration.

Le *potentiel de membrane* est le résultat d'un déséquilibre de la concentration d'ions de part et d'autre de la membrane. À son état de repos, lorsque le neurone n'est soumis à aucune excitation, une différence de potentiel d'environ 70mV est constatée par rapport au milieu extérieur, de plus l'intérieur de la cellule est chargé négativement (voir Fig. 1.2). Ce potentiel de membrane, aussi appelé *potentiel de repos*, est d'environ -70mV.

Le *stimulus* provoque une activité neuronale qui doit être suffisamment intense pour déclencher un influx nerveux. Un ***P**otentiel d'**A**ction (PA)* naît et se caractérise par un changement de polarité électrique de part et d'autre de la membrane cellulaire qui passe de $-70mV$ à $+40mV$, par exemple. Cette inversion de polarité est la conséquence de mouvements d'ions à travers la

membrane par des *canaux ioniques*. Ce passage d'ions se réalise selon le sens de son gradient électrochimique. De plus, si le gradient s'annule, alors le mouvement transmembranaire s'arrête et lorsque le gradient s'inverse, alors le mouvement s'inverse également. Certains canaux n'acceptent qu'un certain type d'ions, comme e.g. les canaux Na^+, Ca^{2+} et K^+.

Au départ, le potentiel de membrane V_m est à son état de repos à environ $-70mV$ (Fig. 1.3(a)). Lorsqu'une stimulation arrive au neurone, la membrane se dépolarise légèrement et si le potentiel de membrane atteint le potentiel de seuil. À ce moment, les canaux Na^+ s'ouvrent et les ions Na^+ traversent les canaux vers l'intérieur de la membrane (Fig. 1.3(b)). Ce chargement en ions positifs permet à la membrane de changer de polarité vers des valeurs plus positives. Ce phénomène est appelé la *dépolarisation* de la membrane qui peut atteindre jusqu'à $+40mV$. Entre-temps, le gradient électrochimique s'inverse et l'entrée des ions Na^+ diminue (Fig. 1.3(c)). Cependant les canaux K^+ s'ouvrent, car la membrane est dépolarisée. Les ions K^+ s'échappe massivement vers l'extérieur de la membrane (Fig. 1.3(d)), si bien que ces canaux s'inactivent et bloquent l'entrée des ions Na^+. Toutefois, les canaux K^+ sont plus lents à la fermeture et laissent échapper un grand nombre de K^+. Ainsi, l'extérieur de la membrane devient plus positif que l'intérieur et le potentiel de membrane passe sous le seuil de repos (Fig. 1.3(e)). Pendant que les canaux K^+ se ferment, la membrane s'*hyperpolarise*. Après la fermeture de tous les canaux, la membrane retrouve son potentiel de repos (Fig. 1.3(e+a)). Ce potentiel d'action est ensuite transmis aux autres neurones par les synapses.

1.1.2 Modèles mathématiques

Les modèles de neurones biologiques modélisent certaines propriétés du neurone, e.g. les PA ou les propriétés des canaux ioniques, afin de comprendre et analyser certains comportement neuronaux. En modélisation, un neurone est décrit comme un système électrique et depuis quelques décennies, plusieurs modèles ont été introduits, des plus simplistes aux plus complexes. Dans cette section, les modèles dits *Intègre-et-tire à fuite*, en anglais ***L**eaky **I**ntegrate-and-**F**ire*, (LIF) et ***H**odgkin-**H**uxley (HH)* vont être présentés.

Modèle Intègre-et-tire à fuite (en anglais «Leaky Integrate-and-Fire», LIF)

Le neurone i émet un PA lorsqu'un seuil Θ est atteint par le potentiel de membrane V_i au temps t. Les modèles de neurones, où le PA est décrit comme un événement, sont appelés des modèles d'*Intègre-et-tire*, en anglais ***I**ntegrate-and-**F**ire*, (IF). Les modèles IF définissent la dynamique d'un neurone par deux éléments ; le premier étant une équation qui décrit l'évolution du potentiel de membrane dans le temps $V_i(t)$ et le deuxième un mécanisme pour générer les PA [41].

L'activité neuronale est décrite par des lois élémentaires de la *théorie de l'électricité*. La membrane du neurone, qui est un assez bon isolant, agit comme un condensateur avec une capacité C_m. Cet isolant n'est pas parfait et laisse échapper la charge avec le temps. Ainsi, un courant fuit à travers la membrane et celle-ci est alors caractérisée par une résistance R. Par conséquent, le modèle LIF [85] consiste en un condensateur C_m en parallèle avec une résistance, R, où un courant extérieur, $I(t)$, est injecté au neurone. La Fig. 1.4 montre le circuit électrique de l'activité neuronale d'un modèle LIF.

La loi de conservation du courant est exprimée comme la somme des courants ; le courant de fuite ("leak"), I_L, et celui de la capacité I_C, c'est-à-dire $I(t) = I_L + I_C$. Le courant qui passe par la résistance R suit la loi d'Ohm $I_L = \dfrac{E_R}{R}$ où $E_R = V(t) - E_L$ et $R = \dfrac{1}{g_L}$. Cependant,

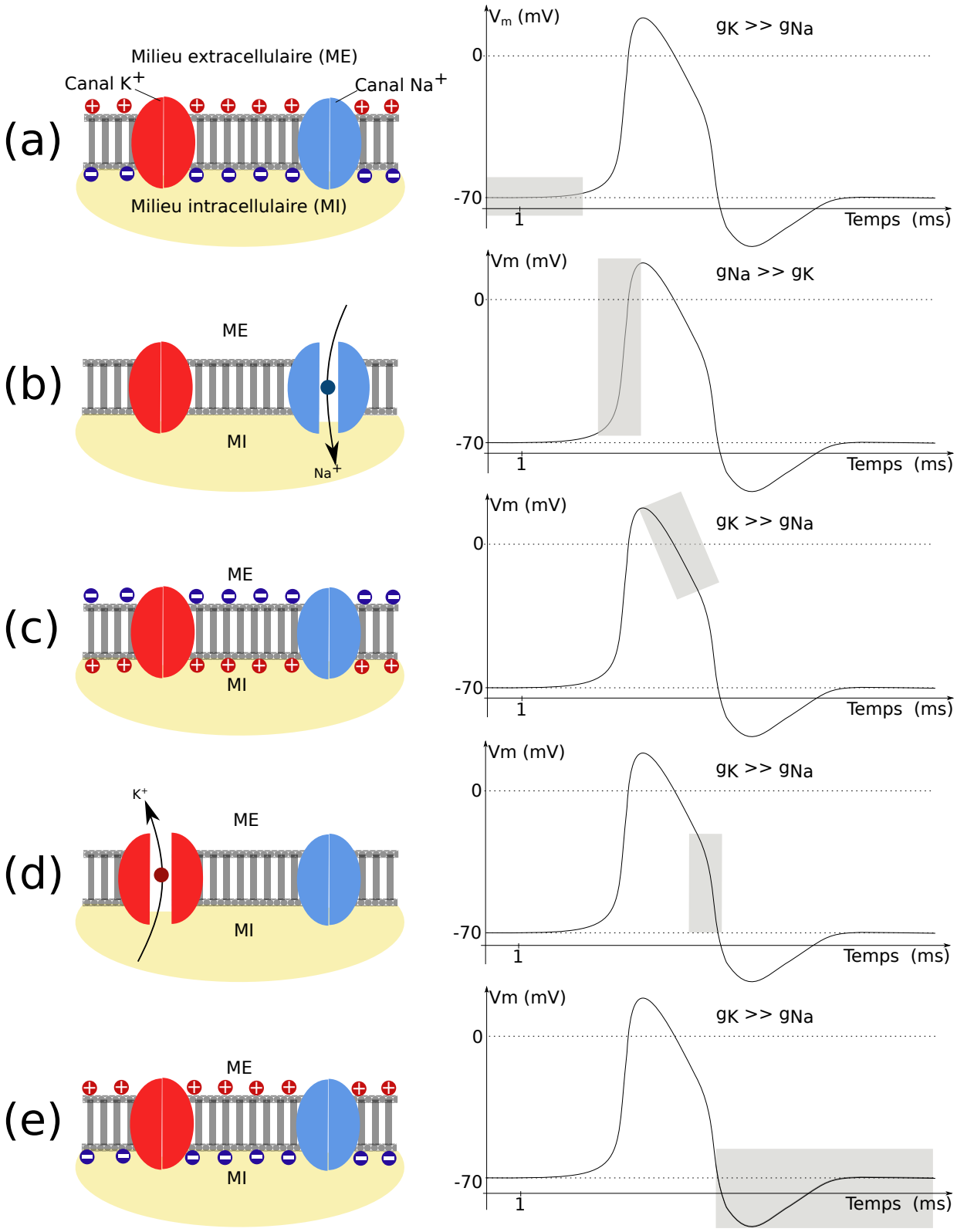

Fig. 1.3 – Illustrations d'ouverture et de fermeture de canaux ioniques et du déclenchement du potentiel d'action.

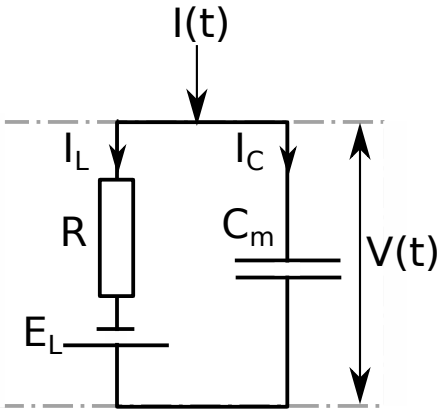

Fig. 1.4 – Représentation de l'activité neuronale du modèle LIF par un circuit électrique.

le deuxième terme de la somme est déterminé par la définition de la capacité $C_m = \frac{q}{V}$, où q est la charge. Par conséquent, $I_C = \frac{dq}{dt} = C_m \frac{dV}{dt}$. Le deuxième élément du modèle LIF est la détermination du temps du PA. Celui-ci est défini par un critère de seuil ($\Theta = -50mV$). Lorsque le potentiel de la membrane dépasse le seuil, alors le neurone i a généré un potentiel d'action au temps $t^{(f)}$. Après cet événement, le potentiel est ramené à sa valeur de reset ($V_r = -70mV$). Ainsi le modèle LIF est défini comme suit :

$$\begin{cases} C_m \dfrac{dV}{dt} & = & I(t) - g_L \left[V(t) - E_L \right] & & \\ t^{(f)} \quad : \quad V\left(t^{(f)}\right) & = & \Theta & \textit{(Passage au seuil)} & \\ t^{(f)} \quad : \quad \lim\limits_{\delta \to 0; \delta > 0} V\left(t^{(f)} + \delta\right) & = & V_r & \textit{(Reset du potentiel)} & \end{cases} \tag{1.1}$$

Inconvénients et intérêts du modèle LIF. Le modèle LIF ne peut pas expliquer la biochimie et la biophysique des neurones, comme e.g., l'ouverture et la fermeture des canaux ioniques ne sont pas prises en compte. Avec ce genre de modèle, les interactions non linéaires, des courants actifs dans certaines parties du neurone, ne sont pas retenues. Par contre, le modèle est précis lorsqu'un PA est généré et le temps du PA est défini très précisément.

Modèle de Hodgkin-Huxley (HH)

Dans la section précédente, un des points négatifs du modèle LIF était qu'il ne prenait pas en compte la biochimie et la biophysique des neurones. Néanmoins, il existe des modèles, comme celui d'Hodgkin-Huxley (HH) [51], dans lesquels les canaux ioniques sont considérés dans l'activité du neurone. Un résultat important du travail d'HH est que *les neurones sont des systèmes dynamiques*. Un système dynamique consiste en un ensemble de variables qui est décrit par son état à chaque pas de temps, plus précisément l'état du système à l'instant suivant dépend de l'entrée et de son état eux instants précédents. Le modèle basique d'HH, décrivant les trois types de canaux ioniques ; les canaux sodiques Na, potassiques K et de fuite ("leak") L, résulte d'une expérimentation dont la caractérisation résulte d'une expérimentation par current clamp et, par la suite, d'une modélisation de l'axone géant d'un calamar. Néanmoins, le modèle peut être étendu à d'autres canaux [41, 51].

Le modèle de HH permet d'étudier l'influence de chaque canal ionique sur la dynamique du réseau. D'après [15, 22, 34, 65], les canaux calciques et les canaux potassiques changent de comportement lors de la **M**aladie de **P**arkinson (MP) (voir section 1.4 sur la MP). De plus, la forme du PA est bien décrite avec ce type de modèle détaillé permettant d'étudier un phénomène appelé le «sag» se produisant lorsque le canal HCN est activé (plus de détails seront donnés dans le chapitre 4). Toutes ces propriétés sont très importantes pour l'élaboration de notre modèle détaillé et afin de tester des hypothèses concernant la maladie de Parkinson.

Le flux net des ions à travers leurs canaux ioniques respectifs est proportionnel à un gradient électrochimique comme décrit dans la section 1.1.1. Le gradient électrochimique s'exprime par la différence de potentiel de membrane (E) et le potentiel d'équilibre de l'ion en question (E_{ion}). Si $E < E_{ion}$, alors le flux est sortant, sinon si $E > E_{ion}$, alors le flux est entrant et pour $E = E_{ion}$, alors l'état d'équilibre est atteint et le gradient électrochimique est nul.

Tout comme dans le modèle LIF, la membrane cellulaire est représentée par un condensateur. Quant aux canaux ioniques, ils sont représentés par des résistances non fixes, puisque la valeur de la résistance va varier selon que le canal ionique est ouvert ou fermé. Le potentiel, généré par la différence de concentration d'ions de part et d'autre de la membrane, est représenté par une batterie. Ainsi la représentation en système électrique du neurone est complétée par un ajout de batterie pour chaque canal ionique (voir Figs. 1.5 et 1.6).

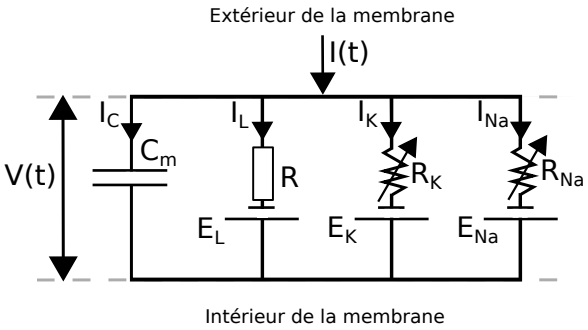

Fig. 1.5 – Circuit électrique représentant l'activité neuronale d'après HH.

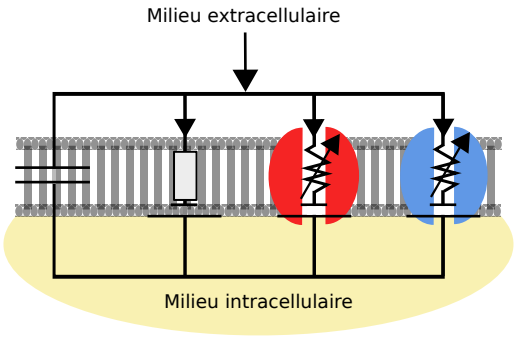

Fig. 1.6 – Comparaison entre la physiologie d'un neurone et le circuit électrique d'après HH.

Traduisons le circuit électrique en équations mathématiques. Comme dans la section précédente sur le modèle LIF, la loi de conservation du courant est appliquée. Ainsi le courant appliqué $I\,(t)$ est divisé en un courant capacitif I_C et la somme des courants ioniques $I_{ions} = I_{Na} + I_K + I_L$ qui passent à travers la membrane du neurone. Comme défini précédemment, le courant de la capacité est défini comme suit : $I_C = \dfrac{dq}{dt} = C_m \dfrac{dV}{dt}$, avec C_m la capacité de la membrane. Pour résumé, nous avons :

$$C_m \frac{dV}{dt} = I\,(t) - I_{Na} - I_K - I_L$$

En termes biologiques, V est le potentiel de la membrane. Chaque équation des canaux ioniques est définie par leur résistance ou par leur conductance.

Le courant de fuite est modélisé par sa conductance que l'on définit par $g_L = \dfrac{1}{R}$ et en appliquant la loi d'Ohm, le courant de fuite est le suivant : $I_L = g_L\,(V(t) - E_L)$.

Les autres courants sont définis comme le courant fuite sauf que les conductances sont dépendantes du temps. Hodgkin et Huxley ont réussi à définir par des équations mathématiques l'ouverture et la fermeture des canaux ioniques, ce qu'ils vont appeler par la suite les *«gating variables»* («gate» signifiant porte en anglais). Il existe deux types de «gating variables» ; les variables actives (ou ouvrant les canaux ioniques) et les variables inactives (ou fermant les canaux). La probabilité, que l'«activation gate» soit à l'état ouvert, est notée par la variable m (ou n), cependant la probabilité, que l'«inactivation gate» soit à l'état ouvert, est notée par la variable h. Ainsi, la proportion de canaux ouverts est désignée par $m^a h^b$ où a est le nombre d'«activation gate» et b le nombre d'«inactivation gate» par canaux ioniques. Les canaux peuvent être, soit

- pas activés (ou désactivés) $(m = 0)$;
- partiellement activés $(0 < m < 1)$;
- complètement activés $(m = 1)$;
- inactivés $(h = 0)$;
- partiellement inactivés $(0 < h < 1)$;
- sortis de l'inactivité (ou désinactivé) $(h = 1)$.

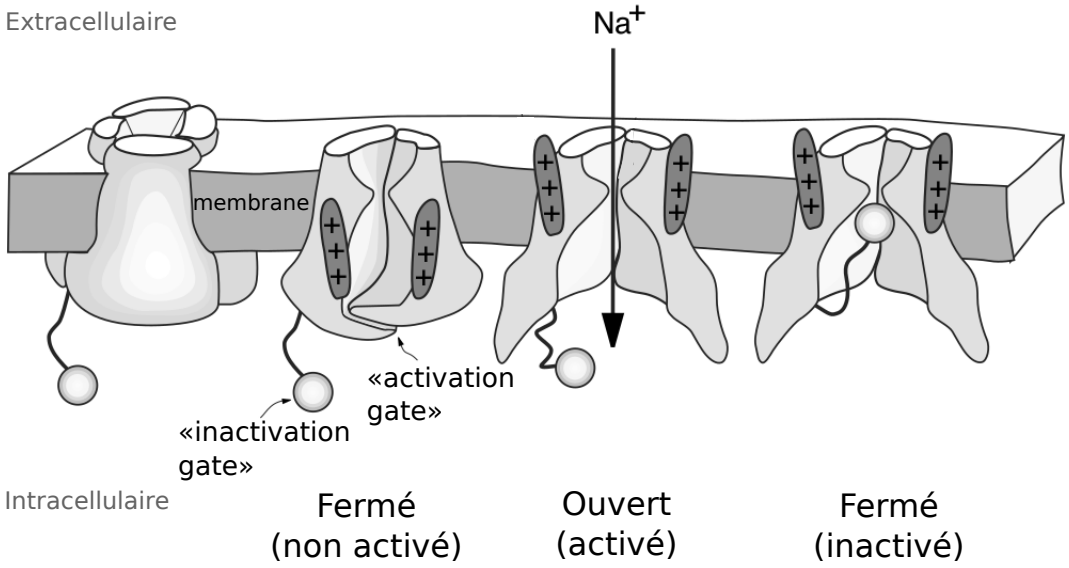

Fig. 1.7 – Structure des canaux ioniques dépendant des «gating variables». L'«activation gate» s'ouvre et les ions sélectionnés circulent dans le sens du gradient électrochimiques. L'«inactivation gate» bloque le canal. [54]

Le courant Na a une variable d'activation m et une variable d'inactivation h, cependant le courant K n'a qu'une variable d'activation n. Finalement, l'équation d'Hodgkin-Huxley est formulée de la manière suivante :

$$C_m \frac{dV}{dt} = I(t) - g_{Na} m^3 h (V(t) - E_{Na}) - g_K n^4 (V(t) - E_K) - g_L (V(t) - E_L) \tag{1.2}$$

Les équations des courants peuvent être étendues et adaptées à d'autres courants ioniques, e.g. calciques, potassiques dépendant du calcium, etc. La méthode est alors appelée le formalisme de HH. Les dynamiques des variables d'activation et d'inactivation sont déterminées par des équations différentielles de premier degré, de la forme :

$$\frac{dx}{dt} = \frac{x_\infty(V) - x(t)}{\tau_x(V)} \tag{1.3}$$

avec $x = m, h, n$. La fonction d'in/activation est notée $x_\infty(V)$, à l'état stable, et dépendante du potentiel et $\tau_x(V)$ la *constante de temps*. Les variables $x_\infty(V)$ et $\tau_x(V)$ sont définis par des équations que je ne spécifierai pas dans ce manuscrit, mais que vous pouvez consulter dans les articles [51] et [41].

Finalement, le modèle de HH est défini par les équations 1.2 et 1.3 :

$$\begin{cases} C_m \dfrac{dV}{dt} = I(t) - g_{Na} m^3 h \left(V(t) - E_{Na}\right) - g_K n^4 \left(V(t) - E_K\right) - g_L \left(V(t) - E_L\right) \\ \dfrac{dm}{dt} = \dfrac{m_\infty(V) - m(t)}{\tau_m(V)} \\ \dfrac{dh}{dt} = \dfrac{h_\infty(V) - h(t)}{\tau_h(V)} \\ \dfrac{dn}{dt} = \dfrac{n_\infty(V) - n(t)}{\tau_n(V)} \end{cases} \tag{1.4}$$

Le modèle de HH est un *système dynamique quadridimensionnel*, i.e. que son état est déterminé de manière unique par le potentiel de membrane V, et les «gating variables» n, m et h pour les courants K et Na. Dans le cas du formalisme d'HH, le système dynamique est de dimension supérieure, i.e. qu'en effet, nous y ajoutons autant de dimension que de «gating variables» pour chaque canal ionique présent.

Les inconvénients et les intérêts du modèle HH. Le modèle de HH peut être étendu à d'autres canaux ioniques. Il est souvent utilisé pour modéliser des neurones à l'échelle cellulaire ou sur des réseaux à petite échelle (< 200 neurones). Le formalisme HH est le modèle qui reproduit le mieux les neurones au niveau physiologique. Toujours est-il que plus de canaux ioniques sont ajoutés plus de paramètres devront être réajustés et plus d'équations différentielles devront être résolues. Par conséquent, le temps de simulation et l'allocation de mémoire sont aussi plus importants.

Autres modèles

D'autres modèles dérivants, principalement, des modèles LIF et HH existent, mais ils ne seront pas détaillés dans ce manuscrit. **IF** [62] modélise l'activité électrique d'un neurone par un modèle de charge et de décharge d'un condensateur. À l'inverse du modèle de LIF, il n'existe pas de courant de fuite dans ce modèle. Ainsi le modèle IF est défini par l'équation $C_m \dfrac{dV}{dt} = I(t)$ et par le critère de seuil vu dans le modèle LIF équation (1.1). ***Résonne-et-tire*, en anglais *Resonate-and-Fire*, (RF)** [94] est un modèle, à deux variable, plus réaliste du modèle LIF dans lequel un courant K^+ persistant, à faible seuil de déclenchement (ou tout autre courant qui est partiellement activé au repos), a été ajouté. ***IF quadratique*** reprend le modèle LIF, mais le potentiel de membrane de l'équation (1.1), «$-V(t)$», est élevé au carré «$V^2(t)$». Le ***modèle d'Izhikevich*** [56] est une combinaison des deux modèles précédents, i.e. à la fois le potentiel de membrane est élevé au carré et il introduit le courant K^+.

1.1.3 Problématiques

Le modèle IF est plus rapide à résoudre et donc son temps de simulation est plus court que celui de HH (voir le tableau 1 de l'article [23]). L'inconvénient est que la dynamique du neurone n'est pas modélisée de façon réaliste contrairement au modèle de HH.

Dans toute cette section, nous avons présenté un modèle de neurones à compartiment unique, i.e. nous modélisons uniquement une partie du neurone biophysique. Cependant il est possible de modéliser plusieurs compartiments en calculant pour chaque compartiment les équations différentielles (1.1) ou (1.4). Lorsque nous modélisons plusieurs compartiments, par exemple un compartiment pour le soma et un compartiment pour la dendrite, le modèle est plus réaliste, toutefois le temps de simulation est rallongé considérablement.

1.2 Réseaux

Le cerveau contient des milliards de neurones, divisé en plusieurs zones anatomiques et histologiques qui sont elles-mêmes aussi subdivisées en sous-régions. Dans la section précédente, nous avons abordé le sujet du neurone, maintenant nous passons à une échelle plus large du cerveau, le *système nerveux* [14].

1.2.1 Systèmes nerveux biologiques

Transmission d'information

D'un point de vue physiologique, lorsque le potentiel membranaire dépasse un certain seuil, alors un PA est généré. Ce PA transmet une information au système nerveux, donc il est nécessaire qu'elle soit envoyée à d'autres neurones, p. ex. les neurones moteurs, responsable des contractions musculaires, ou encore aux neurones du cerveau ou de la moelle épinière responsable des réflexes coordonnés. Ce transfert d'informations, d'un neurone à l'autre, se fait à un niveau spécifique du neurone, nommé *synapse* et le processus est appelé la *transmission synaptique*.

Synpases

Il existe deux types de synapses ; les *synapses électriques* et les *synapses chimiques*, qui nous intéressent tout particulièrement. Les synapses électriques ont un fonctionnement et une structure simples. Le courant ionique passe directement d'une cellule à l'autre et cette transmission est souvent sûre et très rapide. De plus, le PA d'un neurone présynaptique produit presque directement un PA au neurone postsynaptique. Par contre, les synapses chimiques, qui sont un mécanisme de transmission plus complexe, ont besoin d'un intermédiaire pour transmettre l'information qui est appelé, le *neurotransmetteur*. Lorsque l'activité électrique arrive dans la terminaison axonale présynaptique, alors les vésicules synaptiques sont stimulées et elles libèrent des neurotransmetteurs dans l'espace synaptique au moment d'un PA. Les récepteurs, se trouvant dans la membrane postsynaptique, sont activés et entraînent l'ouverture de certains canaux ioniques. Un courant transmembranaire postsynaptique *excitateur* ou *inhibiteur* (*EPSC* ou *IPSC*) se crée.

Lors de la MP, les synapses rapides, $GABA_A$ et AMPA, sont majoritairement atteintes par la maladie et donc elles sont aussi ciblés par les traitements médicamenteux. Ainsi, nous avons fait le choix de modéliser nos synapses par une fonction exponentielle qui défini bien les synapses rapides des neurones des **G**anglions de la **B**ase (GB), voir Fig. 1.8.

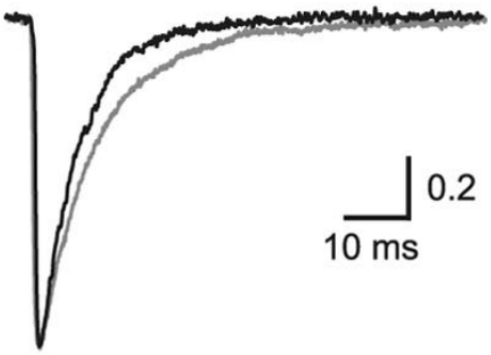

Fig. 1.8 – Courant synaptiques inhibiteur d'un neurone du STN [37].

Connectivité

La *connectivité*, entre les neurones de même groupe ou de différents types, est souvent incertaine dans le cas réel, puisque les données expérimentales sont restreintes. Les connexions sont, soit dépendantes de la distance, soit uniformes dans le voisinage d'une colonne de neurones. La probabilité de connexions, i.e. la probabilité qu'un neurone établisse une connexion avec un autre neurone au voisinage d'une colonne, peut varier en fonction des différents types de neurones.

1.2.2 Modèles mathématiques

Synapses

Les modèles mathématiques de synapses ne sont pas directement basés sur la concentration des récepteurs dans la fente synaptique. L'activation des canaux ioniques par le récepteur est décrite comme une conductivité dépendante du temps $g_{syn}(t)$ qui s'activera dès l'arrivée d'un PA. Le courant synaptique dépend de la différence entre le potentiel d'inversion de la synapse E_{syn} et la valeur du potentiel membranaire [41].

$$I_{syn}(t) = g_{syn}(t)(V - E_{syn}) \tag{1.5}$$

Le potentiel d'inversion E_{syn} et la conductance synaptique $g_{syn}(t)$ sont utilisés pour définir différents types de synapses. P. ex., le potentiel d'inversion d'une synapse inhibitrice est fixé à environ $-75mV$, ce qui correspond au potentiel d'inversion des Cl^- et celui d'une synapse excitatrice à $0mV$. L'évolution temporelle de la conductance synaptique, $g_{syn,jk}(t)$ entre le neurone j et le neurone postsynaptique k, est modélisée par l'équation différentielle suivante :

$$\frac{dg_{syn,jk}(t)}{dt} = -\frac{g_{syn,jk}(t)}{\tau_{syn,jk}} \tag{1.6}$$

où $\tau_{syn,jk}$ est la constante de temps de la synapse.

Une superposition d'exponentielle, est obtenue, en intégrant, l'équation (1.6) entre t, le temps courant, et t_{sp}, le temps du spike :

$$g_{syn,jk}(t) = \overline{g}_{syn,jk} \exp\left(-\frac{t - t_{sp}}{\tau_{syn,jk}}\right) \tag{1.7}$$

où $\overline{g}_{syn,jk} = g_{syn,jk}(t_{sp})$, la conductance maximale. Le courant synaptique du neurone k est alors la somme de tous les courants synaptiques des neurones présynaptiques j et de tous leurs PA présynaptiques. Remplaçons $g_{syn,jk}(t)$ de l'équation (1.5) par l'équation (1.7), nous obtenons :

$$I_{syn,k}(t) = \sum_j \overline{g}_{syn,jk}(V_k - E_{syn,jk}) \sum_{i=1}^{n_j} \exp\left(-\frac{t - t_{sp_i}}{\tau_{syn,jk}}\right) \tag{1.8}$$

Une fonction bi-exponentielle est appliquée pour certains types de synapses pour lesquelles une seule décroissance exponentielle ne suffit pas. Le courant synaptique est plutôt constitué de deux composantes différentes, une rapide avec une constante de temps, $\tau_{syn,jk,1}$, de décroissance de quelques millisecondes, et une seconde, avec constante de temps $\tau_{syn,jk,2}$, dix fois plus lente.

$$I_{syn,k}(t) = \sum_j \overline{g}_{syn,jk}(V_k - E_{syn,jk}) \sum_{i=1}^{n_j} \exp\left(-\frac{t - t_{sp_i}}{\tau_{syn,jk,1}}\right) + \exp\left(-\frac{t - t_{sp_i}}{\tau_{syn,jk,2}}\right) \tag{1.9}$$

Connectivité

En simulation, les neurones sont généralement choisis aléatoirement au sein de leur population ou/et d'autres groupes et il existe plusieurs schémas de connectivité. Ainsi, il faut adapter le schéma de connectivité à la simulation, i.e. si le réseau n'est pas assez dense, alors le schéma de connectivité complet est le plus adapté. La connectivité complète consiste à connecter tous les neurones avec tout le monde. Dans ce cas, il faut préciser si le neurone peut être connecté avec lui-même ou pas. Néanmoins, la probabilité de connexion peut être fixée et les neurones sont choisis aléatoirement avec une probabilité p. Lorsque les neurones sont fixés dans l'espace, une autre solution de connexion est de générer une connectivité dépendante de la distance entre les neurones [41].

1.2.3 Problématiques

D'après la Fig. 1.9, montrant plusieurs observations expérimentales sur plusieurs espèces différentes, en moyenne le réseau de n neurones a au plus $n^{1.4}$ connexions. Lorsque n neurones et $n^{1.4}$ connexions sont stockés, deux problèmes surgissent ; premièrement, la mémoire de l'ordinateur va vite saturer et deuxièmement, le temps de simulation augmente considérablement aussi. Une solution est d'éviter le stockage de la connectivité et de la générer aléatoirement à chaque pas de temps ou à chaque événement, pour être encore plus efficace.

Les solutions à ces problèmes vont être abordées dans la suite du manuscrit.

1.3 Ganglions de la base *(en anglais Basal Ganglia)*

Le terme anglais "basal" signifie que la structure, e.g. des **GB**, est profondément implantée dans le cerveau. Les GB sont principalement connectés avec le cortex cérébral, le thalamus, et le tronc cérébral, entre autres. Cette partie du cerveau a la fonction de traiter les informations relatives au mouvement, i.e. ils affinent et ajustent les actions de mouvements simples [14].

1.3.1 Différentes structures

Les GB sont essentiellement constitués d'un ensemble de structures nerveuses, dont : le *noyau sous-thalamique*, en anglais ***S**ub**T**halamic **N**ucleus*, (STN), le ***STR**iatum (STR)* et le ***G**lobus*

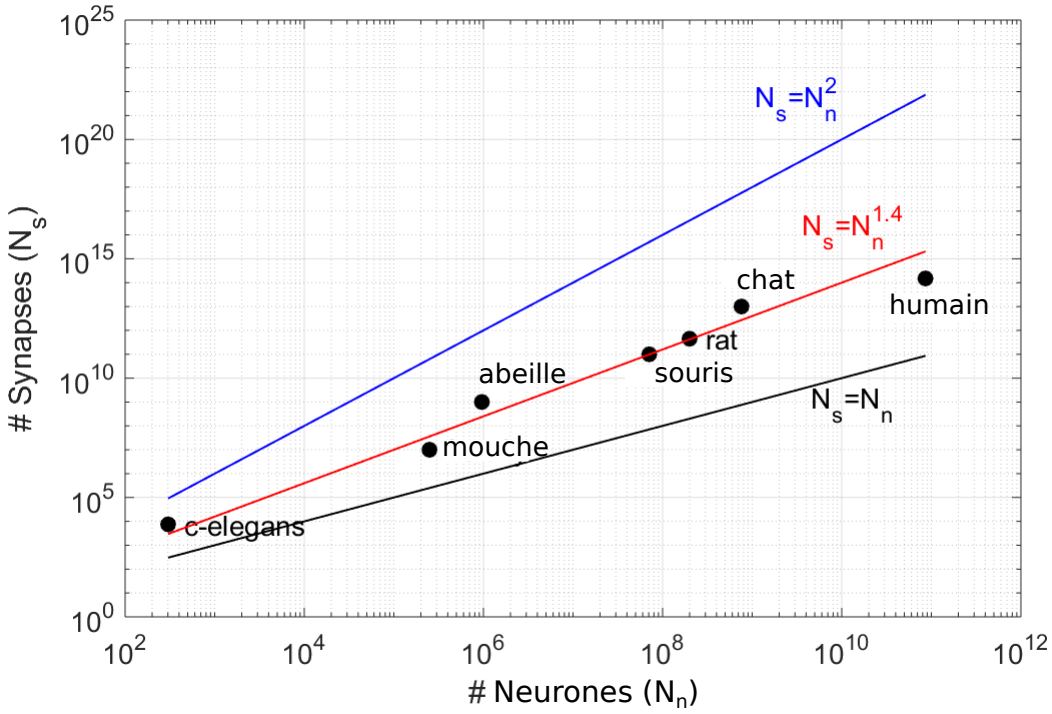

Fig. 1.9 – Nombre de synapses N_s en fonction du nombre de neurones N_n pour différentes espèces animales (vertébrées et inverébrees) [13].

***P**allidus (GP)*, qui lui-même comprend deux structures, ***GP i**nterne (GPi)* et ***GP e**xterne (GPe)*. Récemment, des résultats [7, 69] ont montré qu'il existe deux populations distinctes de GPe avec une activité neuronale et une connectivité différente, les ***GPe A**rkypallidaux (GPeA)* et les ***GPe P**rototypiques (GPeP)*. Les GPeP se projettent sur deux structures bien distinctes, les STN pour les GPeP et le STR pour les GPeA et les deux structures de GPe présentent des fréquences de décharge différentes qui reflètent des propriétés cellulaires distinctes. Cette fréquence est 2 à 3 fois plus élevée pour les GPeP [7]. Dans le STR, nous trouvons principalement des neurones du ***M**edium **S**piny **N**eurons (MSN)* à 95% et des neurones du ***F**ast-**S**piking **N**eurons (FSN)*. Dans la structure MSN, il existe deux types de récepteurs de dopamine ; les ***MSN** à récepteur du type **D1** (MSN-D1)* et les ***MSN** à récepteur du type **D2** (MSN-D2)* [40]. Dans certains cas, la ***S**ubstance **N**oire (SN)* est associée au GB, bien que cette structure ne fait pas partie des GB, mais est majoritairement connectée avec ces noyaux (voir Fig. 1.10).

La voie directe, impliquant les neurones dopaminergiques MSN-D1, passe par les neurones du SN et du GPi et la voie indirecte, avec les MSN-D2, passe par contre par les neurones du GPe et du STN [68]. Dans le STR, les deux voies, directe et indirecte, exercent un contrôle opposé sur l'exécution des mouvements. La voie directe favorise le mouvement, alors que la voie indirecte exerce un frein sur l'activité motrice [68].

Le Tab. 1.1 résume les structures, des GB, modélisées pour la thèse ; avec la référence des modèles sur lesquelles nous nous sommes basés ; le nombre de neurones et aussi les canaux ioniques modélisés.

Le STR est la cible majeure des connexions corticales dans les GB et le GP se projette principalement dans le thalamus. Les autres structures se connectent dans les GB pour moduler

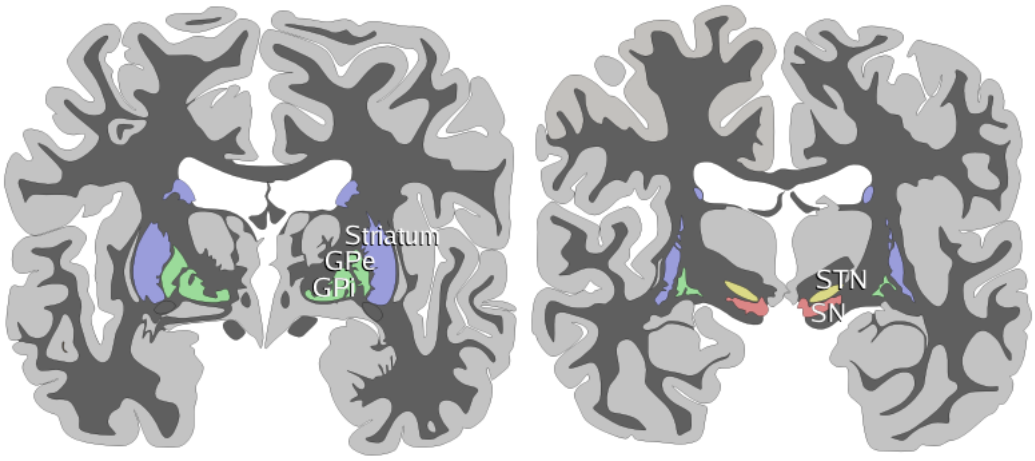

Fig. 1.10 – Deux coupes du cerveau humain identifiant les GB ; en bleu le STR, en vert le GP (externe et interne ; vue du dessus), en jaune le STN et en rouge la SN (vue du dessous). La coupe de droite est plus profonde, plus proche du tronc cérébral [42].

Structures des GB et nombre de neurones chez le rat d'après [79]	**Courants respectifs**
STN [90] *(13 600 neurones)*	I_L, I_K, I_{Na}, I_T (*courant* Ca^{2+} *de type T à seuil bas*), I_{Ca} (*courant* Ca^{2+} *de type T à seuil élevé*), I_{AHP} (*«After Hyper-Polarisation Current»* K^+)
GPe(P/A) [39] *(GPeA : 11 500 neurones et GPeP : 34 500 neurones)*	I_L, I_{NaF} (*courant Na transitoire rapide*), I_{NaP} (*courant Na persistant*), I_{Kv} (*courant K voltage dépendant*), I_{HCN} (*courant de cation modulé par les nucléotides cycliques activé par l'hyperpolarisation*), I_{SK} (*courant de K dépendant du Ca*)
MSN-D2 [72] *(1 300 000 neurones)*	I_L, I_{Na}, I_K, I_M (courant M potassique non inactivant)

Tab. 1.1 – Les différentes populations des GB de notre modèle. Sur chaque population de neurones, nous retrouvons les articles sur lesquels nous nous sommes basés avec leurs nombres de neurones respectifs. Les courants ioniques modélisés sont donnés dans la deuxième colonne.

l'activité du circuit.

Les GB sont responsables des mouvements volontaires, comme dit dans le début de la section, ainsi plusieurs types de maladies sont reliées au mauvais fonctionnement des GB. Deux cas peuvent survenir lors d'une activité irrégulière dans les GB. Le premier cas est une inhibition trop forte du thalamus suite à un dysfonctionnement dans le GB, ce qui engendrait un ralentissement des mouvements ou une *hypokinésie*. Le deuxième cas est une diminution des influences synaptiques sur le thalamus, i.e. une diminution des connexions inhibitrice, ce qui provoquerait un excès de mouvement ou une *hyperkinésie*.

1.3.2 Problématiques

Première problématique. Chez le rat, le STR a environ 1,3 million neurones et 59 600 neurones pour les autres structures STN+GPeA/P [79]. Dans la littérature [26, 72, 90], plusieurs modèles ont été simulés, et validés, avec environ 100 neurones. Notre objective est de valider notre modèle en gardant d'un côté une proportion de neurones correcte et de l'autre avoir une connectivité réaliste. Afin de tester rapidement nos hypothèses, nous pourrions simuler une petite partie du réseau, mais nous allons faire face à un problème, par exemple, si nous simulons 1/1000 du réseau et nous voulons garder la proportion de neurones entre les structures, alors nous devons simuler 1300 neurones dans le STR, 14 neurones dans le STN et 12 dans le GPeA et 35 neurones dans le GPeP, mais, dans ce cas, nous sommes contraints de connecter tous les neurones entre eux, à cause du faible nombre de neurones dans les structures STN+GPeA/P. De notre point de vue, la connectivité complète ne reflète pas correctement la réalité d'un réseau et, de plus, certaines hypothèses, basées sur la connectivité, ne pourraient pas être testées. La solution est, soit de simuler plus de neurones et prendre le risque d'accroître le temps de simulation, soit de lancer les simulations sur des logiciels parallélisés en multiprocessus ou multimachines.

Deuxième problématique. Les modèles de GB proposés dans la littérature, de nos jours, ne sont pas assez rigoureux. Ils ne font pas la distinction entre les deux populations des GPe les GPeA et GPeP [26, 53, 67, 76, 90]. Alors que celles-ci reflètent d'une part de propriétés cellulaires distinctes et de l'autre, elles se projettent sur des structures différentes. De plus, quand nous cherchons à modéliser un neurone en se basant sur des données physiologiques et anatomiques, forcément il faut modéliser le plus de canaux ioniques possibles. Le problème s'est que plus nous ajoutons de canaux ioniques à notre modèle plus il faut résoudre d'équations différentielles non-linéaires et donc notre temps de simulation est allongé.

1.4 Maladie de Parkinson

En 1817, James Parkinson fut le premier médecin à publier un article sur la *MP* [80], dans lequel il décrit très clairement et avec beaucoup de détails les symptômes de six patients atteints de la maladie. Actuellement, il n'existe que des traitements symptomatiques, car la cause et l'évolution de la MP restent inconnues.

1.4.1 Symptômes

La MP se caractérise par des ***S**ymptômes **M**oteurs (SM)*, mais aussi des ***S**ymptômes **N**on **M**oteurs (SNM)* qui ne sont pas souvent évoqués [32]. Les symptômes moteurs se manifestent par une bradykinésie (ou hypokinésie), i.e. les mouvements du patient sont considérablement réduits et ralentis, des difficultés à déclencher des mouvements volontaires, une rigidité au niveau des muscles et des tremblements au niveau des mains au repos. D'autres symptômes sont des difficultés à parler ou à respirer [57]. Les symptômes non moteurs, récemment reconnues comme des symptômes de la MP, sont, entre autres, la dépression, les troubles du sommeil ou encore les anomalies sensorielles [61]. Tous les patients sont affectés par les SNM, qui augmentent lorsque la maladie progresse. Ces symptômes sont très rarement traités par les médecins, ce qui conduit à un handicap et à une mauvais qualité de vie, à longue durée pour les patients [24]. Les causes des SNM pourraient être liées à des pathologies en dehors des GB que nous n'allons pas aborder dans ce manuscrit.

1.4.2 Origines physiologiques

Cette maladie *neurodégénérative* se particularise par la destruction d'une population de neurones ; les neurones à dopamine de la SN. Les GB sont fortement impactés par cette destruction, i.e. que, dans certaines parties des GB, les émissions de PA sont altérées et des fortes synchronisations pathologiques sont observées. La perte dopaminergique dans le STR mène à un déséquilibre de l'activité des PA dans les neurones du STR et plus particulièrement dans les voies directe et indirecte des GB. Des études opto-génétiques ont montré que l'hyperactivité des neurones du STR de la voie indirecte mène à un état parkinsonien chez les rongeurs [60].

Dans un premier temps, les synchronisations pathologiques ont été associées au tremblement, mais il semble qu'elles seraient associées à l'hyperactivité ou/et la rigidité. Les circuits, les plus propices à une synchronisation pathologique, sont le circuit du STN avec le GPe [16, 82] et le circuit du GPeP avec le GPeA [69, 70]. Les connexions synaptiques entre ces populations de neurones sont renforcés à l'état parkinsonien. De la Crompe et al. [27] suggère que l'inhibition du STR induit une activité anormale dans les GP et que celle-ci augmenterait potentiellement les synchronisations pathologiques. Cette inhibition est due, soit à un renforcement synaptique entre les neurones du STR, soit à un changement de comportements des canaux ioniques, plus précisément les canaux K^+ [65] ou/et Ca^{2+} [15, 22, 34].

1.4.3 Traitements symptomatiques

De nos jours, il existe deux types de traitements ; les *traitements médicamenteux* et le *traitement chirurgical*. Depuis la découverte de la MP, le médicament le plus efficace et le plus utilisé, pour soulager les SM, est la L-dopa, ou levodopa. Tout au long de la maladie, la L-dopa remédie au manque de dopamine en simulant l'action de la dopamine [74]. Cette dose est augmentée au fil de l'évolution de la maladie. Les agonistes dopaminergiques complètent ce traitement en réduisant l'effet de certains SNM tels que la dépression. Dans certains cas, les traitements dopaminergiques provoquent des effets secondaires, comme par exemple des nausées, des vomissements, de la dyskinésie, des troubles du comportement qui peuvent aller jusqu'à l'addiction aux jeux ou autres.

Lorsqu'un patient réagit sensiblement aux traitements dopaminergiques, alors la chirurgie est conseillée, mais seulement 5 à 10% des patients subissent cette chirurgie lourde. Plusieurs critères doivent être respectés avant de tenter cette opération (voir le site de l'association France Parkinson pour plus d'informations [3]). La chirurgie consiste en une *stimulation cérébrale profonde*, en anglais ***D**eep **B**rain **S**timulation*, (DBS) dans laquelle une (ou deux) électrode(s) sont implantée(s) dans le cerveau et plus précisément dans le STN [32].

Les deux types de traitements, la L-dopa et la DBS, peuvent être superposés, afin de garantir des résultats plus satisfaisants [74]. Ces traitements, médicamenteux et chirurgical, ont pour but de traiter des SM, mais ont très peu d'effet sur les SNM. Des rééducations, physique et orthophonique, sont proposées pour compléter les traitements de la MP, afin d'améliorer la qualité de vie des patients, cependant l'évolution de la maladie n'est pas stoppée.

Stimulation cérébrale profonde (ou «deep brain stimulation »)

Depuis 30 ans, i.e. lors de la premier chirurgie, DBS, à Grenoble, 100 000 patients ont pu bénéficier de cette chirurgie pour des problèmes neurologiques ou neuropsychiatriques. Les effets de la stimulation sur le cerveau sont encore mal compris. Les chercheurs ne savent pas si c'est la suppression ou l'activation des éléments neuronaux locaux qui interrompent ou modulent le flux d'informations au sein du réseau ou améliore le rapport signal-bruit dans ce système stochastique.

Les électrodes sont souvent implantées dans le STN ou le GPi où elles envoient des impulsions électriques. Un neurostimulateur est implanté sous la peau afin de se connecter à l'électrode et de générer du courant électrique. Un neuromodulateur permet de gérer les paramètres de stimulation de l'extérieur [81].

La stimulation bilatérale du STN améliore efficacement les fluctuations motrices, l'hypokinésie, les tremblements et réduit la prise de médicaments dopaminergiques. Par contre, l'implantation des électrodes dans le GPi a des effets moins importants sur les tremblements [74]. Des effets secondaires, comme des troubles cognitifs, de l'humeur et du comportement, ont été observés et plus particulièrement sur les électrodes implantées dans le STN que celle GPi. En cause, la petite taille du STN qui amène à ce que le courant se propage involontairement sur des structures adjacentes. Dans certains cas, l'implantation de l'électrode côté gauche du STN mène à une dépression du patient et celle de côté droit produit un rire hilarant. Des effets secondaires de la stimulation peuvent être évités en adaptant la stimulation au besoin du patient, ce qui est appelé une DBS en boucle fermée [93].

1.4.4 Problématiques

Quel est le risque de développer la MP ? Les causes de développer la MP sont toujours inconnues. Des nombreuses hypothèses existent comme l'hérédité et l'environnement. Certains cas d'hérédité ont été constatés, mais elles ne concernent que quelques familles. L'exposition importante et prolongée, de plusieurs années, à des produits chimiques du type pesticide, ou certains solvants peuvent entraîner le déclenchement de la MP, mais ne sont pas la seule cause à la maladie. De nos jours, l'hypothèse la plus probable, et qui touche la plupart des patients, est la combinaison de ces deux facteurs, l'environnement et l'héritage.

La cause et les mécanismes des oscillations pathologiques demeurent aussi une inconnue, plusieurs pistes sont étudiées. Est-ce que les MSN-D1 et MSN-D2 sont à l'origine de la génération et la propagation des oscillations pathologiques ? Est-ce que la connectivité GPeA-STR est renforcée en condition pathologiques, créant les mécanismes de «freezing », observée dans la MP chez le rat [10]. L'hypoactivité des MSN-D1 et l'hyperactivité des MSN-D2, sont-elles provoquées par le dysfonctionnement des canaux ioniques calciques [15] ou plutôt par les modifications synpatiques GABAergiques ?

Les traitements pourraient être améliorés afin de diminuer les effets secondaires. Adapter les stimulations dans la DBS en fonction du besoin du patient afin de réduire les effets secondaires, i.e. un système en boucle fermée. Dans le cas où l'origine des oscillations soit dans le circuit GPe-STR ou GPeA-GPeP, ils pourraient penser à stimuler d'autres structures plus grandes, comme le GPeP ou le STR, afin d'éviter la propagation du courant dans d'autres structures non concernées par la maladie.

1.5 Modèles existants des GB dans la littérature

Dans la littérature, plusieurs modèles des GB ont été proposés, allant d'une complexité plus faible avec une équation différentielle par population de neurones, le modèle à population [77, 87], à un modèle plus complexe avec au minimum quatre équations différentielles par neurone, le modèle d'Hodgkin-Huxley [51] et en passant par les modèles intègre-et-tire avec une, ou deux, équations différentielles par neurones [62]. Chaque modèle des GB est différent. Les uns [25, 67, 90]

ne considèrent qu'une seule population de neurones inhibiteurs dans le GPe et ne font pas la distinction entre les neurones du GPeA et du GPeP et d'autres [64, 77] les ont pris en compte, cependant ces modèles ne modélisent pas les canaux ioniques.

De plus certains modèles ont été validés avec une centaine de neurones [26, 72, 90]. Ces modèles ne respectent ni la proportion de neurones ni les connexions synaptiques des ganglions de la base et d'autres modèles vont jusqu'à simuler l'échelle une du rat avec plus d'un million de neurones [38]. Cependant le modèle [38] ne prends pas en compte les canaux ioniques. Effectivement, et comme Eliasmith et Trujillo le disent dans leur article [36], la simulation à taille réelle est nécessaire afin de capturer, le plus possible, les aspects pertinents du système dans le but de comprendre la relation entre le cerveau et le comportement d'un être humain, un objectif fondamental des neurosciences.

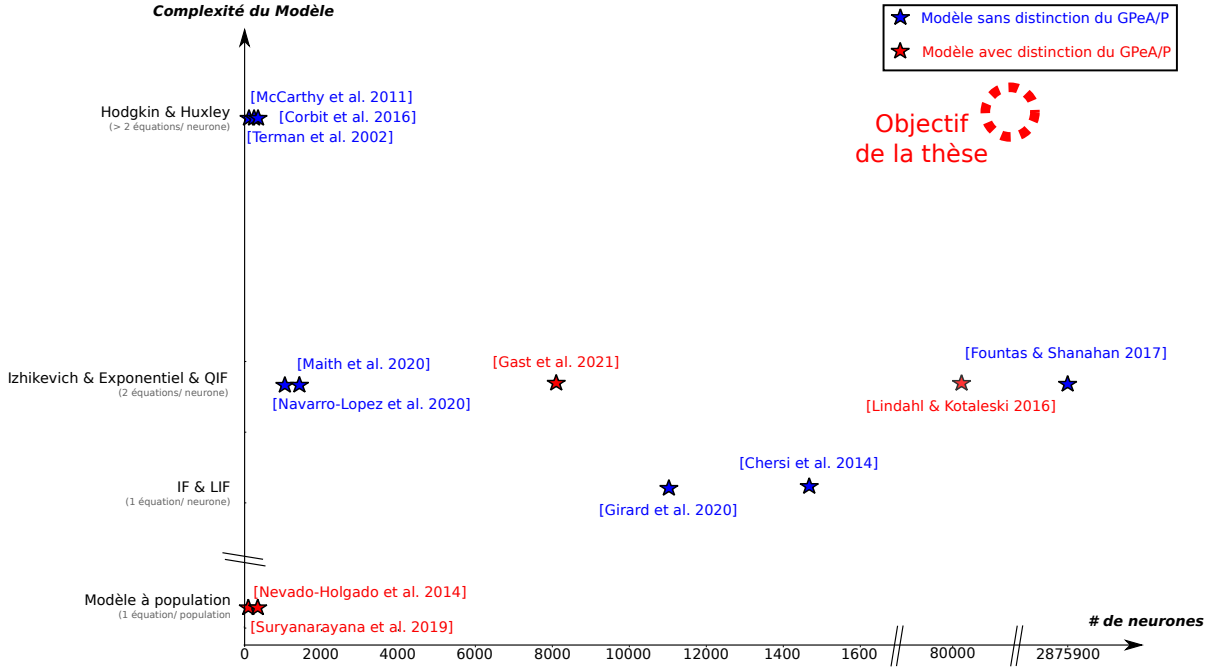

Fig. 1.11 – Représentation graphique des différents modèles des GB.

Dans la Figure 1.11, nous comparons des modèles des GB en évaluant leur complexité en fonction du nombre de neurones. Cette liste de modèles n'est pas exhaustive, mais donne une vue globale des modèles existant des GB. Les modèles en bleu ne font pas de distinction entre les GPeA et GPeP, tandis que ceux en rouge le font.

Le modèle à population modélise chaque structure par une seule équation. Nevado-Halgado et al. [77] ont modélisé les structures du STN, du GPeA, du GPeP, mais aussi le STR, le cortex et le thalamus. Dans cette article, l'hypothèse des auteurs est que la boucle STN-GPe intensifie les oscillations pathologiques dans la MP.

Quant au modèle de Suryanarayana et al. [87], les auteurs ont modélisé les GB au complet, i.e. le STR, le STN, le GPe, le GPi et aussi la SN. Dans ce modèle, les auteurs ont également fait la distinction entre GPeA et GPeP, mais aussi celle entre les neurones du MSN-D1 et ceux du MSN-D2. Les auteurs ont testé plusieurs hypothèses sur l'origine des oscillations pathologiques de la maladie. Leur conclusion est que le circuit GPe-STN-GPi/SN génère des oscillations pathologiques et plus particulièrement le circuit GPeA-STR. Cependant, et de leur

point de vue, ces résultats devront être approfondis.

Malheureusement, avec ce type de modèle, les auteurs ne peuvent tester aucune hypothèse au niveau intrinsèque d'un neurone. De plus, une population de neurones est décrite par une unique équation. Ce type de modèle est très basique, mais idéal pour tester des hypothèses au niveau du renforcement synaptique lors de la MP.

Les modèles linéaires IF et LIF, expliqués dans la sous-section 1.1.2, modélisent un neurone par une équation. Dans le modèle des GB chez le singe de Girard et al. [43], les auteurs ont modélisé le MSN, le FSN, le STN, le GPe et aussi le GPi/SN. Au total, le modèle a environ 11 000 neurones. D'après les auteurs, les oscillations pathologiques sont générées au niveau de la boucle STN-GPe.

En ce qui concerne le modèle de Chersi et al. [25], les auteurs ont modélisé le STR, le STN, le GPe, le SN, le thalamus et aussi le cortex moteur. De plus, les auteurs ont fait la distinction entre les neurones MSN-D1 et MSN-D2. Ce modèle a été validé avec environ 14600 neurones. Le but de cet article était d'améliorer un modèle existant en ajoutant de la plasticité cortico-striatale au niveau neuronal. Aucune hypothèse n'a été émise au niveau des oscillations pathologiques de la MP.

Malheureusement, avec ce type de modèle, les auteurs ne peuvent tester aucune hypothèse au niveau intrinsèque d'un neurone. Cependant, les connexions synaptiques peuvent être étudiées plus en détails.

Dans les modèles non linéaires d'Izhikevich, d'exponentiel ou encore IF quadratique, un neurone est représenté par deux équations différentielles. Le modèle de Navarro-Lopez et al. [76], simulant environ 1000 neurones, considère les neurones du STR, en distinguant les MSN-D1 et les MSN-D2, du GPe, du GPi, du STN, du cortex et aussi du thalamus. Leurs hypothèses portent sur la réduction de la dopamine dans les neurones du STR. Cependant, d'après les auteurs, le dysfonctionnement des GB n'est pas uniquement lié à la perte en dopamine, mais probablement aussi lié à des boucle propice à la synchronisation pathologique dans la voie indirecte des GB.

Le modèle de Maith et al. [67] a modélisé les mêmes structures que l'article [76] avec cette fois-ci environ 2000 neurones. Les auteurs ont comparé leur modèle avec des données de patients sains et parkinsoniens. Ainsi, les auteurs ont renforcé des connexions synaptiques entre certaines structures, comme le cortex et le MSN-D2, le cortex et le STN ou encore le STN et le GPe. Les auteurs ont réussi à adapter leur modèle à des données réelles.

En ce qui concerne le modèle de Lindahl & Kotaleski [64], les auteurs ont modélisé le STR en distinguant les MSN-D1 et les MSN-D2, le FSN, le STN, le GPe en distinguant aussi le GPeA et le GPeP et pour finir le SN. Sur ce modèle, les auteurs testent plusieurs hypothèses différentes sur le renforcement synaptique, comme par exemple la boucle STN-GPe et la boucle GPe-MSN. Dans l'article, ils s'interrogent sur la possibilité que la boucle MSN-MSN génère des oscillations pathologiques lors de la MP.

Le modèle, qui a simulé les GB à l'échelle une du rat, est celui de Fountas & Shanahan[38] avec environ 2,9 millions de neurones. Leur modèle est composé de MSN-D1, de MSN-D2, de FSN, de STN, de GPe et de SN. Les auteurs émettent plusieurs hypothèses sur l'origine des oscillations pathologiques dans la MP, comme la boucle STN-GPe, le STR ou sinon dans le cortex cérébral qui sont propices aux oscillations pathologiques.

Malheureusement, avec ce type de modèle, les auteurs ne peuvent tester aucune hypothèse concernant les dynamiques ioniques. L'avantage de ces modèles «moins complexes» est qu'il est

possible de modéliser toutes les structures des GB sans pour autant avoir des temps de simulation irréalistes.

Le modèle d'Hodgkin & Huxley, expliqué dans la sous-section 1.1.2, est le modèle le plus complexe de la Fig. 1.11 avec un minimum de quatre équations différentielles par neurone. Le modèle Terman et al. [90] considère les neurones du GPe, avec 6 canaux ioniques et 4 équations différentielles, et les neurones du STN, avec 6 canaux ioniques aussi et 5 équations différentielles. Les auteurs ont simulé entre 8 et 20 neurones de chaque type. Dans cet article, les auteurs testent la génération des oscillations pathologiques au sein de la boucle STN-GPe.

Quant au modèle de Corbit et al. [26], les auteurs ont modélisé 8 neurones du GPe, 40 neurones du MSN et 8 neurones du FSN. Les GPe sont décrit par 10 canaux ioniques, et 16 équations différentielles, les MSN par 9 canaux ioniques, et 11 équations différentielles, et les FSN par 4 canaux ioniques, et 5 équations différentielles. Dans cet article, l'hypothèse principale sur l'origine des oscillations pathologiques de la MP est sur la boucle MSN-FSN-GPe. Concluant que le GPe joue un rôle principal dans la génération des oscillations pathologiques dans plusieurs noyaux des GB.

McCarthy et al. [72] ont travaillé sur un réseau de 100 neurones du MSN, chaque neurone composé de 4 canaux ioniques et de 5 équations différentielles, avec comme seule connexion MSN-MSN. Dans cet article, deux hypothèses principales sur l'origine des oscillations pathologiques sont analysées, celle sur la boucle MSN-MSN et celle sur l'influence du courant *M*.

Cependant les auteurs de ces différents articles [26, 72, 90] ont fait le choix de valider leur modèle de réseau avec environ une centaine de neurones. Malheureusement, leur modèle ne respecte ni la proportion de neurones ni les connexions synaptiques des ganglions de la base. De plus, les auteurs n'ont pas fait la distinction entre les GPeA et les GPeP. Nous constatons aussi qu'aucun modèle ne simule toutes les structures des GB.

Les principales hypothèses sur l'origine des oscillations pathologiques, mentionnées dans les articles de la Fig. 1.11, portent sur les boucles STN-GPe [38, 43, 64, 77, 87, 90], GPe-STR [26, 87] et aussi MSN-MSN [38, 64, 72]. La boucle STN-GPe est l'hypothèse, sur le renforcement synaptique lors de la MP, la plus mentionnée, la plus connue, mais aussi la première hypothèse à être citée dans les articles sur l'étude des tremblements dans la MP [21]. Néanmoins la boucle STN-GPe n'est pas l'unique raison des oscillations pathologiques dans la MP [72]. Et donc des nouvelles hypothèses ont fait surface comme celle de la boucle GPe-STR [44] ou encore celle de la boucle MSN-MSN [72]. Plus de détails sur l'origine de la maladie seront donnés dans la prochaine section 1.4. Toutefois la seule hypothèse au niveau intrinsèque testée sur un modèle mathématique est celle de McCarthy et al. [72]. Les auteurs ont testé l'influence du courant *M* des neurones du MSN sur la MP.

L'objectif de la thèse est de tester toutes ces hypothèses sur un même modèle, or le formalisme d'HH est le seul modèle avec lequel nous puissions tester toutes ces hypothèses sans exception :

- Boucles propices à la synchronisation pathologiques :
  - ⋆ STN-GPeP
  - ⋆ GPeP-(MSN-D2)
  - ⋆ GPeA-(MSN-D2)
  - ⋆ (MSN-D2)-(MSN-D2)
- Influence de la voie indirecte sur les synchronisations pathologiques

- Canaux ioniques influant sur les synchronisations pathologiques :
  - ⋆ Courant M dans les MSN
  - ⋆ Canaux Ca^{2+}
  - ⋆ Canaux K^{+}

De plus, nous voulons simuler les GB à l'échelle une du rat à la différence des modèles d'HH cités dans la Fig. 1.11. A contrario du modèle d'Izhikevich [38], seul modèle cité dans la Fig. 1.11 simulant l'échelle une du rat, nous nous appuyons sur le modèle d'HH afin de tester des hypothèses intrinsèques aux neurones. Par ailleurs, nous faisons la distinction entre les neurones du GPeA et du GPeP.

1.6 Simulation du réseau de neurones

1.6.1 Méthodes numériques

Le potentiel membranaire est déterminé par l'équation (1.2), toutefois il faut résoudre le système d'*Équations Différentielles Ordinaires (EDO)* (1.4). Cependant ce système est non linéaire et stochastique, puisque du bruit est ajouté au courant d'entrée. Lors d'un potentiel d'action, des variables, comme la variable m, évoluent rapidement et d'autres plus lentement, e.g. h. Ce système est alors aussi appelé un *système lent-rapide*. De plus, le système est lent au moment du potentiel de repos et rapide lors d'une émission de potentiel d'action. Ainsi, le potentiel, comme aussi les «gating variables», ne peuvent pas être résolus de manière ordinaire et doivent être approchés au moyen de *méthodes numériques*, comme e.g. les méthodes d'Euler ou de Runge-Kutta. Les méthodes numériques consistent à subdiviser l'intervalle de temps t en plusieurs morceaux de taille Δt, Δt appelé le *pas de temps*.

Afin de mieux comprendre les méthodes numériques, nous allons prendre un cas général des EDO [28, 30]. Définissons I_0 un intervalle de $\mathbb{R}$ non réduit à un point et t_0 un point dans I_0. La fonction f est définie et continue sur $I_0 \times \mathbb{R}^m$, un élément y_0 est dans $\mathbb{R}^m$ et la fonction y cherchée est continue et dérivable sur l'intervalle I_0, à valeurs dans $\mathbb{R}^m$, telle que

$$\forall t \in I_0, \qquad y'(t) = f(t, y(t)) \quad (1.10) \qquad \text{et} \qquad y(t_0) = y_0 \quad (1.11)$$

Ce problème s'appelle un *problème de Cauchy* pour le système différentiel (1.10) et (1.11) est la *condition de Cauchy* ou aussi la *condition initiale*. La variable t représente le temps et t_0 est l'*instant initial*. L'équation (1.10) est un système de m équations différentielles dans $\mathbb{R}$ à m inconnues ; notons $y_1, y_2, ..., y_m$ les composantes de y et $f_1(t, y_1, ..., y_m), ..., f_m(t, y_1, ..., y_m)$ les composantes de $f(t, y)$. Les équations peuvent alors être réécrites sous le système d'équations suivant :

$$\begin{cases} y_1'(t) &= f_1(t, y_1(t), ..., y_m(t)) \\ y_2'(t) &= f_2(t, y_1(t), ..., y_m(t)) \\ \vdots & \\ y_m'(t) &= f_m(t, y_1(t), ..., y_m(t)) \end{cases} \quad (1.12)$$

Prenons maintenant notre système d'équations de HH (1.4) et comparons le avec le système d'équations (1.12).

$$\left\{\begin{array}{lcl} y_1'(t) &=& f_1(t, y_1(t), ..., y_m(t)) \\ y_2'(t) &=& f_2(t, y_1(t), ..., y_m(t)) \\ \vdots & & \\ y_m'(t) &=& f_m(t, y_1(t), ..., y_m(t)) \end{array}\right. \Rightarrow \left\{\begin{array}{lcl} \dfrac{dV}{dt} &=& \displaystyle\sum_{ion\in\{Na,K,L\}} \frac{I_{ion}(V, m(t), h(t), n(t))}{C_m} \\ \dfrac{dm}{dt} &=& \dfrac{m_\infty(V) - m(t)}{\tau_m(V)} \\ \dfrac{dh}{dt} &=& \dfrac{h_\infty(V) - h(t)}{\tau_h(V)} \\ \dfrac{dn}{dt} &=& \dfrac{n_\infty(V) - n(t)}{\tau_n(V)} \end{array}\right.$$

Les méthodes numériques sont écrites sous la forme :

$$y_{n+1} = y_n + \Delta t \phi(t_n, y_n) \qquad 0 \leqslant n \leqslant N$$

avec y_n la valeur approchée de la valeur $y(t_n)$ déduite de la solution exacte y, Δt le pas de temps et $t_n = t_0 + n\Delta t$. Nous aborderons la *méthode d'Euler* et de *Runge-Kutta* dans ce manuscrit.

La méthode d'Euler consiste à construire une solution approchée du système (1.4) ou (1.10). En remplaçant $\phi(t_n, y_n) = f(t_n, y_n)$, la méthode d'Euler est définie par ce système d'équations :

$$\left\{\begin{array}{lcl} t_{n+1} &=& t_n + \Delta t \\ y_{n+1} &=& y_n + \Delta t f(t_n, y_n) \end{array}\right. \tag{1.13}$$

L'*ordre d'une méthode numérique* a une influence importante sur la précision que cette méthode permet d'atteindre. Plus l'ordre est grand plus la précision est importante [28, 30]. La méthode d'Euler est une méthode d'ordre 1, c'est-à-dire que l'erreur d'approximation de cette méthode est d'ordre Δt. Cette notion est très importante pour la suite.

La méthode de Runge-Kutta consiste à calculer par récurrence les points (t_n, y_n) en utilisant des points intermédiaires $(t_{n,i}, y_{n,i})$ avec $t_{n,i} = t_n + c_i\Delta t$ où $1 \leqslant i \leqslant q$ avec $q \in \mathbb{N}^*$ et $c_i \in [0, 1]$. La méthode de Runge-Kutta est définie par ce système :

$$\left\{\begin{array}{lcll} t_{n,i} &=& t_n + c_i\Delta t & 1 \leqslant i \leqslant q \\ y_{n,i} &=& y_n + \Delta t \displaystyle\sum_{1\leqslant j\leqslant i} a_{ij} f(t_{n,j}, y_{n,j}) & \\ t_{n+1} &=& t_n + \Delta t & \\ y_{n+1} &=& y_n + \Delta t \displaystyle\sum_{1\leqslant j\leqslant i} b_j f(t_{n,j}, y_{n,j}) & \end{array}\right. \tag{1.14}$$

Une méthode de Runge-Kutta est toujours définie, si les coefficients q et a_{ij}, b_j et c_i sont connus. Il est conventionnel de définir les coefficients comme suit :

Nous n'allons pas donner plus de détails sur ces coefficients, vous pouvez consulter [28] et [30] pour plus d'informations. Dans les prochains paragraphes, quelques exemples concrets de la méthode vont être donnés.

(M_1)	c_1	0	0	...	0	0
(M_2)	c_2	a_{21}	0	...	0	0
	⋮	⋮	⋮	⋱	⋮	⋮
	⋮	⋮	⋮		0	0
(M_q)	c_q	a_{q1}	0	...	a_{qq-1}	0
(M)		b_1	b_2	...	b_{q-1}	b_q

Tab. 1.2 – Coefficients définissant la méthode de Runge-Kutta.

Runge-Kutta d'ordre 1, pour $q = 1$, le système s'écrit ainsi, d'après le Tab. 1.2, $c_1 = 0$, $a_{11} = 0$ et $b_1 = 1$ est

$$\begin{cases} t_{n,1} &= t_n \\ y_{n,1} &= y_n \\ t_{n+1} &= t_n + \Delta t \\ y_{n+1} &= y_n + \Delta t f(t_n, y_n) \end{cases}$$

La méthode d'Euler est un cas spécial de la méthode de Runge-Kutta à l'ordre 1.

Runge-Kutta d'ordre 2, pour $q = 2$ et d'après le Tab. 1.2, nous avons

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \alpha & \alpha & 0 \\ \hline & 1 - \dfrac{1}{2\alpha} & \dfrac{1}{2\alpha} \end{array} \qquad \text{où} \qquad \alpha \in]0, 1]$$

Le système est alors

$$\begin{cases} t_{n,1} &= t_n \\ y_{n,1} &= y_n \\ t_{n,2} &= t_n + \alpha\Delta t \\ y_{n,2} &= y_n + \alpha\Delta t f(t_n, y_n) \\ t_{n+1} &= t_n + \Delta t \\ y_{n+1} &= y_n + \Delta t \left(\left(1 - \dfrac{1}{2\alpha} \right) f(t_n, y_n) + \dfrac{1}{2\alpha} f(t_{n,2}, y_{n,2}) \right) \end{cases} \tag{1.15}$$

Si $\alpha = \dfrac{1}{2}$, la méthode est appelée la méthode du point milieu et la dernière équation du système (1.15) sera écrite :

$$y_{n+1} = y_n + \Delta t f(t_{n,2}, y_{n,2})$$

À partir de maintenant, la méthode de Runge-Kutta au point milieu sera appelée la méthode de **R**unge-**K**utta **2** (RK2). Cette méthode est une méthode d'ordre 2, donc l'erreur d'approximation est d'ordre Δt^2 ($\Delta t \ll 1$). Par conséquent, RK2 est une méthode qui approche mieux la solution cherchée que la méthode d'Euler.

1.6.2 Principaux simulateurs de réseaux de neurones

En neurosciences computationnelles, les modèles sont utilisés pour décrire le comportement des neurones ou réseaux de neurones dans le cerveau à l'aide de simulateurs, comme **BRIAN**,

NEURON ou **SiReNe**. Ces simulateurs servent à tester des hypothèses dans le but de limiter l'expérimentation animale. D'ailleurs, pendant une audition publique à l'Assemblée Nationale, Cédric Villani, député, et Florence Lassarde, sénatrice, ont débattu de ce sujet avec l'Office parlementaire présent [11], en proposant des méthodes alternatives, comme la modélisation et la simulation. Au fil du temps, plusieurs simulateurs ont été développés avec des objectifs et des fonctionnalités différentes de leurs concurrents [17, 18]. Le Tab. 1.3 compare les simulateurs les plus utilisés en neurosciences computationnelles.

NEURON est particulièrement adapté pour simuler des modèles de neurones à câbles avec des anatomies ramifiées complexes. L'objectif est de modéliser des propriétés biophysiques telles que les multiples types de canaux, la distribution inhomogène des canaux, l'accumulation, la diffusion ioniques et les synapses puissent être modélisés de manière indépendante. L'idée principale de ce simulateur est de faire de la modélisation sans que l'utilisateur ait à s'occuper des méthodes de résolution, des processeurs, de la disposition de la mémoire ou encore des détails de l'interface de *langage de programmation* ***C*** *(C)* de **NEURON**.

NEural Simulation Tool (NEST) est principalement conçu pour simuler la dynamique, la taille et la structure des systèmes neuronaux avec une connectivité biologiquement réaliste.

Dans BRIAN, les neurones et les synapses sont définis par des équations en notation mathématique standard. Une petite particularité de ce logiciel est que les dimensions physiques de chaque variable doivent être définies, ainsi, lors de la compilation, la cohérence entre les dimensions est vérifiée par le simulateur.

GPU enhanced Neuronal Network (GeNN) est l'un des rares environnements de simulation de réseaux neuronaux améliorés par le *processeur graphique*, en anglais ***G****raphics* ***P****rocessing* ***U****nit*, (GPU) et qui repose sur la bibliothèque ***C****ompute* ***U****nified* ***D****evice* ***A****rchitecture (CUDA).*

SiReNe a un schéma de simulation hybride, qui combine la résolution des équations du modèle HH à chaque pas de temps avec la mise à jour des courants synaptiques à chaque événement. De plus, la connectivité du réseau n'est pas stockée en mémoire, mais est générée à chaque événement, et plus précisément à chaque PA.

1.6.3 Problématiques

Les équations de HH sont des équations non-linéaires avec un système linéaire lent-rapide. La précision de la méthode numérique doit être adaptée à ce type de système, avec à la fois une méthode numérique

- **assez précise**, pour qu'elle puisse suivre les changements lent-rapide du système, et
- **assez rapide**, afin d'avoir un temps de simulation performant sans perte de précision.

Ensuite, il faut trouver un pas de temps approprié qui ne soit pas trop grand afin de suivre les changements lent-rapide du système, ni trop petit afin d'avoir un temps de simulation raisonnable [73].

Par ailleurs, le pas de temps doit être adapté à la constante de temps, i.e. elle doit être plus petite, ou égale, à la plus petite constante de temps du système d'équations (1.4). En effet, si le pas de temps est plus grand que la constante de temps, alors les PA ne seront pas tous détectés.Toutes ces problématiques vont être abordées dans la suite du manuscrit.

Les principales difficultés de simuler un grand réseau de neurones sont :

Logiciels	**NEURON** [50]	**NEST** [33]	**BRIAN** [46, 47]	**GeNN** [59, 78]	**SiReNe** [13]
Modèles	HH/ IF	HH/ IF	HH/ IF	HH/ IF	HH/ IF
Structure du neurone	multicomp.* / à point	à point	multicomp. / à point	à point	à point
Plasticité	oui	oui	oui	oui	non
Schéma de simulation	événementiel /à pas de temps	événementiel /à pas de temps	à pas de temps	à pas de temps	événementiel /à pas de temps
Interpolation**	oui	oui	non	non	oui
Stockage de connectivité	oui	oui	oui	oui	non
Langage de programmation	C/ C++ / Fortran	C++/ Python / Cython	Python/ C	C++/ Python / CUDA	C
Parallélisation du code	OpenMP/ MPI	OpenMP	OpenMP	GPU	OpenMP/ MPI
Interface graphique	oui	non	non	non	non
En développement	oui	oui	oui	oui	oui
Codes disponibles sur le web	oui	oui	oui	oui	oui

Tab. 1.3 – Comparaison des principaux simulateurs de réseau de neurones. *Neurones multicompartimentaux. ** Interpolation du temps du PA.

- le temps de simulation,
- le stockage en mémoire (principalement causé par la connectivité),
- dans le cas de HH, la précision de la méthode numérique, et
- la précision de détection du temps du PA.

Le simulateur doit être à la fois rapide, mais aussi assez précis pour pouvoir simuler des comportements biophysiques correctement. Ainsi, une solution est de générer les connexions entre les neurones à chaque événement, de faire tourner les simulations sur plusieurs processeurs, voire même sur plusieurs machines, et de calculer le plus précisément possible le temps du PA avec, e.g., une intégration à l'aide de courbes de Béziers, par exemple.

1.7 Conclusion du chapitre

Ce chapitre nous a permis de faire l'état de l'art sur les modèles neuronaux existants faisant le point sur leurs forces, et faiblesses, et sur les méthodes de simulation, pas toujours adaptées à des réseaux de grande taille.

Les méthodes de simulations actuelles stockent habituellement toute la connectivité, ainsi environ 1 terabyte de mémoire est utilisé afin de stocker le schéma de connectivité des ganglions de la base chez le rat. La mémoire peut vite saturer lors d'une simulation. Dans le chapitre 2, une nouvelle approche va être présentée avec laquelle le temps de simulation, ainsi que la mémoire vont être optimisés. Cette approche est implémentée dans le logiciel SiReNe présenté dans le chapitre 3.

Les neurones sont modélisés par le formalisme d'HH, au lieu du modèle LIF, de sorte à décrire la dynamique du neurone et de pouvoir ainsi tester des hypothèses sur le dysfonctionnement des canaux ioniques, constaté dans la MP. Nous avons pu constater que les modèles connus [26, 76, 90] des GB ne sont pas complets, puisque aucun ne fait la distinctions entre les neurones GPeA et les GPeP. Ces deux types de neurones vont être présentés dans le chapitre 4.

Une fois que le modèle et que les méthode de simulations sont en place, la simulation des ganglions de la base est possible, voir le chapitre 5.

2

Une nouvelle approche pour la simulation de réseaux de neurones de très grande taille sans stockage de connectivité

Sommaire

Nous nous sommes appuyés sur le formalisme d'HH, afin de modéliser les différents canaux ioniques des neurones impliqués dans les GB affectés lors de la MP. Ce choix nous permet de simuler un réseau de neurones proche de la dynamique neuronale biologique. Toutefois, comme cité dans le chapitre 1, d'un côté, la mémoire augmente exponentiellement lorsqu'un réseau de neurones de taille réelle est simulé, et ceci principalement à cause de la connectivité, et de l'autre, le temps de simulation est conséquent, compte tenu du choix du formalisme de HH. Le choix de la méthode numérique est aussi important afin de suivre le système d'équations, lent-rapide, de HH. De plus, la précision du calcul du temps du PA est essentielle de manière à bien analyser les synchronisations pathologiques, caractéristiques de la MP.

Il n'existe pas de solution analytique pour les EDO du système de HH (1.4), non linéaire, par conséquent nous avons fait le choix d'approcher la solution par un calcul numérique et plus

précisément par la méthode numérique de RK2 présentée dans la sous-section 1.6.1. RK2 est une méthode d'ordre 2, i.e. l'erreur d'approximation est d'ordre Δt^2 ($\Delta t \ll 1$), permettant d'atteindre avec plus de précision la solution analytique qu'une méthode d'ordre 1, comme la méthode d'Euler. D'après Hansel et al. [48], la méthode numérique et le calcul du temps du PA doivent avoir le même ordre de précision afin que la méthode de simulation conserve le même ordre de précision que la méthode numérique, i.e. l'ordre 2. Deux méthodes calculant le temps du PA vont être proposées et comparées dans ce chapitre.

Un autre problème de simulation est l'allocation de mémoire, explosant lors de simulations à taille réelle de réseaux de neurones. La cause principale est le stockage de connectivité évoluant exponentiellement en fonction du nombre de neurones n, voir la Fig. 1.9. La solution, proposée dans ce chapitre, est d'éviter de stocker cette connectivité et de la générer lors d'un événement. Cette nouvelle approche de simulation de neurones HH est une méthode hybride dans laquelle la dynamique neuronale, i.e. le système d'équations (1.4), est résolu à chaque pas de temps, et où la dynamique synaptique est actualisée à chaque nouvel événement, et plus précisément à chaque PA. Lorsqu'un neurone émet un PA, alors ses neurones post-synaptiques sont générés et leurs courants synaptiques sont actualisés. Ainsi, nous évitons de stocker la connectivité complète du réseau et ne calculons pas le courant synaptique du neurone en fonction des neurones pré-synaptiques à chaque pas de temps. Par conséquent, la consommation de mémoire est considérablement réduite tout en préservant, ou réduisant, le temps d'exécution et la précision des simulations, notamment les temps du PA de neurones du type HH.

Cette nouvelle approche a été publiée dans l'article de Azevedo et al. en 2020 [13]. Le modèle de neurone MSN de McCarthy et al. de l'article [72] a été utilisé afin de valider et de comparer notre approche avec les approches existantes, voir l'annexe A.

La Fig. 2.1 représente le schéma de simulation adopté lors de nos simulations et montre le déroulement de ce chapitre. La première section porte sur la méthode hybride, à pas de temps et événementielle. La deuxième section porte sur les différentes manières d'estimer le temps du PA. Ensuite viennent la section sur la méthode générant la connectivité pseudo-aléatoirement et la conclusion du chapitre.

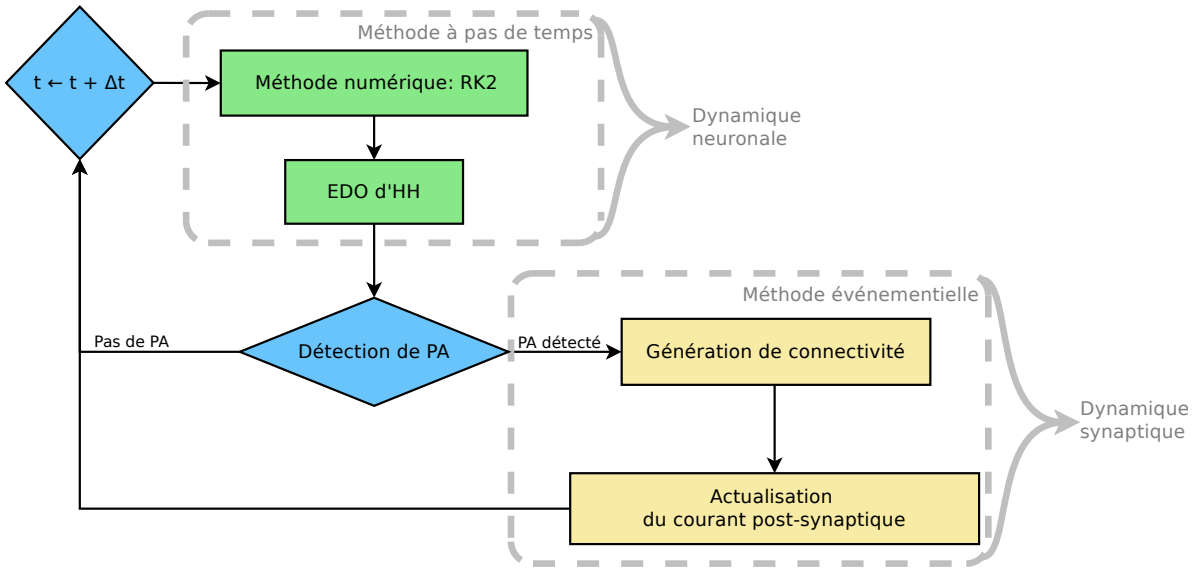

Fig. 2.1 – Représentation schématique de nos simulations, inspirée de l'article [13].

2.1 Approche hybride de simulation

L'approche hybride consiste en une méthode à pas de temps pour la mise à jour des variables d'état du neurone, et une méthode événementielle pour la mise à jour du courant synaptique, schématisée dans la Fig. 2.1.

2.1.1 Intégration de la dynamique neuronale : Méthode à pas de temps

L'approche à pas de temps se base sur l'intégration (ou l'actualisation) des variables d'état des neurones (V,m,n,h) du système (1.4) à chaque pas de temps. Les EDO du modèle de HH sont résolues avec une méthode de discrétisation temporelle itérative explicite. La méthode numérique RK2, vu à la sous-section 1.6.1, approche avec précision les solutions des EDO du modèle de HH. Cette étape est schématisée par la Fig. (2.1) et plus précisément à l'étape «méthode à pas de temps ».

Le potentiel d'action a une dynamique très évolutive, i.e. il s'accroît fortement lorsque les canaux Na^+ s'ouvrent. Afin de bien approcher cette dynamique évolutive, nous pourrions choisir une méthode d'intégration numérique plus précise, i.e. d'ordre plus grand que 2, mais le temps de simulation augmenterait lui aussi. Par conséquent, il faut choisir un pas de temps qui permet de suivre cette dynamique. La Fig. 2.2 montre la trace d'un PA, du modèle de neurone MSN basé sur le modèle McCarthy et al. [72], avec plusieurs pas de temps différents. Lorsque le pas de temps est trop grand par rapport aux différentes constantes de temps du neurone, la solution approchée ne suit pas correctement la dynamique du neurone, voir la Fig. 2.2 pour $\Delta t = 0,05ms$. Par contre, si le pas de temps est trop petit, alors le temps de simulation s'allonge fortement. Plus le pas de temps est petit plus notre solution approche la solution escomptée. Il faut cependant trouver un juste milieu. Dans les prochaines simulations, nous utiliserons un pas de temps $\Delta t = 0,005ms$, mais $\Delta t = 0,01ms$ peut aussi être adopté, lors de simulations à grande échelle, e.g. plus de 1 million de neurones, afin de réduire le temps de simulation.

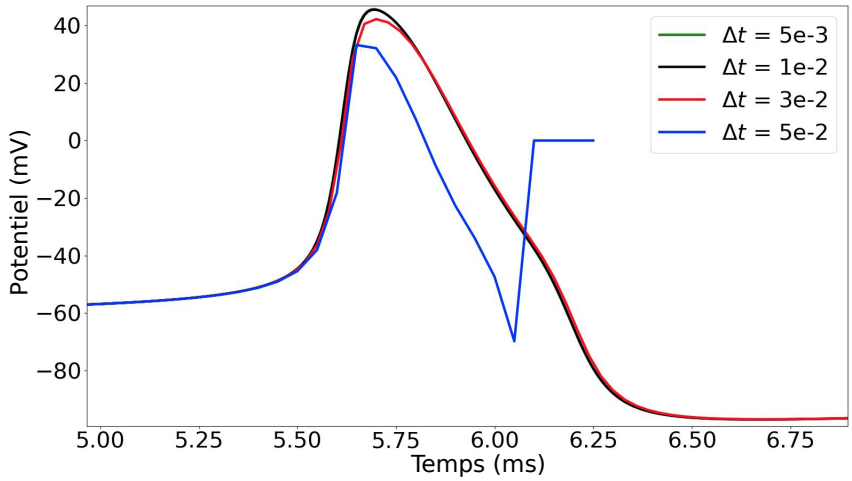

Fig. 2.2 – Trace du premier PA d'un neurone MSN avec différents pas de temps, $\Delta t = 0,05\,;\,0,03\,;\,0,01\,;\,0,005ms$, tirée de l'article [13]. La courbe $\Delta t = 5e-3$ est en dessous de celle $\Delta t = 1e-2$.

Avec l'approche classique, à chaque pas de temps, le courant synaptique du neurone k est calculé à l'aide de la somme des courants de chaque neurone pré-synaptique du neurone k, voir l'équation (1.8). Toutefois, il existe une autre approche moins coûteuse en temps et dans laquelle, il n'est pas nécessaire d'exploiter les courants pré-synaptiques des neurones à chaque pas

de temps. Dans cette nouvelle approche numérique, les courants synaptiques sont uniquement actualisés lors d'un PA, i.e. lorsqu'un neurone émet un PA, les courants des neurones post-synaptique sont actualisés. Ainsi, nous proposons un schéma de simulation hybride à pas de temps, pour la mise à jour des variables d'état du neurone, et événementiel, pour la mise à jour du courant synaptique.

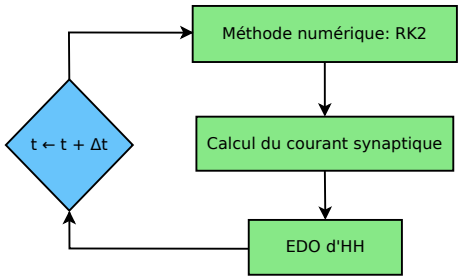

Fig. 2.3 – Représentation schématique de l'approche de simulation dite «classique».

2.1.2 Intégration de la dynamique synaptique : Méthode événementielle

Dans le schéma de simulation hybride, la méthode événementielle est appliquée au niveau des synapses. L'idée est d'être plus proactif, i.e. au lieu de calculer l'état des courants synaptiques, en fonction du neurone pré-synaptique, au temps $t + \Delta t$ en fonction du temps précédent t à chaque pas de temps, nous avons changé l'ordre du processus en le faisant à chaque nouvel événement. L'Algorithme 1 schématise les étapes du processus montrées dans la Fig. 2.1.

À chaque pas de temps l'état de chaque neurone est actualisé en fonction de son état interne et des courants d'entrée. Ensuite, il est nécessaire de détecter si le neurone a émis ou non un PA. À ce moment, si le neurone a émis un PA, alors les courants de neurones post-synaptiques au neurone émetteur sont mis à jour en fonction des paramètres du neurone émetteur.

Algorithme 1 : Schéma de simulation

```
pour chaque Pas de temps t faire
    Mise à jour des états des neurones en fonction de leur état interne et des courants
     d'entrée au temps t
    Détection des neurones qui ont émis un PA
    pour chaque neurone qui ont émis un PA faire
        Mise à jour de leurs courants post-synaptiques pour le temps t + Δt
    fin
fin
```

L'avantage de cette méthode est que seuls les neurones post-synatiques des neurones qui ont émis un PA actualisent leur courant synaptique, alors que dans la méthode classique, à pas de temps, tous les neurones sont systématiquement mis à jour à l'aide des neurones pré-synaptiques, même ceux qui n'ont pas émis de PA récemment. Ce schéma hybride permet de réduire significativement les calculs globaux lorsqu'une fraction de neurones émet au cours d'un même pas de temps. De plus, cette méthode peut-être combinée à une génération pseudo-aléatoire de la connectivité, réduisant ainsi le coût mémoire de la simulation.

Actualisation des courants synaptiques à chaque événement

Dans la sous-section 1.2.2, l'équation (1.8) définit le courant synaptique $I_{syn,k}$ d'un neurone k comme la somme des courants de tous les neurones pré-synaptiques j et respectivement de tous leurs PA. Posons, t_P le temps précédent et $t_C = t_P + \Delta t$ le temps courant. De plus, notons $Fact_{jk}\left(t_{sp_{n_j}}\right)$ la somme de tous les événements postsynaptiques opérés par le neurone pré-synaptique j sur le neurone k

$$\begin{aligned}
I_{syn,k}(t_C) &= \sum_j \overline{g}_{syn,jk}(V_k - E_{syn,jk}) \sum_{i=1}^{n_j} \exp\left(-\frac{t_C - t_{sp_i}}{\tau_{syn,jk}}\right) \\
&= \sum_j \overline{g}_{syn,jk}(V_k - E_{syn,jk}) \sum_{i=1}^{n_j} \exp\left(-\frac{t_C - t_P + t_P - t_{sp_i}}{\tau_{syn,jk}}\right) \\
&= \sum_j \overline{g}_{syn,jk}(V_k - E_{syn,jk}) \sum_{i=1}^{n_j} \exp\left(-\frac{t_C - t_P}{\tau_{syn,jk}}\right) \exp\left(-\frac{t_P - t_{sp_i}}{\tau_{syn,jk}}\right) \\
&= \sum_j \overline{g}_{syn,jk}(V_k - E_{syn,jk}) \underbrace{\exp\left(-\frac{\Delta t}{\tau_{syn,jk}}\right) \sum_{i=1}^{n_j} \exp\left(-\frac{t_P - t_{sp_i}}{\tau_{syn,jk}}\right)}_{Fact_{jk}\left(t_{sp_{n_j}}\right)}
\end{aligned} \tag{2.1}$$

A chaque pas de temps, le facteur $Fact_{jk}$ est mis à jour de la façon suivante :

$$Fact_{jk} \longleftarrow Fact_{jk} \times \exp\left(-\frac{\Delta t}{\tau_{syn,jk}}\right) \tag{2.2}$$

De plus, le facteur $Fact_{jk}$ est mis à jour à chaque événement, i.e. lorsque le neurone j émet un PA, que nous notons de la manière suivante :

$$\begin{aligned}
n_j &\longleftarrow n_j + 1 \\
Fact_{jk} &\longleftarrow Fact_{jk} + \exp\left(-\frac{t_P - t_{sp_{n_j+1}}}{\tau_{syn,jk}}\right)
\end{aligned} \tag{2.3}$$

La première partie du calcul de l'équation (2.1) dépend uniquement de paramètres synaptiques (voir Algorithme 2) tandis que la deuxième partie est actualisée à chaque nouvel événement (voir Algorithme 3), et dépend du temps du dernier PA du neurone j.

Algorithme 2 : Courant synaptique I_{syn} actualisé à chaque pas de temps (interne au neurone)

Entrée : $E_{syn,jk}$ - potentiel d'inversion de la synapse entre un neurone pré-synatique et post-synaptique
Entrée : $Fact_{jk}$ - facteur du courant synaptique actualisé à chaque événement
Entrée : $g_{syn,jk}$ - conductance synaptique entre un neurone pré-synatique et post-synaptique
Entrée : $\tau_{syn,jk}$ - constante de temps de la synapse entre un neurone pré-synatique et post-synaptique
Entrée : V_k - potentiel de membrane d'un neurone post-synaptique
Sortie : $I_{syn,k}$ - courant synaptique du neurone post-synaptique

1 **pour chaque** *Neurone k* **faire**
2 $\quad I_{syn,k} \leftarrow 0$
3 $\quad$ **pour chaque** *Neurone pré-synaptique du neurone k* **faire**
4 $\quad\quad Fact_{jk} \leftarrow \exp\left(-\frac{\Delta t}{\tau_{syn,jk}}\right) \times Fact_{jk}$
5 $\quad\quad I_{syn,k} \leftarrow I_{syn,k} + g_{syn,jk} \times (V_k - E_{syn,jk}) \times Fact_{jk}$
6 $\quad$ **fin**
7 **fin**

Algorithme 3 : $Fact_{jk}$ du courant synaptique I_{syn} actualisé à chaque PA

Entrée : t_P - temps précédent
Entrée : t_{sp} - temps du PA
Entrée : τ_{syn} - constante de temps de la synapse entre un neurone présynatique et postsynaptique
Sortie : $Fact_{jk}$ - facteur du courant synaptique actualisé à chaque PA

1 **pour chaque** *Neurone j émettant un PA entre* t_P *et le temps courant* **faire**
2 $\quad$ **pour chaque** *Neurone post-synaptique k, du neurone j, généré aléatoirement* **faire**
3 $\quad\quad Fact_{jk} \leftarrow Fact_{jk} + \exp\left(-\frac{t_P - t_{sp}}{\tau_{syn,jk}}\right)$
4 $\quad$ **fin**
5 **fin**

Les algorithmes 2 et 3 définissent le processus algorithmique de l'actualisation du courant synaptique et correspondent aux lignes 2 et 5 de l'Algorithme 1. L'équation 2.1 est décrite par l'Algorithme 2 et calcule à chaque pas de temps la montée (ou la décroissance) exponentielle du courant synaptique inhibiteur (ou excitateur) entre deux PA jusqu'au prochain PA. Par conséquent, l'équation 2.3 calcule le pic du courant synaptique lors d'un PA à chaque nouvel événement, i.e. lorsque le neurone j émet un PA. Schématiquement, l'Algorithme 2 s'exécute à l'étape «méthode à pas de temps» alors que l'Algorithme 3 à l'étape «méthode événementielle» de la Fig. 2.1.

2.2 Détection des événements

D'un point de vue biologique, l'étude des synchronisations pathologiques caractéristiques de la MP exige une bonne précision des temps des PA. D'un point vu numérique, le temps du PA

doit être bien estimé afin de préserver l'ordre 2 de la méthode numérique, RK2.

En générale, le PA est détecté lorsque le potentiel de membrane dépasse un seuil donné (voir Fig. 1.3), ensuite le temps du PA est aligné au pas de temps. Cette méthode conduit à une erreur d'ordre 1, i.e. que le temps du PA est estimé avec une moins bonne précision que la méthode numérique choisie, RK2, qui est d'ordre 2 (voir sous-section 1.6.1). D'après l'article de HANSEL ET AL. [48], notre méthode de simulation perdrait en précision, d'un ordre 2, elle passerait à un ordre 1. Comme notre but est d'approcher des données biologiques et de préserver l'ordre de notre méthode de simulation, nous avons cherché une méthode de détection plus précise.

D'un point de vue mathématique, lorsque le potentiel de membrane est à son maximum, nous estimons que le temps du PA est à ce moment-là. Ainsi, le maximum du potentiel de membrane est calculé en fonction de données connues, i.e. le potentiel et la dérivée du potentiel aux points t et $t + \Delta t$. Deux méthodes sont proposées ; la méthode de l'intersection de deux droites et la méthode de la courbe de Bézier.

2.2.1 Intersection de deux droites

Le temps du PA, t_{PA}, se trouve à l'intersection des deux droites déduites des dérivées du potentiel de membrane aux frontières des temps t et $t + \Delta t$, voir Fig. 2.4(A). Au temps t, resp. $t+\Delta t$, le potentiel de membrane V_0, resp. V_1 a une dérivée $\frac{dV_0}{dt}$, resp. $\frac{dV_1}{dt}$, donnée par l'équation (A.1) dans l'annexe A.

L'équation réduite de la droite D, non parallèle à l'axe des ordonnées, est définie par l'équation $y = a_D x + b_D$ avec a_D la pente de la droite D et b_D l'ordonnée par laquelle la droite D passe. Ainsi, nos deux droites D_0, resp. D_1, passant par le point $P_0 = (t, V_0)$, resp. $P1 = (t + \Delta t, V_1)$, avec une pente $\frac{dV_0}{dt}$, resp. $\frac{dV_1}{dt}$, sont définies :

$$\begin{array}{rcl} D_0 : y = \frac{dV_0}{dt} x + b_0 \\ D_1 : y = \frac{dV_1}{dt} x + b_1 \end{array} \qquad \forall x \in [t, t + \Delta t] \text{ avec } y \in \mathbb{R} \tag{2.4}$$

où $b_0 = V_0 - \frac{dV_0}{dt} t$ et $b_1 = V_1 - \frac{dV_1}{dt} (t + \Delta t)$. Les deux droites se croisent au point $P_{PA} = (t_{PA}, V_{PA})$, ainsi

$$\begin{array}{rcl} D_O \cap D_1 & \Leftrightarrow & b_0 + \frac{dV_0}{dt} t_{PA} = b_1 + \frac{dV_1}{dt} t_{PA} \\ & \Leftrightarrow & t_{PA} = \frac{b_1 - b_0}{\frac{dV_0}{dt} - \frac{dV_1}{dt}} \end{array} \tag{2.5}$$

à condition que $\frac{dV_0}{dt} \neq \frac{dV_1}{dt}$. Effectivement, ce cas ne se présentera jamais, car lors de la détection du PA les deux dérivées doivent être de signes opposés.

Cette méthode permet de calculer le temps du PA avec plus de précision, que la méthode classique alignant le temps du PA au pas de temps. Cependant nous proposons une autre méthode exploitant les trois points P_0, P_1 et P_{PA}, calculé précédemment, étant la *courbe de Bézier*.

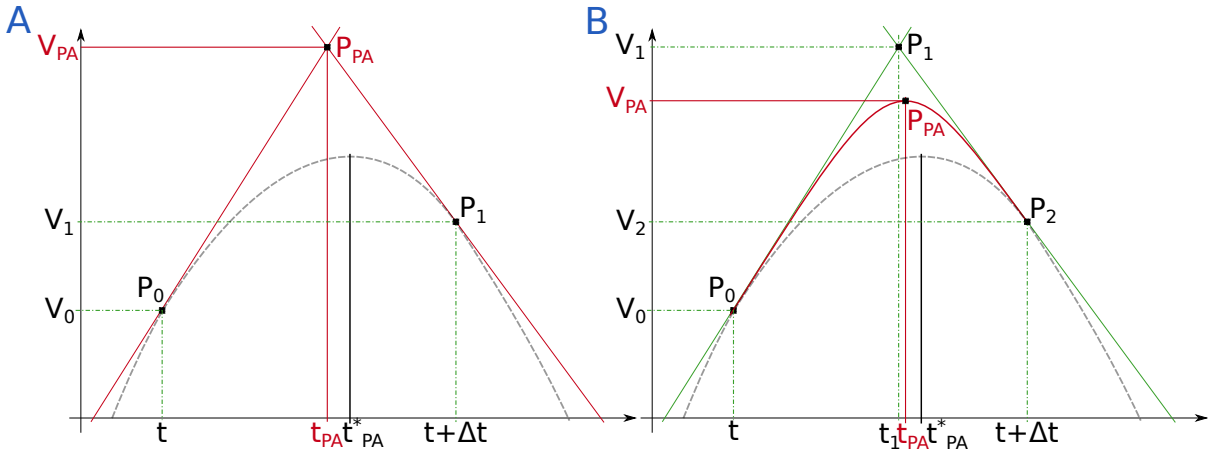

Fig. 2.4 – Représentation des deux méthodes déterminant le temps du PA. Le temps de référence du PA, t^*_{PA}, aligné au pas de temps $\Delta t = 1e^{-7}$. (A) Temps du PA, t_{PA}, obtenu à partir de l'intersection des droites issues des dérivées aux points P_0 et P_1, ligne pleine rouge. (B) Temps du PA, t_{PA}, obtenu à partir du maximum de la courbe de Bézier définie aux points P_0, P_1 et P_2. Inspirée de l'article [13].

2.2.2 Courbe de Bézier

La courbe de Bézier, $B(z)$, quadratique, i.e. d'ordre deux, est définie par trois points de contrôle P_0, P_1 et P_2, définissant les tangentes P_0P_1 et P_1P_2. La courbe de Bézier quadratique convient bien à notre approximation de courbe car elle définit une courbure proche du pic de potentiel à ces petites échelles. Un degré supérieur ne serait donc pas utile dans notre cas et nécessiterait des informations supplémentaires appelant des calculs additionnels. En effet, cette courbe est définie par ses deux extrémités et un troisième point qui correspond à l'intersection des deux tangentes aux extrémités. Or, la résolution de nos équation différentielles nous donnent déjà les deux extrémités et leurs tangentes. Et nous avons vu dans la partie précédente qu'il est très simple et rapide de déduire l'intersection des deux tangentes. Enfin, nous montrons ci-dessous que le calcul de l'ordonnée maximale et la déduction du temps du PA à partir de cette courbe s'obtiennent aisément et sont très rapides à calculer.

La forme paramétrique de $B(z)$, avec $z \in [0, 1]$, est donnée par :

$$B(z) = (1-z)^2 P_0 + 2(1-z)zP_1 + z^2 P_2 \tag{2.6}$$

Le temps du PA, t_{PA}, est déduit de la courbe de Bézier par les points $P_i = (t_i, V_i)$ définis aux temps t_i, avec $i \in \{0, 1, 2\}$, et aux potentiels de membrane V_i. Les points à l'extrémité de la courbe P_0, resp. P_2, sont donnés par les potentiels V_0, resp. V_2 aux instants t, resp. $t+\Delta t$. Le point $P_1 = (t_1, V_1)$ est déterminé par l'intersection des droites passant par P_0 et P_2 et de leurs dérivées respectives, $\frac{dV_0}{dt}$ et $\frac{dV_2}{dt}$. Le temps t_1 est le temps du PA de la sous-section 2.2.1, i.e. l'équation (2.5), et le potentiel V_1 se déduisant des équations (2.4) et (2.5) au point $(x, y) = (t_1, V_1)$:

$$t_1 = \frac{b_2 - b_0}{\frac{dV_0}{dt} - \frac{dV_2}{dt}} \quad \text{et} \quad V_1 = \frac{dV_2}{dt} t_1 + b_2 = \frac{b_2 \frac{dV_0}{dt} - b_0 \frac{dV_2}{dt}}{\frac{dV_0}{dt} - \frac{dV_2}{dt}} \tag{2.7}$$

La plus grande valeur du potentiel V est le maximum de la courbe de Bézier et donc le temps du PA se trouve à ce point :

$$\begin{aligned} \text{maximum de } B(z) &\Leftrightarrow \frac{dB}{dt}(\hat{z}) = 0 \\ &\Leftrightarrow 2\hat{z}(V_0 - 2V_1 + V_2) + 2(V_1 - V_0) = 0 \\ &\Leftrightarrow \hat{z} = -\frac{V_1 - V_0}{V_0 - 2V_1 + V_2} \quad \text{avec } V_0 - 2V_1 + V_2 \neq 0 \end{aligned} \tag{2.8}$$

avec $\hat{z} \in [0, 1]$. Ainsi, le temps du PA, t_{PA}, est

$$t_{PA} = (1 - \hat{z})^2 t_0 + 2\hat{z}(1 - \hat{z}) t_1 + \hat{z}^2 t_2 \tag{2.9}$$

2.2.3 Résultats : Ordre de la méthode de simulation

Contexte expérimental Les simulations présentées ci-dessous ont été réalisées sur `Euphrosyne`. `Euphrosyne` est un Dell R720 sous Linux Debian 4.9 amd64 avec 2 Inter(R) Xeon(R) CPU E5-2640 v2 @ 2.00 GHZ avec 8 coeurs chacun, et 128GB de RAM. Les temps ont été mesurés avec la fonction d'*interface de programmation pour le calcul parallèle*, en anglais ***Open Multi-Processing***, (OpenMP), `omp_get_wtime()`, et le programme a été compilé avec `gcc` 6.3.0 et le niveau d'optimisation `-O3`.

Sur plusieurs simulations répétées ($N_t = 10$), de $20ms$ avec une initialisation aléatoire, l'ordre de la méthode de simulation est estimé sur le dernier PA d'un neurone. Trois méthodes ont été comparées, l'intersection des droites (**RK2Droites**), le maximum de la courbe de Bézier (**RK2Bezier**) et la méthode alignant le temps du PA au pas de temps (**RK2Seuil**), lorsque le seuil est atteint. Nous avons fait varier le pas de temps, entre $\Delta t = 5e^{-5} ms$ et $\Delta t = 0,01 ms$, afin d'estimer l'ordre de notre méthode. La liste complète des paramètres de simulation de cette expérience est donnée dans le Tab. 2.1.

Paramètres de simulation		
Paramètres généraux	Taille du réseau (# de neurones) Durée $[ms]$	1 20
Paramètres du courant appliqué	Type du courant appliqué $I_{app}\,[\mu A]$	constant 0.5
Paramètres synaptiques	aucune connexion	

Tab. 2.1 – Liste des paramètres de simulation estimant l'ordre de la méthode.

Il n'existe pas de solution analytique pour l'équation (A.1). L'erreur du temps de simulation va être calculée à l'aide d'une référence de simulation basée sur un très petit pas de temps,

$\Delta t = 1e^{-7} ms$, qui est considéré assez petit pour que l'erreur soit jugée négligeable. Par conséquent, l'ordre de la méthode simulation ne sera pas exactement deux, malgré le schéma numérique RK2. L'erreur sur le temps du PA est définie comme :

$$\epsilon_t = \frac{1}{N_t} \sum_{i=0}^{N_t} \mid \hat{t}_i - t_i \mid \tag{2.10}$$

où N_t est le nombre d'essais, $\hat{t}_i$ est le temps du dernier PA du neurone à l'essai i et t_i est le temps de référence du PA avec $\Delta t = 1e^{-7} ms$ à l'essai i. Dans la Fig 2.5, l'erreur ϵ_t sur le temps du PA est représentée en fonction du pas de temps Δt pour les trois méthodes d'interpolation différentes, en échelle logarithmique. La régression linéaire, i.e. la relation entre Δt et ϵ_t, fournit une droite dont la pente représente l'ordre des trois méthodes respectives. Ainsi, les méthodes **RK2Seuil** et **RK2Droites** sont des méthodes de premier ordre, $\mathcal{O}(\Delta t)$ tandis que la méthode **RK2Bezier** est plus précise et conserve l'ordre de la méthode de RK2, l'ordre 2, $\mathcal{O}(\Delta t^2)$. Sur toutes les prochaines simulation, le temps du PA est calculé avec la méthode **RK2Bezier**.

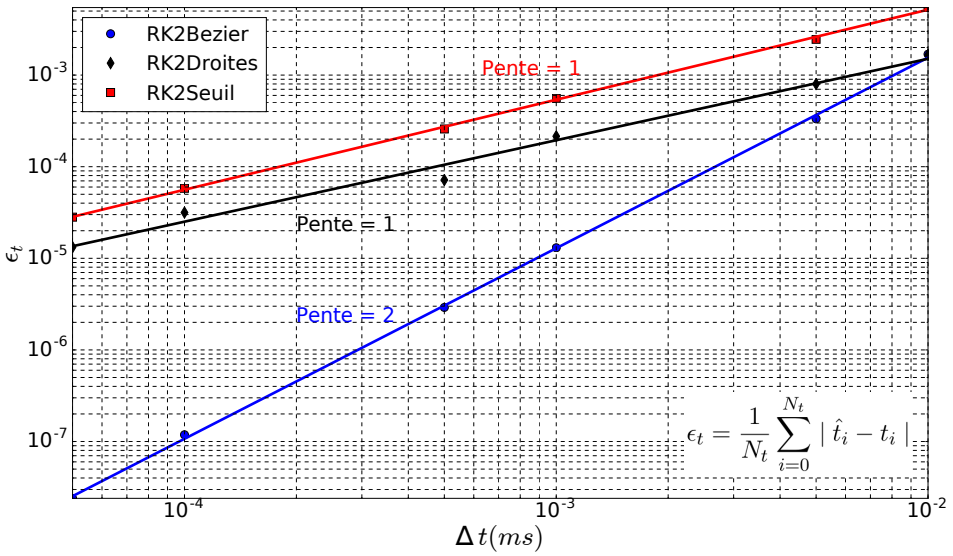

Fig. 2.5 – Comparaison de l'ordre des trois méthodes, évaluant le temps du PA. L'erreur d'évaluation du temps du PA, ϵ_t, en fonction du pas de temps Δt. Tirée de l'article [13].

2.3 Génération de la connectivité à chaque événement

Dans cette section, une approche, évitant le stockage de l'ensemble de la connectivité du réseau de neurones, va être proposée afin de réduire considérablement la consommation de mémoire lors de nos simulations. L'ensemble des neurones post-synaptiques sont générés de manière pseudo-aléatoire après chaque événement. Lytton et al. [66] est le premier article proposant cette idée, de génération de connectivités événementielles, mais dans le cas de neurones abstraits. Cette approche a ensuite été reprise par Izikevich et Edelman [55] pour des neurones IF quadratiques et plus tard par Knight et Nowotny [59] en 2021 pour des neurones IF.

2.3.1 Génération pseudo-aléatoire

La génération pseudo-aléatoire [4] garantit que pour une même graine, la séquence est reproductible, i.e. pendant toute la simulation et aussi longtemps que la graine du neurone est identique, nous générons exactement les mêmes connexions synaptiques pour un neurone. La graine de chaque neurone est définie par le numéro global du neurone dans le réseau. Afin de renforcer notre approche, nous avons donc utilisé des générateurs *générateur de nombres pseudo-aléatoire*, en anglais ***P**seudo**R**andom **N**umber **G**enerator*, (PRNG) distincts, initialisés avec une graine unique pour chaque neurone, afin que la connectivité soit différente d'un neurone à l'autre. Le PRNG est un algorithme générant une suite de nombres avec certaines propriétés mathématiques afin que la séquence ressemble le plus possible à une séquence réellement aléatoire, où l'on ne trouve pas de corrélation simple. Par exemple, les nombres sont indépendants les uns des autres, de plus il est difficile de repérer des suites de nombres suivant une même règle. Ainsi, lorsqu'un neurone émet un PA, ses connexions post-synaptiques sont générées de manière pseudo-aléatoire par rapport à sa graine, ensuite les courants synaptiques de ces neurones post-synaptiques sont mis à jour.

Le PRNG que nous avons utilisé est un générateur de Lehmer64 [63], qui fonctionne sur 64 bits. Ainsi il fournit $2^{64} \approx 10^{19}$ graines. En comparaison le cerveau humain comporte environ 10^{11} neurones. Le nombre de graines est donc largement suffisant dans notre cas. La période de ce générateur est également bien au-delà du nombre de neurones dans le cerveau.

Dans notre contexte, nous avons utilisé l'Algorithme 4 pour effectuer un tirage sans remise parmi tous les neurones post-synaptiques possibles. Cet algorithme est similaire au *selection sampling* [31, 92] qui génère les nombres en ordre croissant et présente l'avantage d'avoir un coût mémoire constant et très faible. Lors de l'initialisation de la simulation, le nombre de neurones post-synaptiques de chaque neurone est fixé, soit de façon absolue, soit par une densité de connexion. Ainsi, l'algorithme tire uniformément M éléments (les neurones post-synaptiques) dans un intervalle $[0, N]$ (tous les candidats possibles) sans répétition. L'algorithme consiste à analyser de façon successive l'ensemble des candidats et à tirer le candidat pseudo-aléatoirement selon une probabilité définie en fonction du nombre de candidats restants r_c et du nombre d'éléments restant à sélectionner r_s. Ainsi, la probabilité, que le neurone i soit sélectionné, est de $P(i) = \dfrac{r_s}{r_c}$, cette valeur étant bien dans un intervalle $[0, 1]$ et la distribution de probabilité est uniforme.

Dans l'Algorithme 4, la fonction `Seed()` génère une graine unique à partir du numéro du neurone. Cette opération n'est pas obligatoire, mais elle permet de garantir une distance minimale entre les graines de neurones ayant des paramètres de connectivité similaires, afin d'éviter des similitudes dans les distances entre les neurones post-synaptiques sélectionnés. La fonction `InitPRNG()` initialise l'état interne du PRNG, ainsi des états distincts sont produits à partir de graines distinctes. L'état interne est ensuite utilisé et mis à jour dans la fonction `RandomReal()` tirant une valeur aléatoire entre 0 et 1. Lorsque le modèle de neurones post-synaptiques est le même que le modèle de neurones pré-synaptiques, alors l'auto-tirage est exclu dans le but d'éviter les autapses. Par conséquent, la boucle de la ligne 4 est divisée en deux boucles, la boucle des neurones strictement inférieurs, resp. strictement supérieurs à `NumNeurone`, le neurone à partir duquel les neurones post-synaptiques sont générés.

D'autres méthodes de tirage de M éléments existent et sont théoriquement plus précises en termes de probabilités, comme le tirage aléatoire de la plus petite valeur de k réels tirés aléatoirement dans l'intervalle $[x, 1]$, avec $0 \leqslant x \leqslant 1$, théoriquement donnée par $1-(1-R)^{1/k} \times (1-x)$ où R est un réel aléatoire dans $[0, 1]$. Cette méthode nécessite néanmoins une étape supplémentaire afin d'obtenir des valeurs entières. Ces algorithmes sont plus coûteux en termes de calcul par

Algorithme 4 : Génération de connectivités pseudo-aléatoire

Entrée : NumNeurone - neurone à partir duquel les neurones post-synaptiques sont générés
Entrée : NbNeuronesModelePost - nombre total de neurones dans le modèle post-synaptique
Entrée : NbASelect - nombre de neurones post-synaptiques à sélectionner
Sortie : Liste - Liste de neurones sélectionnés

```
1  État ← InitPRNG(Seed(NumNeurone))
2  Liste ← ∅
3  NbSelectionne ← 0
4  pour i ← 0 à NbNeuronesModelePost − 1 faire
5  |  r_c ← NbNeuronesModelePost − i
6  |  r_s ← NbASelect − NbSelectionne
7  |  si RandomReal(État) × r_c < r_s alors
   |  |  // RandomReal() renvoie une valeur dans [0;1]
8  |  |  Liste ← Liste ∪ {i}
9  |  |  NbSelectionne ← NbSelectionne + 1
10 |  fin
11 fin
```

rapport à celui présenté précédemment. Il est utile de rappeler que notre priorité est d'obtenir des séquences reproductibles en ayant un précision statistique suffisante mais pas nécessairement maximale. Notre choix en utilisant le premier algorithme est donc un bon compromis entre la rapidité en terme de calcul et la qualité de distribution des candidats sélectionnés.

2.3.2 Illustration de la méthode

Le Tab 2.2 est un exemple utilisant l'Algorithme 4 pour la génération pseudo-aléatoire de 4 éléments sur une liste de 8 éléments.

1. Lorsque l'algorithme parcourt le premier élément de la liste, l'algorithme doit sélectionner 4 candidats sur les 8 éléments pouvant être sélectionnés. La probabilité, que l'élément d'indice 1 soit sélectionné, est de $\frac{4}{8} = \frac{1}{2}$. À ce stade, l'algorithme tire une valeur dans $[0; 1]$ (voir ligne 7 de l'Algorithme 4) et retient l'élément, si cette valeur est dans $[0; \frac{1}{2}[$. Dans notre exemple, la valeur donnée par la fonction `RandomReal()` est plus grande que $\frac{1}{2}$, et cet élément n'est donc pas tiré.
2. L'algorithme passe alors à l'indice suivant, le 2. Il reste encore, 4 candidats pouvant être sélectionnés, sur les 7 éléments restant à sélectionner. L'élément, d'indice 2, a 4 chances sur 7 d'être tiré. Lorsque l'Algorithme 4 arrive à la ligne 7, il faut que la valeur tirée soit strictement plus petite que $\frac{4}{7}$ afin que l'indice 2 soit tiré et c'est le cas, dans notre exemple. L'élément d'indice 2 est le premier élément à être sélectionné par l'algorithme.
3. L'exemple se poursuit ainsi de suite. (...)

On peut vérifier que les probabilités de tirage de chaque élément sont bien identiques entre elles :

Liste des indices des éléments		1	2	3	4	5	6	7	8
# d'éléments restant à sélectionner	$\mathbf{r_s}$	4	4	3	2	2	1	1	1
# de candidats restants	$\mathbf{r_c}$	8	7	6	5	4	3	2	1
Probabilité de sélection	$\mathbf{P_S(i)}$	$\frac{4}{8}$	$\frac{4}{7}$	$\frac{3}{6}$	$\frac{2}{5}$	$\frac{2}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{1}$
Élément sélectionné	**i**		2	3			6		8

Tab. 2.2 – Exemple de l'Algorithme 4 utilisé pour la génération de nombres pseudo-aléatoires avec $M = 4$ et $N = 8$

$$
\begin{aligned}
P(2) &= \frac{4}{8}.\frac{3}{7} + \left(1 - \frac{4}{8}\right).\frac{4}{7} \\
&= \frac{4}{8}.\frac{3}{7} + \frac{4}{8}.\frac{4}{7} \\
&= \frac{4}{8}
\end{aligned}
$$

$$
\begin{aligned}
P(3) &= \frac{4}{8}.\left(\frac{3}{7}.\frac{2}{6} + \left(1 - \frac{3}{7}\right).\frac{3}{6}\right) + \left(1 - \frac{4}{8}\right).\left(\frac{4}{7}.\frac{3}{6} + \left(1 - \frac{4}{7}\right).\frac{4}{6}\right) \\
&= \frac{4}{8}.\left(\frac{3}{7}.\frac{2}{6} + \frac{4}{7}.\frac{3}{6}\right) + \frac{4}{8}.\left(\frac{4}{7}.\frac{3}{6} + \frac{3}{7}.\frac{4}{6}\right) \\
&= \frac{4}{8}.\frac{3}{7} + \frac{4}{8}.\frac{4}{7} \\
&= \frac{4}{8}
\end{aligned}
$$

et ainsi de suite.

2.3.3 Discussion sur l'algorithme de tirage sans remise

Vers la fin de mes travaux de thèse, une nouvelle méthode de tirage sans remise a été proposée par Daniel Ting [91], basée sur des distributions β-binomiales. Outre un coût mémoire équivalent, elle présente l'avantage de nécessiter un nombre de tirages aléatoires équivalent au nombre d'éléments à sélectionner et non à la taille de l'ensemble de sélection. Il est clair qu'un tel algorithme est préférable dans notre cas puisqu'il permet de réduire le temps de génération des connexions entre neurones.

N'ayant pas connaissance de ce nouvel algorithme au moment des développements de SiReNe, les tests présentés dans la suite ont été réalisés avec l'algorithme 4. Cependant, les résultats et leur interprétation restent totalement pertinents car le changement d'algorithme de sélection implique uniquement une réduction des temps de calcul d'un pourcentage constant du temps total, et il ne change donc pas le la tendance globale des performances.

Il est prévu de remplacer, dans une version ultérieure de SiReNe, notre algorithme actuel de sélection par une version adaptée de celui de Daniel Ting [91].

2.3.4 Résultats : Comparaison des performances des différentes méthodes

Contexte expérimental Les simulations présentées ci-dessous ont été réalisées sur `Euphrosyne`. `Euphrosyne` est un Dell R720 sous Linux Debian 4.9 amd64 avec 2 Inter(R) Xeon(R) CPU E5-2640 v2 @ 2.00 GHZ avec 8 coeurs chacun, et 128GB de RAM. Pour les simulations sur SiReNe, les temps ont été mesurés avec la fonction d'OpenMP, `omp_get_wtime()`, et le programme a été compilé avec `gcc` 6.3.0 et le niveau d'optimisation `-O3`. Cependant, pour BRIAN2, les temps ont été mesuré avec la fonction *time* du module *time* de python.

Dans **SiReNe**, deux méthodes de simulation existent ; l'approche classique, à pas de temps et l'approche hybride, qui combine un calcul à pas de temps pour la dynamique du neurone et un calcul événementiel pour les synapses. De plus, il est possible de stocker la connectivité ou de la générer à chaque événement. Dans cette section, nous comparons 4 approches différentes et déterminons laquelle est la plus efficace en comparant le temps d'exécution et la consommation mémoire. Par ailleurs, ces approches seront comparées avec BRIAN2 [47] un simulateur référence dans la communauté et le plus utilisé pour les neurones à PA.

Dans les Fig. 2.6 et 2.7, le temps d'exécution et l'allocation de la mémoire sont comparés en fonction de la taille d'un réseau MSN avec une connectivité partielle, de 30%. Pendant $100ms$ de temps biologique, les cinq approches comparées sont l'approche classique, *à **P**as de **T**emps avec **S**tockage de connectivité (PT-S)*, la même approche *à **P**as de **T**emps avec **G**énération de connectivité (PT-G)*, l'approche hybride, ***M**ise à **J**our à chaque **É**vénement avec **S**tockage de connectivité (MJE-S)*, la même approche ***M**ise à **J**our à chaque **É**vénement avec **G**énération de connectivité (MJE-G)* et la simulation avec BRIAN2. La liste complète des paramètres de simulation de ces expériences est donnée dans le Tab. 2.3.

Paramètres de simulation		
Paramètres généraux	Taille du réseau	100-10k
	Durée $[ms]$	100
Paramètres du courant appliqué	Type du courant appliqué	escalier avec bruit
	moment où le courant change $[ms]$	50
	avec un bruit	$\times [0, 1]$
	$I_{app}\,[\mu A]$ à la 1^{ire} partie	-10
	$I_{app}\,[\mu A]$ à la 2^{ime} partie	0,5
	Amplitude du bruit	300
Paramètres synaptiques	[%] de connectivité	30
	$\tau[ms]$	12
	$E_{syn}[mV]$	-80
	$g_{syn}[nS]/\#connections$	0,6

Tab. 2.3 – Liste des paramètres de simulation pour la comparaison des différentes approches.

Globalement sur les cinq approches comparées, les deux approches événementielles ont le plus petit temps de simulation. BRIAN2 est plus long que les 4 autres approches jusqu'à 1 000 neurones, ensuite il est plus rapide que les deux approches à pas de temps. Niveau mémoire, la génération de connectivité est l'approche allouant le moins de mémoire sur toutes les approches.

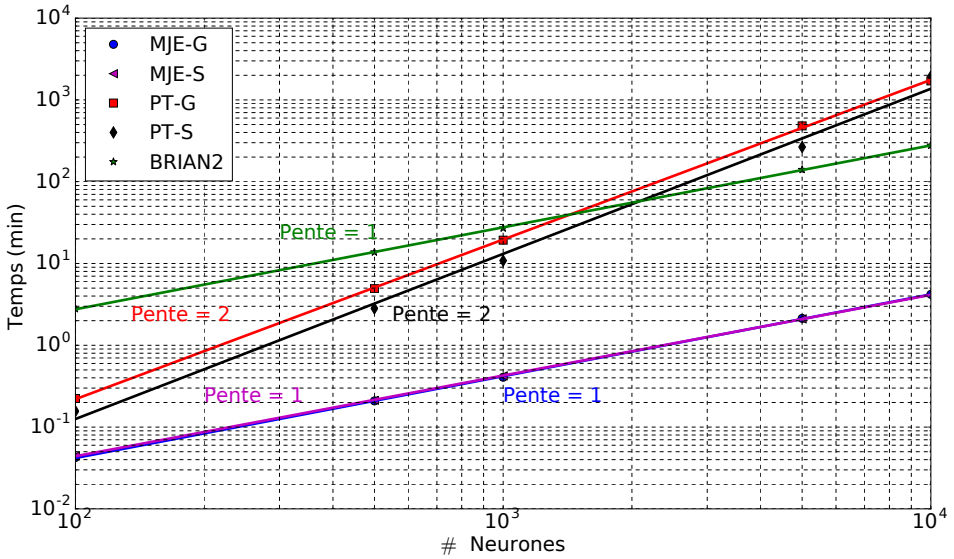

Fig. 2.6 – Le temps d'exécution est comparé en fonction de la taille d'un réseau MSN avec 30% de connectivité pour différentes approches de simulation, PT-S, PT-G, MJE-S, MJE-E et le simulateur BRIAN2. . Tirée de l'article [13].

L'allocation en mémoire de l'approche MJE-G est quasi constante en fonction du nombre de neurones. En revanche, l'allocation de mémoire augmente polynomialement pour les approches à pas de temps, PT-G et PT-S, et BRIAN2.

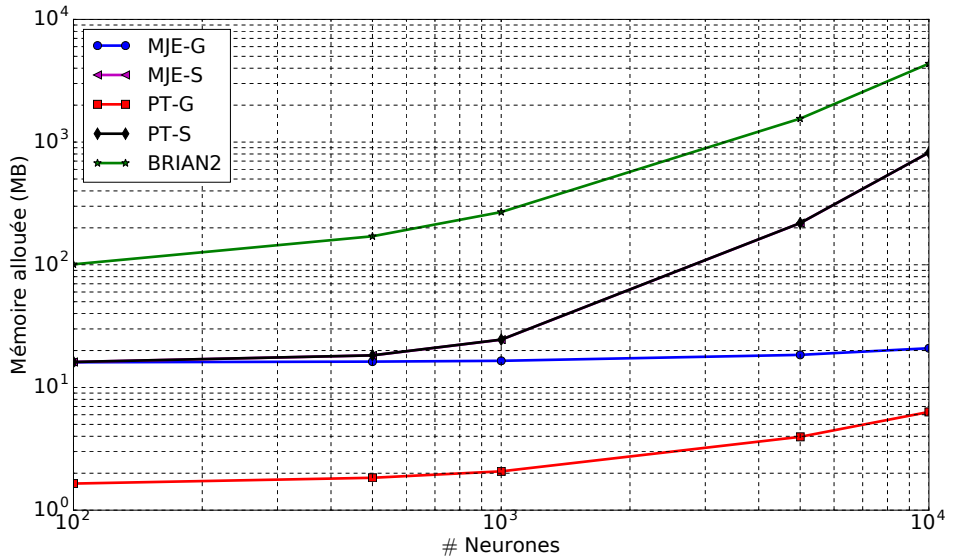

Fig. 2.7 – La mémoire allouée est comparée en fonction de la taille d'un réseau MSN avec 30% de connectivité pour différentes approches de simulation, PT-S, PT-G, MJE-S, MJE-E et le simulateur BRIAN2, tirée de l'article [13].

Une relation entre le temps d'exécution et le nombre de neurones est observée, i.e. le temps augmente comme la puissance du nombre de neurones, donc $t \propto n^k$, où t est le temps d'exécution, n le nombre de neurones et k l'exposant de la loi de puissance. Dans la Fig. 2.6, les pentes des

méthodes MJE-S, MJE-G et de la simulation de BRIAN2 sont d'ordre un ($k = 1$), alors que les deux autres méthodes, les pentes sont d'ordre deux ($k = 2$). En d'autres termes, le temps d'exécution s'accroît linéairement pour les trois méthodes, citées précédemment, et quadratiquement pour les deux autres. Néanmoins, les méthodes MJE-S et MJE-G, qui ont des temps de simulation similaire, ont des temps de simulation plus petit que celui de BRIAN2.

2.4 Conclusion du chapitre

Ce chapitre, nous a permis de mettre en avant (1) une méthode de détection de PA, l'interpolation par une courbe de Bézier quadratique, conservant l'ordre de la méthode numérique, RK2 et (2) une méthode hybride, à pas de temps pour la dynamique du neurone et événementielle pour la dynamique synaptique. Par conséquent, nous évitons le stockage complet de la connectivité d'un réseau et la générons pseudo-aléatoirement à chaque événement. Ainsi, d'un côté, nous diminuons le temps de simulation et de l'autre, l'allocation de la mémoire est fortement réduite permettant de simuler des réseaux de neurones de très grandes tailles. Les différentes améliorations proposées dans ce chapitre sont mises en œuvre dans le logiciel SiReNe présenté dans le chapitre suivant.

3

Développement du logiciel SiReNe

Sommaire

Depuis plusieurs années, le logiciel **SiReNe** est développé dans le laboratoire LORIA-INRIA. Au début, il s'agissait d'un simulateur uniquement à pas de temps. Dans le cadre de cette thèse, nous l'avons étendu à un simulateur hybride, i.e. à pas de temps pour la dynamique neuronale, et événementiel pour la dynamique synaptique. Ces aspects ont été développés dans le chapitre précédent. Le logiciel, implémenté en C, est maintenant capable de simuler des grands réseaux neuronaux, soit avec la méthode classique à pas de temps, soit avec la méthode hybride.

SiReNe est un logiciel en libre accès. Il peut être utilisé, et modifié, selon les termes de la licence publique générale GNU. Le code source est disponible à l'adresse suivante : `https://sirene.gitlabpages.inria.fr/sirene`

Sur la dernière version de SiReNe, le **M**ulti**T**hreading (MT) a été implémenté avec OpenMP [2, 6]. Ainsi, les calculs relatifs aux neurones sont distribués sur différents threads. Une méthode utilisant OpenMP et **M**essage **P**assing **I**nterface (MPI) [5, 1] a aussi été développée afin d'utiliser MT et **M**ulti**M**achines (MM) simultanément.

Ce chapitre se compose de trois sections principales. La première section détaille les différentes structures principales de SiReNe. La deuxième section décrit le fonctionnement du logiciel et la dernière section est sur le calcul parallèle implémenté dans SiReNe.

3.1 Structures

SiReNe comporte 4 structures principales. La structure `Simulation` englobe tous les paramètres de simulation. La structure `SynapseModel` contient tous les paramètres des synapses. La structure `NeuralModel` concerne les paramètres du modèle de neurones. Enfin, la structure `Neuron` comprend les paramètres des neurones. Dans la Fig 3.1, les différentes structures y sont exposées avec leurs principaux paramètres.

Il existe une autre structure moins importante et que l'utilisateur n'utilise pas directement, puisqu'elle est interne au logiciel, c'est `SpikeEvent` dans laquelle le temps du PA et le numéro du neurone émettant un PA sont stockés.

3.1.1 Structure : `Simulation`

La structure principale de SiReNe est `Simulation` où les paramètres de simulation sont fixés. Les principaux paramètres sont le pas de temps, `TimeStep`, aussi le temps de simulation, `TotalTimeSim`, et le nombre de threads[1] utilisé pour la parallélisation du code, `NbThreads`. D'autres paramètres en lien avec les modèles y sont aussi initialisés, plus précisément, le nombre de modèles de neurones dans le réseau, `NumberNeuralModels`, le nombre de variables sur l'ensemble du réseau, `NumberOfVariables`, le nombre total de neurones sur tout le réseau, `TotalNumberOfNeurons` et pour finir le nombre de différents modèles de synapses dans le réseau, `NumberSynapseModels`. Les tableaux `Variables` et `DVar` de longueur `NumberOfVariables` comportent toutes les variables du réseau, resp. les dérivées des variables. Les variables V, m, h et n du système d'équations (1.4), resp. les dérivées de ces variables, sont sauvegardées dans l'ordre des modèles et de leurs neurones respectifs. De plus, les pas de temps précédent `PrtNum`, et actuel `CrtStep` de ces variables sont sauvegardés dans ces tableaux. Un exemple sur la construction du tableau `Variables` est donné dans Tab. 3.1. Le tableau `DVar` a exactement la même organisation sauf que ce sont les dérivées. D'autre part, il existe deux tableaux temporaires, `TmpDer` et `TmpVar`, ayant la même construction que les tableaux `Variables` et `DVar`, sauf qu'ils ne contiennent que le temps courant. Ces tableaux sont uniquement utiles pour des calculs intermédiaires et internes à SiReNe.

Quant aux fonctions génériques, elles peuvent être instanciées par plusieurs fonctions, voir la Tab. 3.2. Cela permet à l'utilisateur de choisir ou créer sa propre fonction et de la faire pointer sur une fonction générique. Trois fonctions génériques sont définies dans cette structure, la fonction `NumericalMethod` représentant les différentes méthodes numériques, vue à la section 1.6.1, la fonction `Interpolator` définissant la méthode de calcul du temps du PA, vue aux sous-sections 2.2.1 et 2.2.2 et la fonction `RandGenerConnect` donnant la méthode de génération aléatoire des connectivités, vue à la section 2.3.

1. Un thread est similaire à un processus, il représente l'exécution d'une, ou plusieurs instruction(s), cependant un processus possède sa propre mémoire, alors que les threads d'un même processus se partagent la mémoire.

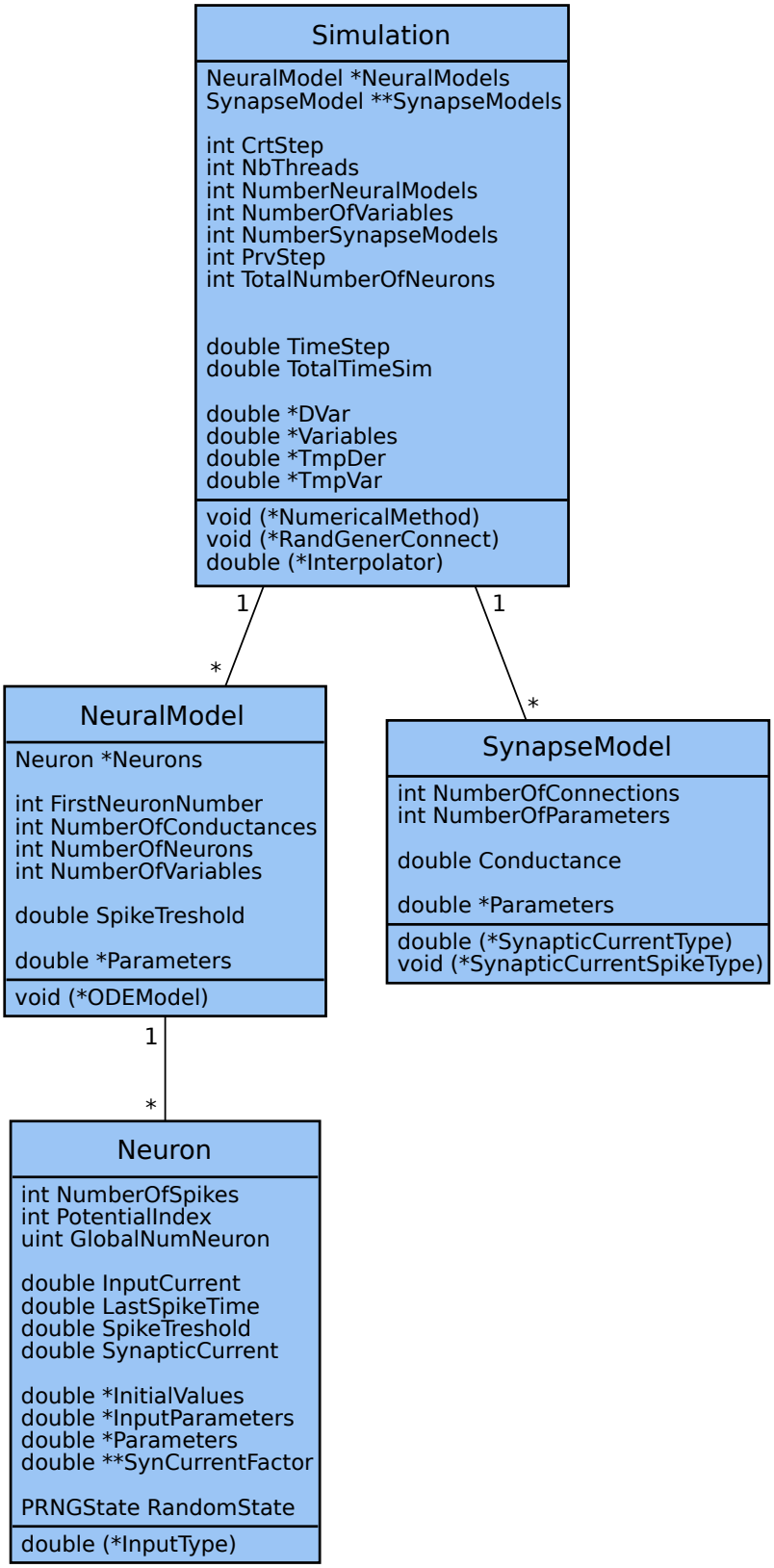

Fig. 3.1 – Structures fondamentales de SiReNe. Le «1» en dessous des boîtes signifie que la structure, p.ex. `NeuralModel`, est associé une seule fois à la structure `Simulation`. L'étoile au dessus des boîtes signifie que la structure, p.ex. `Simulation`, peut contenir plusieurs structures, p.ex. `NeuralModel`.

		Variables	
		PrvStep	CrtStep
1^{er} *modèle*	1^{er} *neurone*	$V_{11,p}$	$V_{11,c}$
		$m_{11,p}$	$m_{11,c}$
		$h_{11,p}$	$h_{11,c}$
		$n_{11,p}$	$n_{11,c}$
	$2^{ième}$ *neurone*	$V_{12,p}$	$V_{12,c}$
		$m_{12,p}$	$m_{12,c}$
		$h_{12,p}$	$h_{12,c}$
		$n_{12,p}$	$n_{12,c}$
	$\vdots$	$\vdots$	$\vdots$
$2^{ième}$ *modèle*	1^{er} *neurone*	$V_{21,p}$	$V_{21,c}$
		$m_{21,p}$	$m_{21,c}$
		$h_{21,p}$	$h_{21,c}$
	$2^{ième}$ *neurone*	$V_{22,p}$	$V_{22,c}$
		$m_{22,p}$	$m_{22,c}$
		$h_{22,p}$	$h_{22,c}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Tab. 3.1 – Exemple des tableaux `Variables` et `DVar`.

Fonctions génériques	**Fonctions implémentées**		
`NumericalMethod`	`Euler`	`RungeKutta2`	
`Interpolator`	`NoInterpolation`	`LinearInterpolation`	`BezierInterpolation`
`RandGenerConnect`	`RandomConnections`		

Tab. 3.2 – Les fonctions génériques pointent vers ces fonctions implémentées qui se trouvent actuellement sur SiReNe. `NoInterpolation` où le temps du PA est aligné à un seuil, `LinearInterpolation` utilisant l'intersection entre deux droites et `BezierInterpolation` implémentant la courbe de Bézier.

Deux structures fondamentales sont définies dans `Simulation`, la structure `NeuralModel` et `SynapsesModels`. Les différents modèles de neurones du réseau sont représentés par la structure `NeuralModel` et nous définissions le nombre de modèles par `NumberNeuralModels`.

Les synapses entre les neurones ont été regroupées en fonction des modèles de neurones connectés. Lorsqu'un modèle de neurone est connecté à un autre modèle de neurone alors nous avons fait le choix de ne définir qu'un seul type de synapse entre ces deux modèles, e.g. un modèle de neurones excitateur sera connecté à un autre modèle de neurones via une synapse excitatrice (resp. inhibitrice). En outre, les synapses au sein de deux modèles de neurones connectés auront les mêmes paramètres de connectivités.

Par conséquent pour les modèles de synapses, l'allocation mémoire augmente comme le nombre de modèles de neurones au carré (au lieu du nombre de tous les neurones dans le réseau au carré). Ce qui représente un gain énorme en mémoire. Un exemple schématisant le tableau de synapses est donné dans Tab. 3.3. Les valeurs booléennes, 0 et 1, représentent si oui, ou non, les modèles sont connectés entre eux i.e. 1 lorsque les modèles sont connectés et 0 sinon.

	1^{er} *modèle*	$2^{ième}$ *modèle*	...	$n^{ième}$ *modèle*
1^{er} *modèle*	1	0	...	1
$2^{ième}$ *modèle*	0	1	...	1
⋮	⋮	⋮	⋱	⋮
$n^{ième}$ *modèle*	1	0	...	0

Tab. 3.3 – Exemple d'un tableau de modèles de synapses `**SynapseModel`.

Dans la structure `Simulation`, des fichiers de sauvegarde y sont aussi définis. Il est possible de sauvegarder les potentiels, les PA, le *potentiel de champ local*, en anglais ***L**ocal **F**ield **P**otentiel*, (LFP) et d'autres données en lien avec la simulation du réseau. Le temps passé sur les différentes fonctions peut aussi être affiché.

3.1.2 Structure : `SynapseModel`

La structure `SynapseModel` se compose des paramètres de synapses connectant les neurones entre eux. Dans cette structure, nous y trouvons le nombre de connexions, `NumberOfConnections`, le nombre de paramètres définissant la synapse, `NumberOfParameters`, et la conductance du modèle de synapses, `Conductance`. Dans le tableau `Parameters`, de longueur `NumberOfParameters`, la constante de temps et le potentiel d'inversion du modèle de synapses y sont définis.

Les fonctions, `SynapticCurrentType` et `SynapticCurrentSpikeType`, définissent le type de courant synaptique, i.e. le modèle à exponentielle simple, vu à la sous-section 1.2.2, ou le modèle à exponentielle avec hétérogénéité. L'hétérogénéité est ajoutée à la conductance synaptique afin que les synapses soient différentes entre les neurones de deux modèles. `SynpaticCurrentType` calcule le courant synaptique entre deux pas de temps, définie par l'équation (2.1), et `SynapticCurrentSpikeType` actualise le courant postsynaptique du neurone émetteur lors d'un PA, définie par l'équation (2.3).

3.1.3 Structure : `NeuralModel`

La structure `NeuralModel` mutualise les différents paramètres des neurones en fonction du modèle de neurones. Elle est constituée du numéro global du premier neurone du modèle, `FirstNeuronNumber`, le nombre de neurones dans le modèle, `NumberOfNeurons`, le nombre de conductances ioniques pour ce modèle, `NumberOfConductances`, le nombre de variables, `NumberOfVariables`, et le seuil de détection du PA, `SpikeThreshold`. Le tableau `Parameters` contient les conductances ioniques du modèle.

Le modèle de neurone est choisi avec la fonction générique `ODEModel`, e.g. les GB avec le STN, le GPeA ou d'autres, vu à la sous-section 1.3.1, peuvent être choisis. Chaque modèle de neurones a sa fonction avec son système d'équations d'HH (1.4), voir le Tab. 3.4.

Fonction générique	**Fonctions pointant**				
`ODEModel`	`STN`	`GPeA`	`GPeP`	`MSN`	`FSN`

Tab. 3.4 – Fonction générique `ODEModel` et les fonctions pointant sur cette fonction générique.

3.1.4 Structure : `Neuron`

Dans la structure `Neuron`, nous y trouvons tous les paramètres spécifique aux neurones. Ainsi, elle se compose du numéro global du neurone dans le réseau, `GlobalNumNeuron`, du nombre de PA émis par le neurone `NumberOfSpikes`, du premier index attribué à ce neurone dans le tableau `Variables`, `PotentielIndex`, du courant appliqué au neurone `InputCurrent`, du temps du dernier PA du neurone, `LastSpikeTime`, du seuil de détection du PA, `SpikeThreshold`, et le courant synaptique du neurone, `SynapticCurrent`. Dans la structure, l'état du PRNG associé au neurone s'y trouve aussi.

Le tableau `InitialValues` se compose de variables d'initialisation du système d'équations (1.4). Une certaine hétérogénéité peut être ajoutée afin que tous les neurones ne s'initialisent pas avec les mêmes valeurs. `Parameters` est un tableau reprenant le tableau des conductances de la structure `NeuralModel`, mais avec la possibilité d'ajouter de l'hétérogénéité entre chaque neurone afin que les neurones n'aient pas exactement les mêmes conductances.

`InitialValues`	`Parameters`
V_{init}	g_L
m_{init}	g_{Na}
h_{int}	g_K
n_{int}	

Tab. 3.5 – Exemple des tableaux de `InitialValues` et `Parameters`.

Afin de bien comprendre le rôle du tableau `SynCurrentFact`, nous allons donner un exemple, voir Fig. 3.2. Au temps t, le neurone 4 du modèle 2 et les neurones 23 et n du modèle m émettent un PA. Comme dit dans la sous-section 2.1.2, les neurones postsynaptiques aux neurones émetteurs sont actualisés et plus précisément le facteur du courant synaptique $Fact$ des neurones postsynaptiques, voir l'équation (2.3). Si SiReNe était lancé sur uniquement un processeur, nous n'aurions qu'à implémenter le $Fact$ comme une valeur que nous actualiserions à chaque PA. Cependant, nous avons intégré au logiciel SiReNe le calcul parallèle multithreadé. En bref, les neurones sont distribués sur différents processeurs. Par contre, la mise à jour d'un même neurone postsynaptique ne doit pas être effectuée par deux processeurs en même temps. Sinon le résultat pourrait être incohérent. Le calcul parallèle sera expliqué plus clairement et plus précisément dans le chapitre suivant. Le calcul $Fact$ est un de ces cas où le calcul dépend d'autres neurones.

Ainsi, lorsque $Fact$ est actualisé, le facteur sera sauvegardé en fonction du modèle émetteur et du processeur qui a détecté le PA. Revenons à notre exemple, lorsque le neurone 4 du modèle 2 a émis un PA, alors le $Fact$ du neurone 1 du modèle 1 est actualisé à la position (2,1) du tableau `SynCurrentFact`, i.e. 2 pour le modèle émetteur 2 et 1, car c'est le processeur 1 qui a détecté le PA. Cependant le neurone 10 du modèle 2 et le neurone 23 du modèle m ont aussi émis un PA et sont connectés au neurone 1 du modèle 1. Donc le $Fact$ du neurone 1 du modèle 1 est aussi actualisé, mais aux positions (2,2) et (m,2) du tableau `SynCurrentFact`, cette fois-ci c'est le processeur 2 qui a détecté le PA. Le Tab. 3.6 est un exemple du tableau de `SynCurrentFact` du neurone 1 du modèle 1.

Avec la fonction `InputType`, nous déterminons le type de courant injecté au neurone, et le Tab 3.7 reprend les différents courants existant dans SiReNe. Les paramètres du courant appliqué sont fixés en fonction du courant choisi, dans le tableau `InputParameters`. Le Tab. 3.7 synthétise les différents paramètres à préciser lors de l'initialisation du courant injecté. Le courant constant

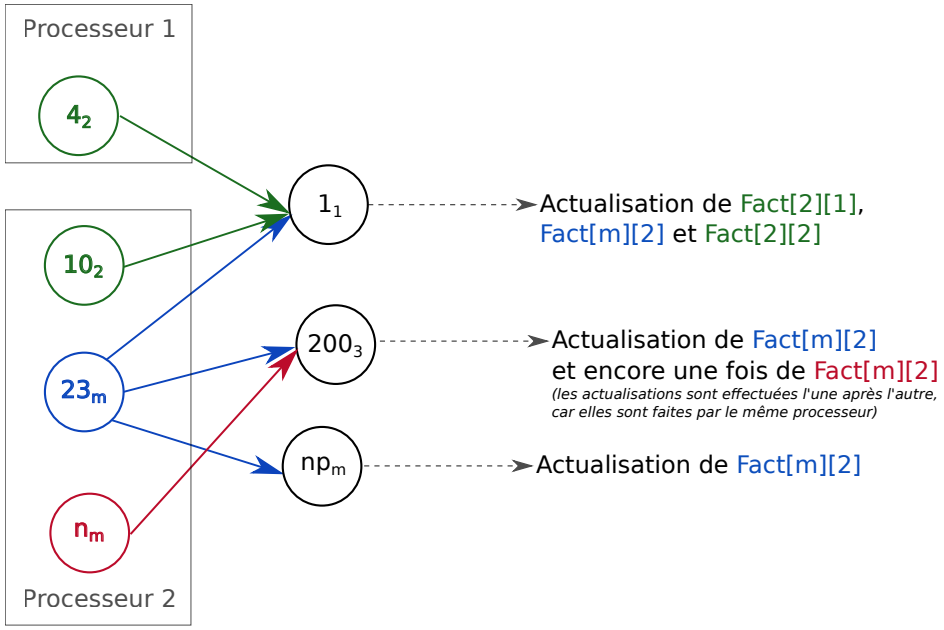

Fig. 3.2 – Exemple d'actualisation du facteur du courant synaptique, *Fact* de l'équation (2.3). Les cercles représentent les neurones, i.e. que 4_2 est le neurone 4 du modèle 2 (en vert), 10_2 est le neurone 10 du modèle 2 (en vert), n_m est le neurone n du modèle m (en rouge), np_m est le neurone np du modèle m et ainsi de suite.

processeur \ **modèle**	1	2	3	4	...	m
1	*	*val*	*	*	...	*
2	*	*val*	*	*	...	*val*

Tab. 3.6 – Exemple du tableau `SynCurrentFact` du neurone 1 du modèle 1.

n'a qu'un seul paramètre et c'est celui de l'intensité du courant. Le courant en échelon en a trois, l'intensité du courant de la première partie, l'intensité du courant de la deuxième partie et le moment du changement de l'intensité. Le courant en créneau se définit de la même manière, que le courant en échelon, sauf qu'un paramètre est ajouté pour la période du créneau.

3.2 Fonctionnement de SiReNe

Le fichier principal de SiReNe est appelé `Test.c`. Les paramètres du réseau y sont initialisés et les variables internes à SiReNe sont initialisées lorsque les fonctions `InitSimu()` et `PrepareSimu()` sont appelé dans le fichier `Test.c`. Ces fonctions sont définies dans le fichier `Sirene.c`. L'appel à la fonction `Advance()` dans `Test.c` lance le début de la simulation. Cette fonction se trouve dans `Sirene.c`.

Les différentes étapes de la simulation sont schématisées dans la Fig. 3.3. À chaque pas de temps la fonction `Iter()` est appelée, faisant appel à `NumericalMethod()`. Les courants appliqués et synaptiques sont mis à jour avant d'être rapportés dans le système d'équations de HH (1.4) de la fonction `EDOModel()`. La prochaine étape est la détection de PA. Si des PA ne sont pas

Type de courant injecté	**Paramètres à préciser selon le courant**
Échelon de courant	1. Valeur de l'intensité
Créneau de curant	1. Valeur de l'intensité dans un premier temps 2. Valeur de l'intensité dans le deuxième temps 3. Temps du changement d'intensité
Courant à impulsion	1. Valeur de l'intensité dans un premier temps 2. Valeur de l'intensité dans le deuxième temps 3. Temps du changement d'intensité 4. Période de l'impulsion

Tab. 3.7 – Type de courants appliqués `(*InputType)` et les paramètres à préciser selon le courant choisit `*InputParameters`.

détectés, la simulation passe directement au prochain pas de temps. Si des PA sont détectés, alors le temps du PA est calculé avec la courbe de Bézier. Les connexions postsynaptiques du neurone émetteur sont générées et leur courant synaptique est actualisé en fonction du PA reçu. Ensuite, la simulation passe au prochain pas de temps.

3.2.1 Initialisation des variables

Pour commencer, les paramètres de simulation sont spécifiés dans `Test.c`, e.g. le temps de simulation ou le pas de temps. Ensuite, l'utilisateur doit définir les paramètres du réseau, i.e. le nombre de modèle, le type de modèle de neurones, les conductances ioniques dans le tableau `Parameters`, les valeurs initiales du système d'équations (1.4) dans `InitialValues`, le type de courant appliqué `InputType`, les paramètres du courant appliqué dans `InputParameters`, le type de synapse dans `SynapticCurrentType`, le nombre de connexions dans `NumberOfConnections` et la conductance synaptique, resp. constante de temps et le potentiel d'inversion, dans `Conductance`, resp. `Parameters`.

Du côté interne à SiReNe dans les fichiers `InitSimu()` et `PrepareSimu()`, plusieurs variables sont initialisées, comme le nombre global de neurones dans `TotalNumberOfNeurons`, le nombre total de variables du réseau dans `NumberOfVariables` et le numéro global du neurone dans le réseau est fixé dans `GlobalNumNeuron`. Les variables d'initialisation sont recopiées dans le tableau `Variables`. Les conductances ioniques du tableau `Parameters` sont transcrites dans le même tableau de la structure `Neuron` afin d'y ajouter de l'hétérogénéité au réseau. Les autres variables sont initialisées à zéro et les autres tableaux sont alloués en mémoire.

L'Algorithme 5 définit la fonction utilisée afin d'ajouter de l'hétérogénéité au réseau. La fonction `SpecificNoise(borne,RandomState)` génère aléatoirement une valeur entre `1-borne` et `1+borne`. Ensuite cette valeur est généralement multipliée à la valeur à laquelle nous voulons ajouter une hétérogénéité.

Prenons un exemple, l'utilisateur de SiReNe donne une conductance, g_L. L'utilisateur veut que chaque neurone ait une conductance g_L différente, mais ce serait pénible de le faire à la main. Ainsi, nous avons implémenté cette fonction, afin qu'elle génère une conductance différente

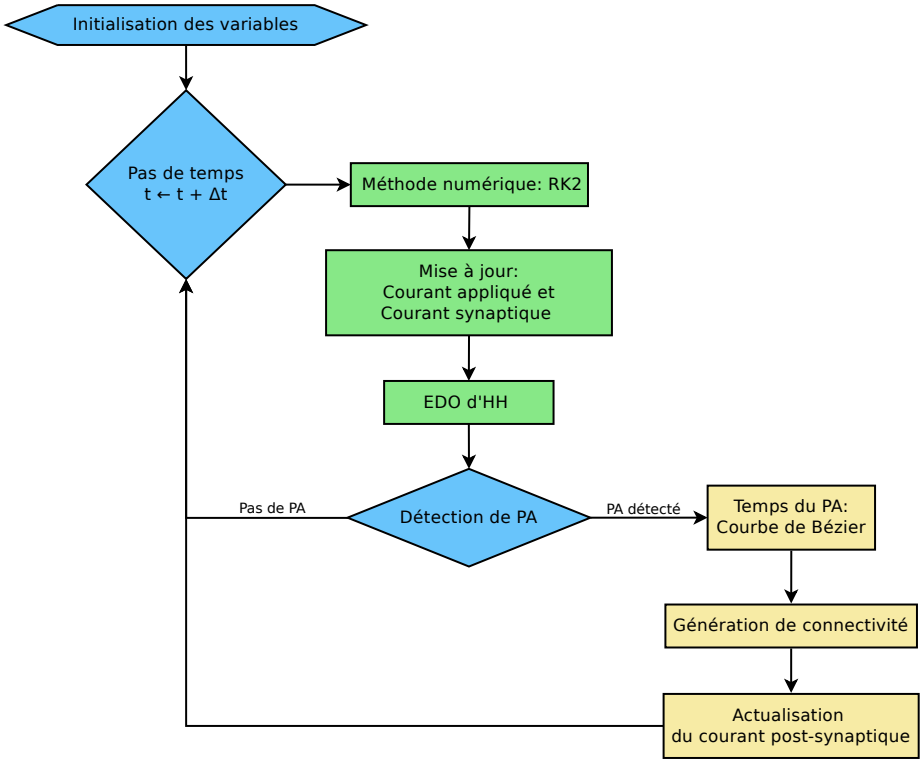

Fig. 3.3 – Représentation schématique de SiReNe, inspirée de l'article [13].

pour chaque neurone. Les conductances devront se trouver entre par exemple +/-30% de la valeur de g_L. La fonction `SpecificNoise(0.3,RandomState)` va alors générer une valeur aléatoire uniforme entre 0,7 et 1,3 pour chaque neurone. Cette valeur est ensuite multipliée à la valeur de la conductance, g_L. Ainsi, chaque neurone aura une conductance allant de $g_L * 0,7$ et $g_L * 1,3$.

Algorithme 5 : Fonction générant aléatoirement une valeur, `SpecificNoise()`

Entrée : Borne - Borne de l'intervalle
Entrée : RandomState - état associé au neurone
Sortie : Valeur se trouvant dans l'intervalle

```
1 retourner 1 + Borne/100 * ( 2 * RandomReal(RandomState) - 1)
  // RandomReal() renvoie une valeur aléatoire uniforme dans [0;1]
```

3.2.2 Méthode numérique : RK2

Lorsque toutes les variables sont initialisées, la simulation du réseau commence par l'appel à la fonction `Advance()` se trouvant dans le fichier `Sirene.c`. La fonction `NumericalMethod` du fichier `NumericalMethod.c` est appelée par la fonction `Iter()` qui elle a été appelée par la fonction `Advance()`.

La méthode RK2 est la seule à être présentée dans cette section, toutefois la méthode d'Euler est aussi implémentée dans SiReNe. L'Algorithme 6 illustre la méthode RK2 dans le logiciel SiReNe. Pour commencer, les variables au demi-pas de temps, `TmpVar`, sont calculées à l'aide des variables et des dérivées `DVar` au pas de temps précédent. Quant aux dérivées au demi-pas de temps, `TmpDer`, elles sont estimées avec la fonction `ODEModel` présentée dans la prochaine

sous-section. Ensuite, les variables au pas de temps courant sont calculées avec les variables au pas de temps précédent et les dérivées `TmpDer` au demi-pas de temps. Finalement, les dérivées au pas de temps courant sont estimées afin de les exploiter au prochain pas de temps.

Algorithme 6 : Méthode numérique de RK2

Entrée : t_p - temps précédent
Entrée : Δt - pas de temps
Entrée : NbVar - nombre de variables dans tout le réseau `NumberOfVariables`
Entrée : dVar - dérivées des variables du tableau `DVar`
Entrée : dTmpVar - dérivées temporaires des variables du tableau `TmpDer`
Entrée : VarTmp - variables temporaire du tableau `TmpVar`
Sortie : Var - variables du tableau `Variables`

1 **pour** $i \leftarrow 0$ **à** *NbVar* **faire**
2 | $\text{TmpVar}_i \leftarrow \text{Var}_i + 0.5\Delta t \times \text{dVar}_i$
3 **fin**
4 Calcul des dérivées des variables, dTmpVar, au demi-pas de temps, $t_p + 0.5\Delta t$
5 **pour** $i \leftarrow 0$ **à** *NbVar* **faire**
6 | $\text{Var}_i \leftarrow \text{Var}_i + \Delta t \times \text{dTmpVar}_i$
7 **fin**
8 Calcul des dérivées des variables, dVar, au pas de temps, t_c

L'Algorithme 7 explique plus clairement les lignes 4 et 8 de l'Algorithme 6. Le courant appliqué et synaptique, au demi-pas de temps, resp. au pas de temps, sont actualisés pour ensuite être injectés dans le système d'équations (1.4) de la fonction `EDOModel`.

Algorithme 7 : Actualisation des courants et des dérivées du système d'EDO

1 **pour chaque** *Modèle m* **faire**
2 | **pour chaque** *Neurone n* **faire**
3 | | Mettre à jour le courant appliqué avec `InputCurrent`
4 | | Mettre à jour le courant synaptique avec `SynapticCurrent`
5 | | Mettre à jour les dérivées des équations EDO d'HH
6 | **fin**
7 **fin**

3.2.3 Mise à jour : Courant appliqué et Courant synaptique

Le courant appliqué et le courant synaptique sont actualisés dans le but de calculer les dérivées du système d'équations d'HH (1.4). Les fonctions `InputType()` et `SynapticCurrent()` sont appelées dans la fonction `UpdateNeuron()` se trouvant dans le fichier `Updates.c`.

Courant appliqué

Les différents types de courant ont été condensés dans l'Algorithme 8 qui détaille un peu plus la ligne 3 de l'Algorithme 7. Cependant dans SiReNe, chaque type de courant a sa propre fonction se trouvant dans le fichier `InputCurrents.c`. La fonction du courant constant est appelée `ConstantInputCurrent`, celle du courant à échelon `StepwiseInputCurrent` et celle du courant en créneau `PulseInputCurrent`. Si l'utilisateur souhaite ajouter de l'hétérogénéité, alors il suffit

d'ajouter le mot «WithNoise» à la fin du nom de la fonction. Dans l'Algorithme 8, l'ajout d'hétérogénéité est spécifié avec «avec bruit ».

Algorithme 8 : Courant appliqué

Entrée : ParamCourantApp - paramètres du courant appliqué du tableau `*InputParameters`
Entrée : t - temps courant
Sortie : Courant appliqué

```
1  suivant le type de courant appliqué faire
2      cas où courant constant avec bruit faire
3          retourner ParamCourantApp[0]*SpecificNoise()
4      cas où courant à étage avec bruit faire
5          si t < ParamCourantApp[2] alors
6              retourner ParamCourantApp[0]*SpecificNoise()
7          sinon
8              retourner ParamCourantApp[1]*SpecificNoise()
9          fin
10     cas où courant à impulsion avec bruit faire
11         si ParamCourantApp[2] < t < ParamCourantApp[2]+ParamCourantApp[3] alors
12             retourner ParamCourantApp[1]*SpecificNoise()
13         sinon
14             retourner ParamCourantApp[0]*SpecificNoise()
15         fin
16     fin
17 fin
```

Courant synaptique

Le courant synaptique interne au neurone est actualisé. Ce calcul a déjà été abordé dans le chapitre précédent avec l'Algorithme 2, mais l'Algorithme 9 reproduit plus exactement le calcul effectué par SiReNe, de plus, il détaille la ligne 4 de l'Algorithme 7. Comme les synapses sont mutualisées en fonction des modèles de neurones connectés, alors les termes $E_{syn,km}$, $Fact_{nm}[k]$, $g_{syn,km}$ et $\tau_{syn,km}$ ne dépendent pas des neurones présynaptiques et postsynaptiques, mais ils dépendent des modèles pré-synaptiques et post-synaptiques.

Algorithme 9 : Courant synaptique I_{syn} du neurone n dans le modèle m

Entrée : $E_{syn,km}$ - potentiel d'inversion de la synapse entre le modèle pré-synatique et post-synaptique
Entrée : $Fact_{nm}[k]$ - facteur du courant synaptique du neurone n du modèle m actualisé à chaque PA du modèle k
Entrée : $g_{syn,km}$ - conductance synaptique entre le modèle pré-synatique et post-synaptique
Entrée : t_C - temps courant
Entrée : t_P - temps précédent
Entrée : $\tau_{syn,km}$ - constante de temps de la synapse entre le modèle pré-synatique et post-synaptique
Entrée : V_n - potentiel de membrane d'un neurone n
Sortie : $I_{syn,n}$ - courant synaptique du neurone n

1 **pour chaque** *Modèle k* **faire**
2 **si** *Modèle k est connecté au Modèle m* **alors**
3 $Fact_{nm}[k] \leftarrow \exp\left(-\frac{t_C - t_P}{\tau_{syn,km}}\right) \times Fact_{nm}[k]$
4 $I_{syn,n} \leftarrow I_{syn,n} + g_{syn,km} \times (V_n - E_{syn,km}) \times Fact_{nm}[k]$
5 **fin**
6 **fin**

3.2.4 Actualisation des variables du système d'HH

Maintenant que les variables ont été estimées avec la méthode numérique et que les courants appliqués, et synaptiques, ont été calculés, le simulateur peut enfin calculer les dérivées du système d'équations (1.4), c'est l'étape à la ligne 5 de l'Algorithme 7. L'Algorithme 10 schématise le système d'équations (1.4) utilisé dans SiReNe. Les variables V, m, h et n du système d'équations, et les conductances ioniques, sont substituées par les valeurs du tableau `Variables`, resp. par les valeurs du tableau `Parameters`. Les fonctions d'in/activation x_∞ et τ_x sont définies par d'autres fonctions internes à SiReNe, mais qui ne seront pas précisées dans cette section. Les valeurs des dérivées sont stockées dans le tableau `DVar`.

D'autres canaux ioniques peuvent être implémentés sur SiReNe, dans cette sous-section, nous donnons juste un exemple d'une implémentation d'un modèle.

Algorithme 10 : Système d'équations d'HH

Entrée : C_m - capacité membranaire
Entrée : E_L, E_{Na}, E_K - potentiel d'inversion ionique
Entrée : g_L, g_{Na}, g_K - conductance ionique du tableau `Parameters`
Entrée : $m_\infty, h_\infty, n_\infty$ - états stables des fonctions d'in/activation
Entrée : τ_m, τ_h, τ_n - constantes de temps des fonctions d'in/activation
Entrée : Var - variables du tableau `Variables`
Sortie : dVar - dérivées de variables du tableau `DVar`

$\mathrm{dVar}[0] \leftarrow (-g_L(\mathrm{Var}[0] - E_L) - g_{Na}\mathrm{Var}[1]^3\mathrm{Var}[2](\mathrm{Var}[0] - E_{Na}) - g_K\mathrm{Var}[3]^4(\mathrm{Var}[0] - E_K)$
$\quad -I_{syn} + I_{app})/C_m$ `// Potentiel membranaire (mV/ms)`
$\mathrm{dVar}[1] \leftarrow (m_\infty - \mathrm{Var}[1])/\tau_m$ `// Gate d'activation du courant Na (1/ms)`
$\mathrm{dVar}[2] \leftarrow (h_\infty - \mathrm{Var}[2])/\tau_h$ `// Gate d'inactivation du courant Na (1/ms)`
$\mathrm{dVar}[3] \leftarrow (n_\infty - \mathrm{Var}[3])/\tau_n$ `// Gate d'activation du courant K (1/ms)`

3.2.5 Détection des PA

À présent, les PA sont détectés avec la fonction `SpikeDetection()` se trouvant dans le fichier `Spikes.c`. Habituellement, les PA sont détectés lorsque le potentiel membranaire dépasse un certain seuil. Nous avons ajouté une condition afin d'améliorer notre approche de détection. Si la dérivée au temps t, resp. $t + \Delta t$, est positive, resp. négative, et que le potentiel de membrane a dépassé le seuil `SpikeThreshold`, alors un PA est détecté. En ajoutant cette deuxième condition, nous nous assurons que le maximum du potentiel de membrane, se situe bien entre le temps précédent, t, et le temps courant, $t + \Delta t$. Dans l'Algorithme 11 décrivant la détection de PA, deux points, P_0 et P_2, sont définis à partir des deux potentiels au temps précédent et courant, $P_0 = (t_P, V_P)$ et $P_2 = (t_C, V_C)$. Le troisième point, P_1, est le point d'intersection entre les droites passant par P_0 et P_2. Le temps du PA est calculé avec la courbe de Bézier.

Algorithme 11 : Détection du PA

Entrée : Seuil - seuil de détection de PA, `SpikeThreshold`
Entrée : t_c, t_p - temps courant et temps précédent
Entrée : V_c, V_p - potentiel au temps courant et au temps précédent, se trouvant dans `Variables`
Entrée : dV_c, dV_p - dérivée du potentiel au temps courant et au temps précédent, se trouvant dans `DVar`
Sortie : t_{PA} - temps du PA

pour chaque *Neurone n* **faire**
  si $[(V_P > Seuil)$ **ou** $(V_C > Seuil)]$ **et** $(dV_P > 0)$ **et** $(dV_C < 0)$ **alors**
    $P_0 \leftarrow (t_P, V_P)$
    $P_2 \leftarrow (t_C, V_C)$
    Calculer le point P_1 à partir des points P_0 et P_2
    Calculer le temps du PA avec la courbe de Bézier à l'aide des points P_0, P_1 et P_2
    Génération des connections postsynaptiques
  fin
fin

3.2.6 Calcul du temps de PA : Courbe de Bézier

Plusieurs méthodes pour calculer le temps du PA existent et sont présentées dans le tableau 3.2, deux d'entre elles sont décrites dans les sous-sections 2.2.1 et 2.2.2. La méthode doit être précisée dans la fonction générique `Interpolator()` lors de l'initialisation. Nous détaillons ici uniquement la version la plus précise, qui est basée sur la courbe de Bézier.

Tout d'abord, il faut calculer la courbe de Bézier avec les trois points de contrôles, décrit dans l'Algorithme 12. Ensuite, il faut calculer la dérivée de la courbe et chercher le maximum. Le temps du PA est ensuite déduit.

Algorithme 12 : Calcul du temps de PA avec l'interpolation de Bézier

Entrée : P_0, P_1, P_2 - points de contrôle
Sortie : t_{PA} - temps du PA

1 Calcul de la courbe de Bézier avec les trois points de contrôle P_0, P_1 et P_2
2 Calcul de la dérivée de la courbe de Bézier en fonction du potentiel
3 Trouver le maximum du potentiel de la courbe de Bézier lorsque la dérivée est nulle
4 Calculer le temps du PA, t_{PA}

3.2.7 Génération de connectivité et actualisation du courant post-synaptique

L'approche de génération de connectivité, et l'actualisation du courant post-synaptique lors d'un PA, sont décrites dans la section 2.3 et la sous-section 2.1.2.

Génération de connectivité

Après qu'un PA a été détecté, la fonction `RandGenerConnect()` est appelée dans la fonction `SpikeProcessing` du fichier `Spikes.c`. Les connexions postsynaptiques du neurone émetteur sont générées. La sélection pseudo-aléatoire est appliquée indépendamment à chaque structure post-synaptique en utilisant le tirage pseudo-aléatoire présenté dans l'Algorithme 4. Les connexions ne sont pas stockées dans une `Liste`, comme nous pouvons le voir dans l'Algorithme 4, au contraire, le courant synaptique du neurone tiré est actualisé tout de suite afin d'éviter de stocker la connectivité dans une liste.

Actualisation du courant post-synaptique

La fonction `SynapticSpike()` actualise le courant synaptique, lors d'un PA, vu dans l'Algorithme 3 du chapitre précédent et aussi dans l'équation (2.3). Ces valeurs sont ensuite rapportées dans le tableau `**SynCurrentFactor`.

3.2.8 Résultats

Contexte expérimental

Les simulations présentées ci-dessous ont été réalisées sur `Euphrosyne`. `Euphrosyne` est un Dell R720 sous Linux Debian 4.9 amd64 avec 2 Inter(R) Xeon(R) CPU E5-2640 v2 @ 2.00 GHZ avec 8 coeurs chacun, et 128GB de RAM. Les temps ont été mesurés avec la fonction d'OpenMP, `omp_get_wtime()`, et le programme a été compilé avec `gcc` 6.3.0 et le niveau d'optimisation `-O3`.

Comparaison avec McCarthy

Dans le but de valider le fonctionnement de SiReNe, nous comparons la dynamique neuronale de nos résultats avec ceux de l'article de McCarthy et al. [72]. La dynamique neuronale peut être décrite par un raster plot des PA ou par la trace du LFP. Le raster plot des PA est une trace des temps de PA de chaque neurone durant le temps de simulation. Le LFP, quant à lui, est approché ici par la somme des courants synaptiques de chaque neurone à chaque pas de temps [72] et le spectre de puissance du LFP est déduit d'une transformée de Fourier standard. Généralement, le LFP est enregistré localement, i.e. une électrode est placée à proximité des neurones et leurs courants synaptiques sont enregistrés. Nous avons fait le choix de calculer le LFP sur l'ensemble du réseau.

Comme nous n'avons pas accès aux valeurs numériques des résultats de l'article de McCarthy et al. [72], nous comparons les comportements globales du réseau de McCarthy et al. avec SiReNe et le comportement du même modèle, mais de l'article.

Paramètres de simulation		
Paramètres généraux	Taille du réseau	1.3M
	Durée $[ms]$	4000
	# thread(s)	32
Paramètres du courant appliqué	Type du courant appliqué	escalier avec bruit
	moment où le courant change $[ms]$	500
	avec un bruit	$\times [0, 1]$
	$I_{app}\,[\mu A]$ à la 1^{ire} partie	-10
	$I_{app}\,[\mu A]$ à la 2^{ime} partie	1.19
	Amplitude du bruit	300
Paramètres synaptiques	[%] de connectivité	0.0004
	$\tau[ms]$	15
	$E_{syn}[mV]$	-80
	$g_{syn}[nS]/\#connections$	0.1-0.6

Tab. 3.8 – Liste des paramètres de simulation du réseau de neurones MSN complet.

Dans les Fig. 3.4 et 3.5, resp. 3.6 et 3.7, le raster plot, et le LFP, sont tracés en conditions saine et parkinsonienne. Afin de distinguer les conditions parkinsonienne et saine, nous diminuons la conductance du courant M lors de la condition parkinsonienne, comme l'article [72], i.e. en condition saine la conductance g_m vaut $1,34mS$ tandis qu'en parkinsonienne g_m vaut $1,1mS$, comme dans l'article de McCarthy et al [72]. Le Tab. 3.8 complète les informations sur les paramètres de simulation de cette expérience.

Dans les Fig. 3.4 et 3.5, montrant les simulations entre $2s$ et $4s$, seuls les neurones entre les index 100 000 et 100 100 sont affichés. En condition parkinsonienne, Fig. 3.5, nous notons une synchronisation pathologique, en revanche, et comme attendu, elle est moins importante en condition saine, Fig. 3.4.

La moyenne de fréquence de décharge des neurones MSN, lors de nos simulations, est de $0,94 \pm 0,63Hz$ en condition saine, en accord avec les données *in vivo* de Kish et al. [58], $1,1 \pm 0,18Hz$ et l'article [72], $0,96 \pm 0,03Hz$. Tandis que la moyenne de fréquence de décharge de notre modèle de neurones, à l'état parkinsonien, est de $3,71 \pm 1,03Hz$ et les données *in vivo* de [58] indiquent $2,11 \pm 0,43Hz$ et l'article [72] donne un taux de $4,9 \pm 0,15Hz$.

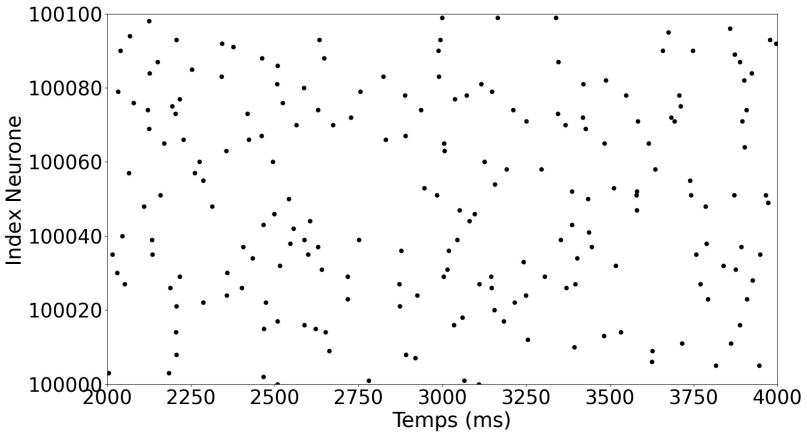

Fig. 3.4 – Tracé des temps des PA des neurones MSN, entre l'index 100 000 et 100 100, à chaque pas de temps en condition saine. Simulation du réseau MSN à l'échelle du rat, i.e. 1,3 million de neurones, pendant $4s$ de temps biologique. Tirée de l'article [13].

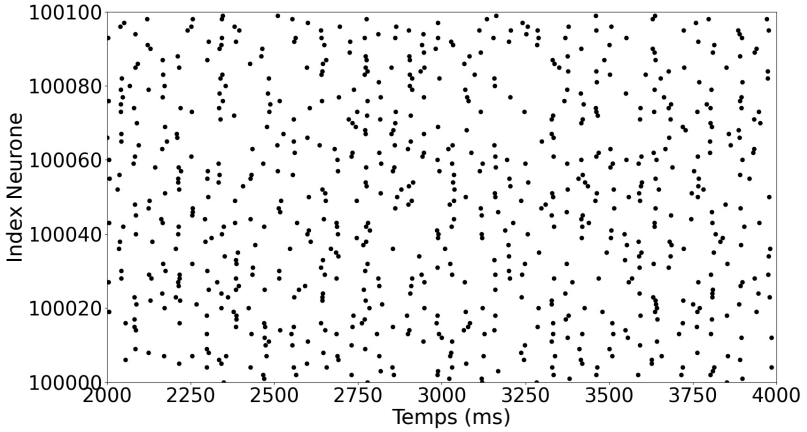

Fig. 3.5 – Tracé des temps des PA des neurones MSN, entre l'index 100 000 et 100 100, à chaque pas de temps en condition parkinsonienne. Simulation du réseau MSN à l'échelle du rat, i.e. 1,3 million de neurones, pendant $4s$ de temps biologique. Tirée de l'article [13].

Dans les Fig. 3.6 et 3.7, seuls les courants synaptiques entre $2s$ et $4s$ ont été pris en compte pour le calcul du LFP afin d'éviter de prendre en compte une éventuelle phase transitoire du réseau.

Notre modèle montre des résultats similaires à ceux rapportés par l'article [72]. Dans la Fig. 3.6, en condition saine, un petit pic est observé à maximum $30dB$ à $17Hz$, indiquant un faible état de synchronisation. Cependant, la Fig. 3.7, exposant l'état parkinsonien, atteint des oscillations de bande β plus élevées, d'environ $80dB$ à $23Hz$ représentant bien les oscillations pathologiques parkinsoniennes, i.e. des fortes synchronisations neuronales [72].

En prenant en compte les résultats obtenus, nous pouvons conclure que notre modèle de neurones, à 1,3 million de neurones, se comporte avec beaucoup de similitudes à celui de McCarthy et al. [72]. D'un côté, à l'état parkinsonien, la fréquence de décharge des MSN augmente et de l'autre, la puissance du LFP atteint la bande β élevée. Ainsi, le fonctionnement de SiReNe est validé pour simuler des réseaux de neurones et, plus précisément, des réseaux de

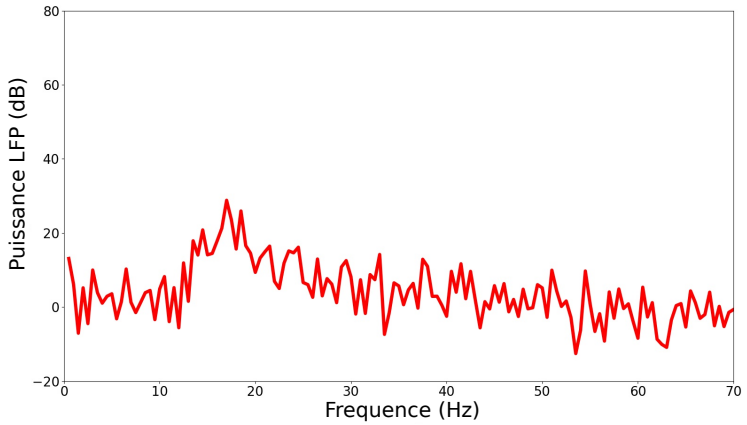

Fig. 3.6 – Puissance du LFP en fonction de la fréquence en condition saine lors de la simulation du réseau à l'échelle une du rat, i.e. 1,3 million de neurones, pendant $4s$ de temps biologique. Tirée de l'article [13].

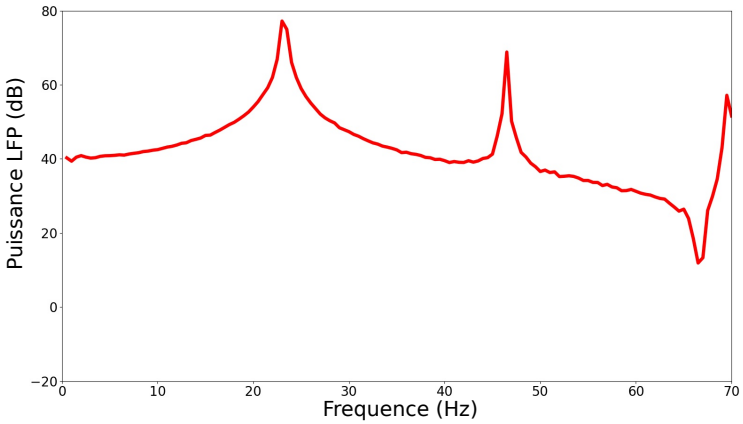

Fig. 3.7 – Puissance du LFP en fonction de la fréquence en condition parkinsonienne lors de la simulation du réseau à l'échelle une du rat, i.e. 1,3 million de neurones, pendant $4s$ de temps biologique. Tirée de l'article [13].

neurones à grande échelle, $> 10^6$ neurones dans notre cas, contre 100 neurones pour l'article [72]. Dans le prochain chapitre, le réseau complet des GB sera simulé avec le logiciel **SiReNe**.

Comparaison avec BRIAN

SiReNe est comparé avec **BRIAN** sur le benchmark de l'article [18], le modèle **COBAHH**. Le modèle **COBAHH** est un réseau de 4 000 neurones, dont 80% de neurones excitateurs et 20% de neurones inhibiteurs du type Hodgkin-Huxley, avec une connectivité complète. La dynamique du neurone est intégrée par la méthode d'Euler, avec un pas de temps $0,01ms$, et les PA sont détectés par le franchissement d'un seuil à $-20mV$ avec une période réfractaire de $3ms$ et le temps du PA est aligné au pas de temps. Ces mêmes méthodes ont été adaptées, ou implémentées, dans **SiReNe**. Le modèle a été simulé sur la même machine pour les deux logiciels et pendant une seconde de temps biologique. Les résultats numérique pour les deux simulations sont exactement les mêmes. Le temps d'exécution est de $1min$ environ pour les deux, cependant, **BRIAN** alloue

$375Mb$ tandis que **SiReNe**, alloue $18Mb$ soit 20 fois moins de mémoire.

3.3 Calcul parallèle : OpenMP et MPI

Avec la nouvelle approche de simulation, présentée dans le chapitre précédent, non seulement le stockage de mémoire a été réduit, mais aussi le temps de simulation. Néanmoins, ce dernier point pourrait être davantage amélioré par le calcul parallèle. Ainsi nous avons intégré dans le logiciel SiReNe le calcul parallèle en utilisant OpenMP et MPI. La partie MPI a été principalement implémentée par Sylvain Contassot-Vivier, mais j'ai jugé intéressant de l'intégrer à ce manuscrit afin d'avoir une vision complète des développements effectués dans le logiciel SiReNe ainsi que l'étendue de ses possibilités.

3.3.1 OpenMP

OpenMP est une interface de programmation pour le calcul parallèle multi-threads à mémoire partagée en C/C++ et Fortran. Cet outil est adaptable à beaucoup d'algorithmes et de plates-formes, e.g. ça passe de l'ordinateur de bureau aux supercalculateurs, tout en permettant de rester proche du code séquentiel (ou mono**T**hread (1T)). Les threads gérés par OpenMP peuvent être utilisés lorsqu'une série d'instructions peut être exécutée simultanément. L'idée est de répartir les tâches de calcul sur les threads disponibles afin que ces tâches soient exécutées simultanément, c'est ce que nous appelons du calcul *parallèle*. La Fig. 3.8 donne un exemple du passage d'un calcul séquentiel à un calcul parallèle.

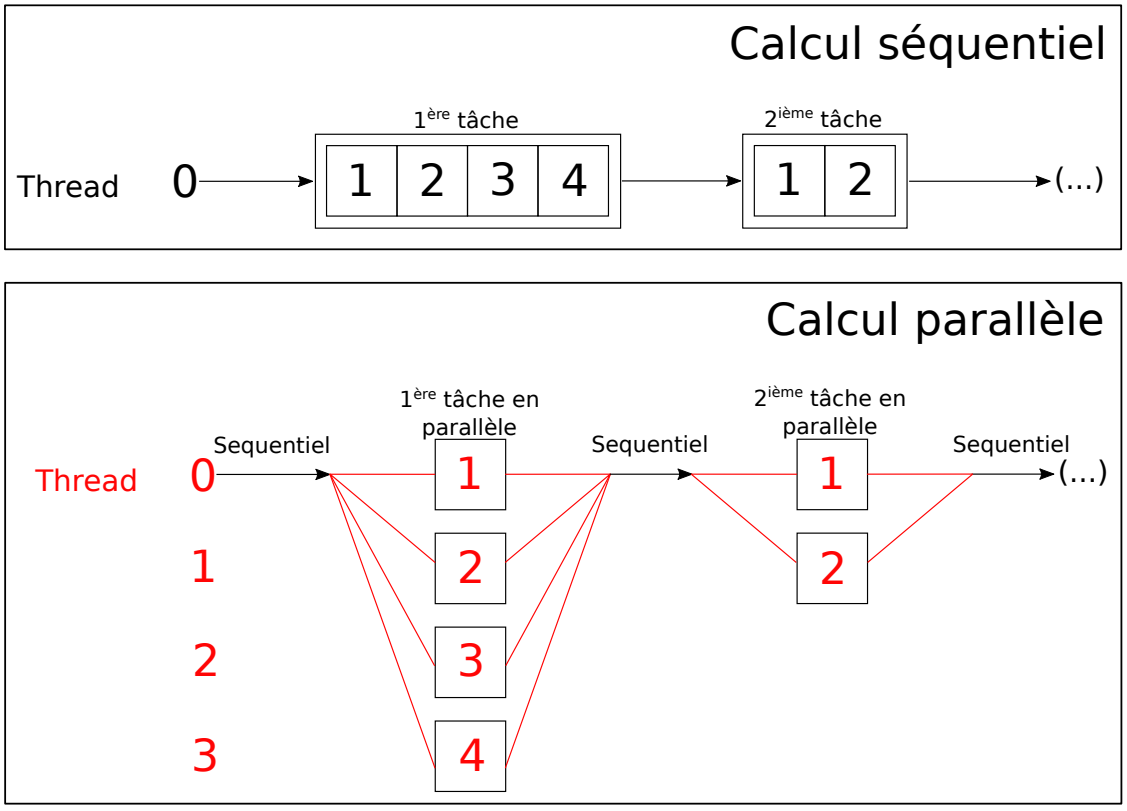

Fig. 3.8 – Comparaison du calcul séquentiel avec le calcul parallèle.

Dans SiReNe, les différents calculs sur la dynamique des neurones et la détection de PA sont éligibles au calcul parallèle. En effet, les calculs de la dynamique des neurones sont indépendants des autres neurones, i.e. les calculs sont locaux aux neurones. Il n'y a donc aucun risque de chevauchement de calcul lors des modifications des variables entre les différents threads. Dans notre cas, le MT est donc efficace lorsque nous parcourons tous les neurones. Ces cas surviennent lorsque nous calculons la méthode numérique, où toutes les variables de tout le réseau sont parcourues, voir l'Algorithme 6, ou aussi lorsque nous faisons la mise à jour du système d'équations, voir l'Algorithme 7 ou pour finir, lorsque nous essayons de détecter des PA, voir l'Algorithme 11. D'ailleurs, comme la génération aléatoire est effectuée à cette étape, le générateur vu à la section 2.3, a dû être adapté au MT. La Fig. 3.9 montre la présence de calculs parallèles sur ces différentes étapes.

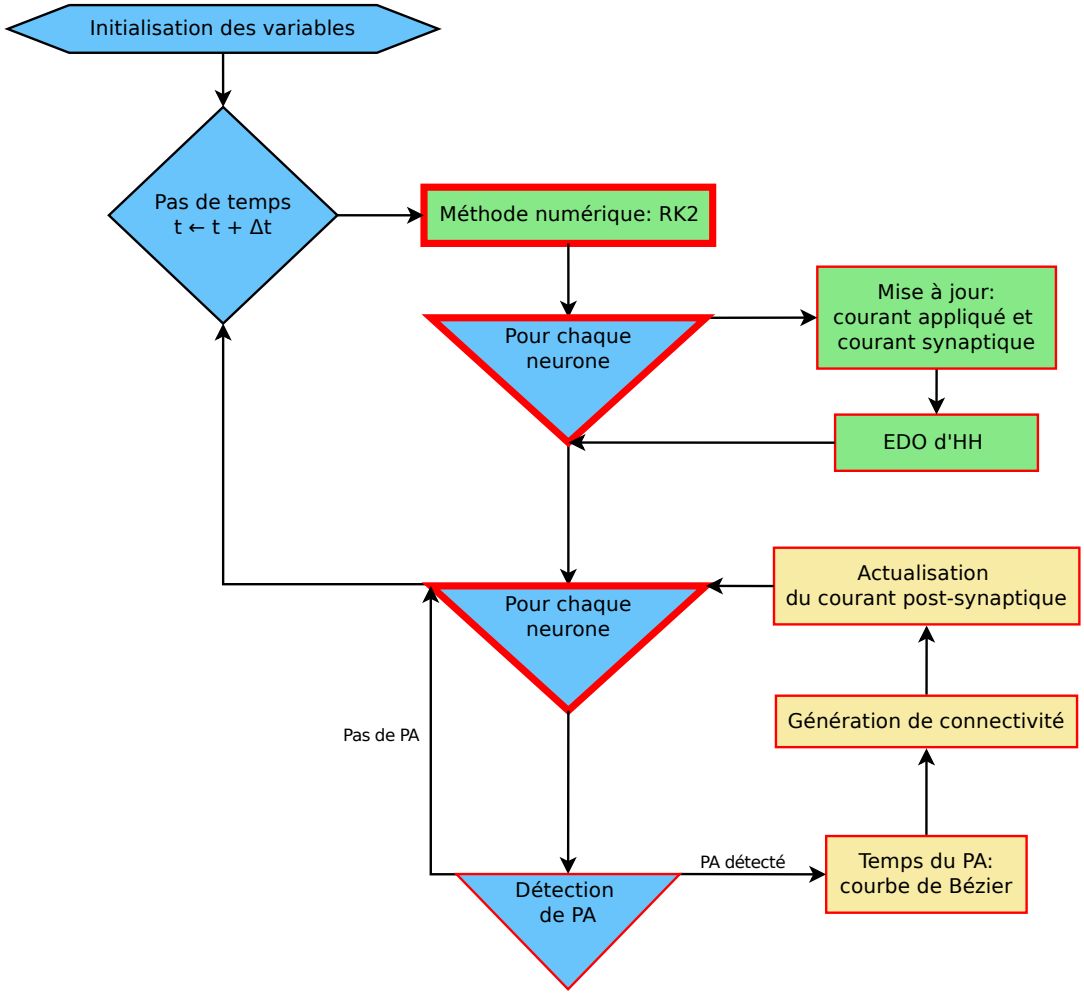

Fig. 3.9 – SiReNe schématisé avec les calculs parallèles entourés en rouge. Les grosses bordures signifient qu'à cet instant de l'exécution, les tâches sont distribuées aux différents threads, e.g. si 200 neurones sont simulés et que 10 threads sont utilisés, alors le système attribue 20 neurones à chaque thread. Concernant la bordure fine rouge, cette étape est maintenant effectuée simultanément par les différents threads. À l'étape «Méthode numérique», la distribution des tâches et le lancement en MT s'effectuent à l'intérieur du processus «Méthode numérique».

En OpenMP, une seule machine est exploitée. Cependant, pour aller encore plus vite, nous

pourrions lancer le calcul parallèle sur plusieurs machines et ainsi construire une méthode **M**ulti**T**hreading et **M**ulti**M**achines (MTMM) pouvant s'exécuter sur une grappe de machines (cluster [2]) en utilisant à la fois OpenMP et MPI. Ainsi, sur chaque nœud individuel, le calcul parallèle se réalise avec OpenMP, tandis qu'entre les nœuds, le calcul parallèle s'opère avec MPI pour échanger les données via des communications explicites.

Adaptation du code

L'actualisation du courant postsynaptique du neurone émetteur est une partie du code en MT. Par conséquent, le tableau `SynCurrentFact` est modifié par plusieurs threads au même moment. Nous aurions pu utiliser une synchronisation mutuelle des threads afin que chaque thread puisse avoir accès au tableau l'un après l'autre, sauf que cette solution est très coûteuse en temps. Ainsi, nous avons décidé de dupliquer le tableau en mémoire par le nombre de threads, i.e. chaque thread a accès à sa version du tableau. Ensuite, lorsque nous calculons le courant synaptique interne au neurone, nous faisons la somme sur tous les threads du tableau `SynCurrentFact`, comme nous pouvons le voir dans l'Algorithme 13.

Algorithme 13 : Courant synaptique I_{syn} du neurone n dans le modèle m en MT

Entrée : $E_{syn,km}$ - potentiel d'inversion de la synapse entre le modèle pré-synatique et post-synaptique
Entrée : $Fact_{nm}[k][t]$ - facteur du courant synaptique du neurone n du modèle m actualisé à chaque PA du modèle k
Entrée : $g_{syn,km}$ - conductance synaptique entre le modèle pré-synatique et post-synaptique
Entrée : t_C - temps courant
Entrée : t_P - temps précédent
Entrée : $\tau_{syn,km}$ - constante de temps de la synapse entre le modèle pré-synatique et post-synaptique
Entrée : V_n - potentiel de membrane d'un neurone n
Sortie : $I_{syn,n}$ - courant synaptique du neurone n

1 $Somme \leftarrow 0$ **pour chaque** *Modèle k* **faire**
2 **si** *Modèle k est connecté au Modèle m* **alors**
3 **pour chaque** *Thread t* **faire**
4 $Fact_{nm}[k][t] \leftarrow Fact_{nm}[k][t] * \exp\left(-\frac{t_C - t_P}{\tau_{syn,km}}\right)$
 $Somme \leftarrow Somme + Fact_{nm}[k][t]$
5 **fin**
6 $I_{syn,n} \leftarrow g_{syn,km} \times (V_n - E_{syn,km}) \times Somme$
7 **fin**
8 **fin**

3.3.2 MPI

MPI est une interface de programmation pour le calcul parallèle par passage de messages entre ordinateurs distants ou ordinateur à multiprocesseur, en C/C++ et Fortran. MPI permet

2. Une grappe de machines (cluster) est un ensemble de plusieurs ordinateurs indépendants, aussi appelés nœuds.

de distribuer des calculs/tâches à différents ordinateurs sans partage de mémoire. Lorsqu'il faut partager les calculs ou tâches avec les autres ordinateurs, alors le passage de messages entre les machines est nécessaire. Cette étape peut-être très coûteuse en temps si elle est exécutée de nombreuses fois pendant une simulation et selon la quantité de données échangées.

Nous avons développé deux stratégies différentes de simulation en MTMM. Cependant seulement une seule sera exposée dans ce manuscrit, car la deuxième est encore en cours de validation. La difficulté majeure, lorsqu'on utilise plusieurs machines, est que la solution à une équation d'un neurone doit être envoyée à tous les neurones auxquels il est connecté, e.g. quand le courant synaptique lors d'un PA est calculé, il faut transmettre l'actualisation du courant aux autres neurones des autres machines. Ces communications entre machines sont très coûteuses en temps. La nature du problème est donc plutôt défavorable à l'utilisation du calcul distribué (multi-machines), car il implique nécessairement une quantité importante d'échanges d'informations entre les machines. Le gain potentiel de performance que nous pouvons obtenir en utilisant plusieurs machines est donc plus limité que ce à quoi nous pourrions nous attendre. Cependant, cette solution reste potentiellement intéressante pour réduire sensiblement les temps de calcul et présente également l'avantage de permettre le traitement de réseaux neuronaux de très grande taille, pour lesquels la mémoire d'une seule machine ne serait pas suffisante.

Deux nouvelles structures ont dû être introduites afin d'implémenter MPI dans SiReNe. La structure `NodeData` définit la distribution des données dans un nœud. Les variables fixées dans cette structure correspondent à un nœud. `FirstNeuron` est l'indice global du premier neurone. `FirstVariable` est l'indice de la première variable du premier neurone. `FirstModel` est l'indice du premier modèle. `FirstNeuronInModel` est l'indice du premier neurone du premier modèle. `LastModel` est l'index du dernier modèle. `NbNeurons` est le nombre de neurones. Cette structure est très utile lorsqu'une boucle sur les neurones, resp. les variables, est réalisée, e.g. dans la fonction `NumericalMethods` ou autres, voir la Fig. 3.9.

La structure `DataSubSet` définit l'intervalle de variables sauvegardées par modèle et est utile pour la sauvegarde des données dans des fichiers. Deux variables y sont spécifiées, l'index global de la première variable du modèle gérée par le nœud courant, `GlobalIndexFirstVariables`, et le nombre de neurones de ce modèle géré par le nœud courant, `NbNeurons`.

Dans SiReNe, le nombre global des neurones est découpé selon le nombre de machines, i.e. chaque machine gère un bloc de neurones. La Fig. 3.10 illustre les étapes de simulation en MPI.

1. Le courant postsynaptique du neurone émetteur, lors d'un PA, est calculé par chaque machine en fonction de son bloc de neurones.
2. Ensuite, les courants synaptiques sont ajoutés en fonction des différents threads dans un tableau unique.
3. Chaque machine envoie ensuite les différentes sous-parties aux machines gérant ces blocs.
4. Les machines réceptionnent les courants synaptiques gérés localement par les autres machines et font une somme dans le tableau du bloc géré localement.

3.3.3 Résultats : Performances du calcul parallèle

Dans le but de comparer les performances du calcul parallèle sur **SiReNe**, la consommation mémoire et le temps de simulation sont évalués en fonction du nombre de threads avec l'approche MJE-G (**M**ise à **J**our à chaque **É**vénement avec **G**énération de connectivité).

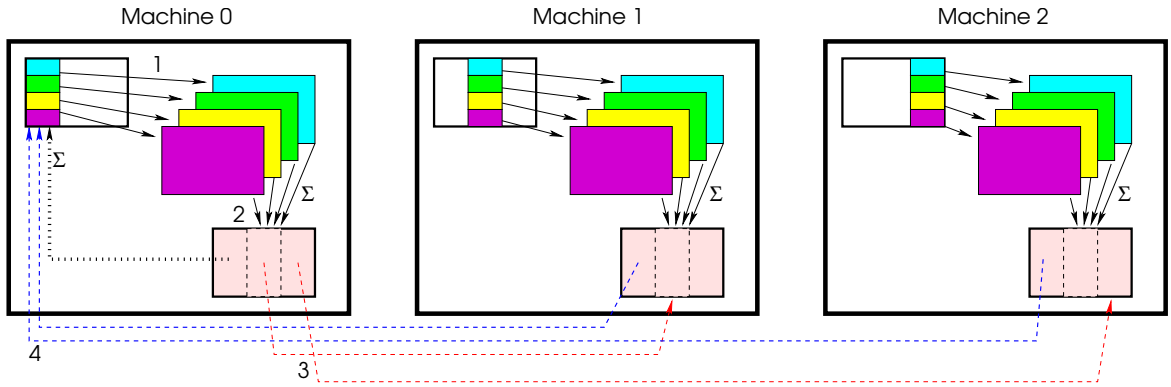

Fig. 3.10 – Schéma de distribution des tâches de SiReNe en MPI. Les couleurs représentent des données (neurones ou courants synaptiques) traitées par des threads différents.

Résultats en OpenMP

Contexte expérimental Les simulations présentées ci-dessous ont été réalisées sur `Euphrosyne`. `Euphrosyne` est un Dell R720 sous Linux Debian 4.9 amd64 avec 2 Inter(R) Xeon(R) CPU E5-2640 v2 @ 2.00 GHZ avec 8 coeurs chacun, et 128GB de RAM. Les temps ont été mesurés avec la fonction `omp_get_wtime()` d'OpenMP et le programme a été compilé avec `gcc` 6.3.0 et le niveau d'optimisation `-O3`.

Plusieurs simulations avec 10 000 neurones MSN, et 504 connexions pour chaque neurone, sont effectuées pendant une minute de temps biologique afin de comparer les performances du simulateur en OpenMP. Le temps de simulation et la mémoire allouée sont comparés en fonction du nombre de threads utilisés. La liste complète des paramètres de simulation de cette expérience est donnée dans le Tab. 3.9.

Paramètres de simulation		
Paramètres généraux	Taille du réseau	10k
	Temps biologique $[ms]$	1000
	# thread(s)	1-32
Paramètres du courant appliqué	Type du courant appliqué	échelon avec bruit
	moment où le courant change $[ms]$	100
	avec un bruit	$\times\,[0,1]$
	$I_{app}\,[\mu A]$ à la 1^{ire} partie	-10
	$I_{app}\,[\mu A]$ à la 2^{ime} partie	0.4
	Amplitude du bruit	300
Paramètres synaptiques	[%] de connectivité	0.05
	$\tau[ms]$	12
	$E_{syn}[mV]$	-80
	$g_{syn}[nS]/\#connections$	0.25/# connections

Tab. 3.9 – Liste des paramètres de simulation analysant la performance du simulateur **SiReNe**.

Dans la Fig. 3.11, nous observons que la consommation de mémoire augmente linéairement en fonction du nombre de threads avec une pente de 0,08 MB/thread. La différence entre les valeurs les plus distantes est de moins de $3Mb$, i.e. moins de 12% de la consommation initiale.

Le facteur du courant synaptique de l'équation (2.3) est dupliqué sur tous les threads afin que les résultats, entre la version 1T et MT, soient les mêmes.

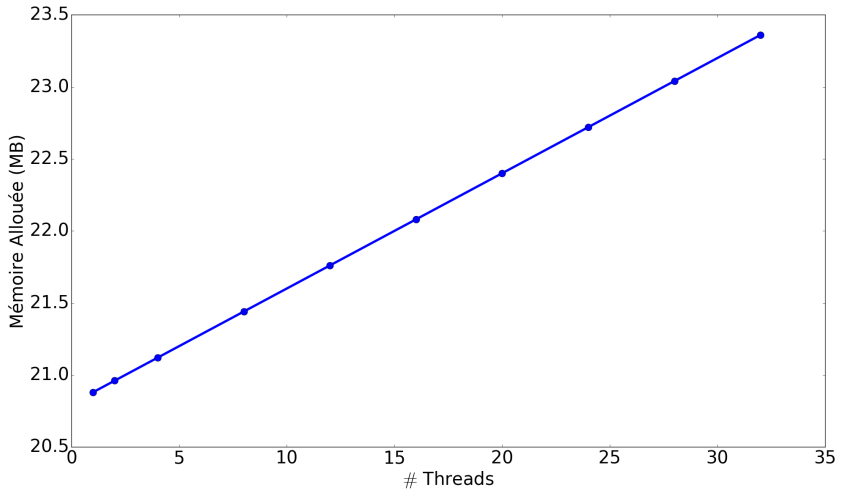

Fig. 3.11 – Allocation de mémoire en fonction du nombre de threads avec OpenMP pour la simulation d'un réseau MSN avec 10 000 neurones, et 504 connexions entrantes par neurone, pendant $1s$ de temps biologique. Tiré de l'article [13].

La machine utilisée dispose de 16 cœurs réels et de 16 cœurs virtuels. Les cœurs virtuels améliorent les performances, mais moins que les cœurs réels. La Fig. 3.12 montre que le temps de simulation diminue significativement en fonction du nombre de threads employés, i.e. plus il y a de threads exploités plus le temps d'exécution est réduit. Entre 2 et 12 threads, le temps de simulation diminue fortement, mais à partir de 16 threads, la baisse est moins importante. En effet, les cœurs virtuels apportent un gain de performance limité pour ce type de problème. Le temps de simulation diminue proportionnellement à $\frac{t_m}{\#Threads}$ et plus précisément à $1,5+\frac{t_m}{\#Threads}$.

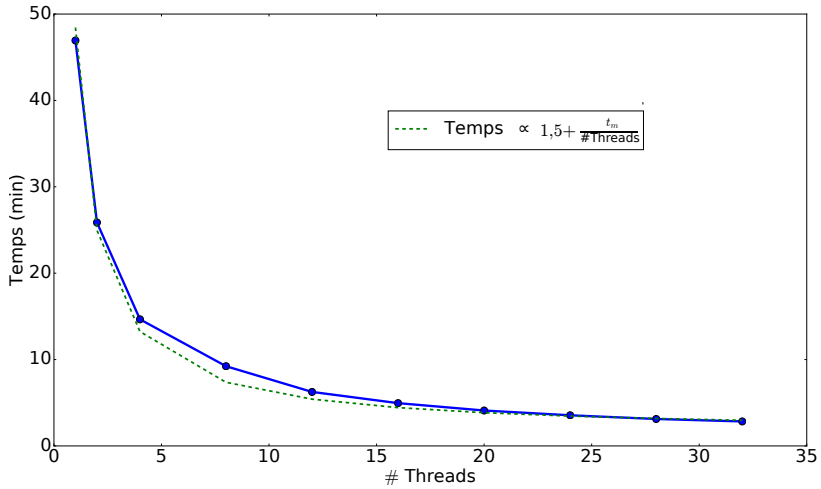

Fig. 3.12 – Temps d'exécution en fonction du nombre de threads avec OpenMP pour la simulation d'un réseau MSN avec 10 000 neurones, et 504 connexions entrantes par neurone, pendant $1s$ de temps biologique. t_m : le temps en monothread. Tiré de l'article [13].

L'accélération d'une simulation, notée a, est calculée en fonction du temps de la simulation en

1T et en MT, i.e. le temps de simulation en mono-thread t_m est divisé par le temps de simulation en MT t_i avec $i \in \{2, 4, 8, 12, 16, 20, 24, 28, 32\}$, ainsi $a = \frac{t_m}{t_i}$. L'efficacité d'une simulation, notée e, est calculée en fonction de l'accélération et du nombre de threads, i.e. l'accélération a est divisée par le nombre de threads nbT, ainsi $e = \frac{a}{nbT}$.

La Fig. 3.13 montre que l'accélération est quasi linéaire en fonction du nombre de threads. Nous observons également une décroissance quasiment régulière de l'efficacité, avec un léger artefact à 8 coeurs.

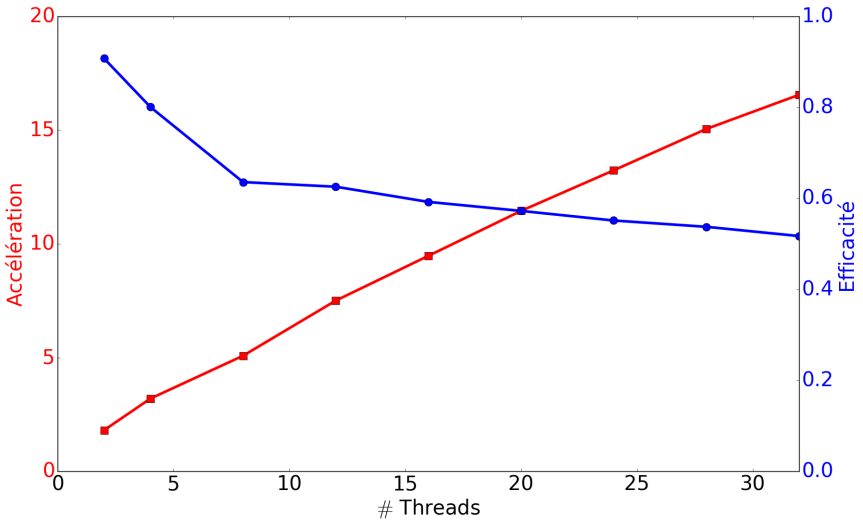

Fig. 3.13 – Accélération et efficacité de la simulation en fonction du nombre de threads avec OpenMP, pour 10 000 neurones MSN avec 504 connexions entrantes en 1s de temps biologique. Tiré de l'article [13].

Résultats en OpenMP/MPI

Contexte expérimental Les simulations présentées ci-dessous ont été réalisées sur `Grid'5000` sur le site de Nancy avec le cluster `grvingt`. `grvingt` est sous Linux Debian 4.19 amd64 avec 2 CPUs Intel Xeon Gold 6130 avec 16 coeurs chacun, et 192GB de RAM. Les temps ont été mesurés avec la fonction `omp_get_wtime()` d'OpenMP et le programme a été compilé avec `gcc` 8.3.0 et le niveau d'optimisation `-O3`.

Les simulations ont été exécutées" sur 1 à 8 machines avec 1 à 16 threads par machine afin de n'utiliser que les cœurs réels des machines. La configuration des paramètres de simulation de cette expérience est la même que précédemment, voir Tab. 3.9.

Version MTMM : 10 000 neurones MSN

Dans la Fig. 3.14, la consommation de mémoire augmente très lentement en fonction du nombre de threads, en restant très raisonnable, comme observé sur les résultats précédents.

La Fig. 3.15 montre que le temps de simulation diminue significativement en fonction du nombre de processus employés.

Le nombre de processus est le produit du nombre de machines par le nombre de threads. En utilisant 2 machines avec 2 threads ou 4 machines avec 1 thread, le temps de simulation ne varie pas fortement. Cependant, plus nous utilisons de machines avec des threads, moins le temps de

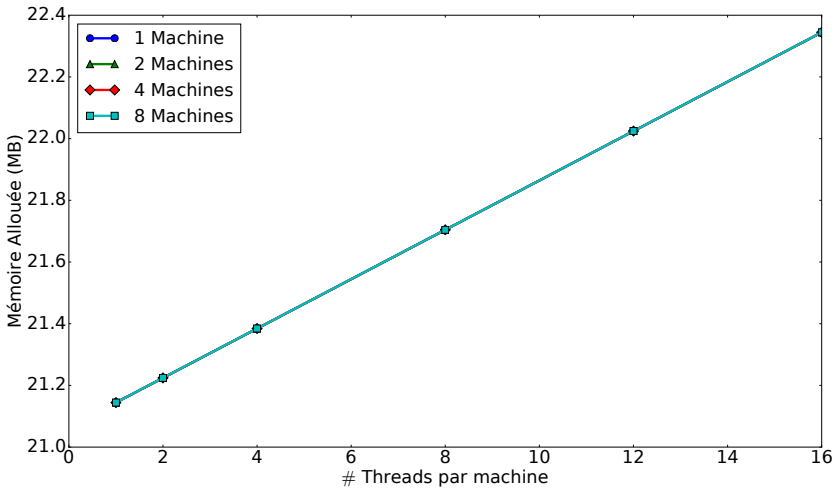

Fig. 3.14 – Allocation de mémoire en fonction du nombre de threads avec la méthode MTMM pour la simulation d'un réseau MSN avec 10 000 neurones, et 504 connexions entrantes par neurone, pendant 1s de temps biologique. Inspiré de l'article [13].

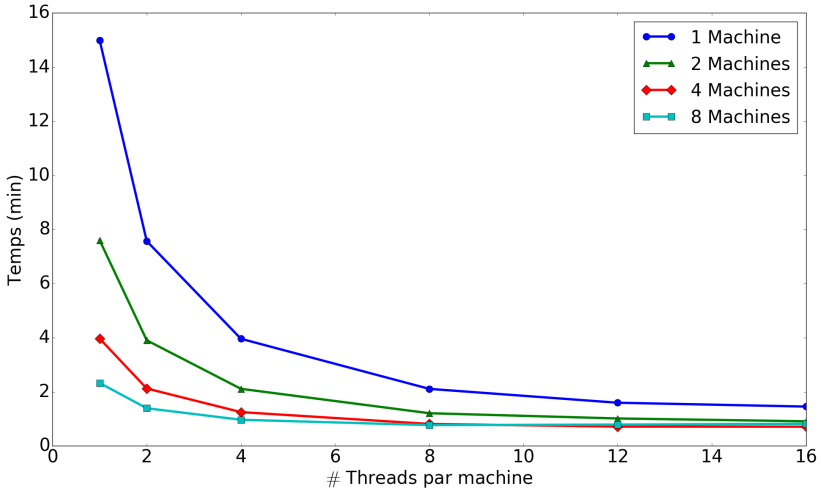

Fig. 3.15 – Temps d'exécution en fonction du nombre de threads avec la méthode MTMM pour la simulation d'un réseau MSN avec 10 000 neurones, et 504 connexions entrantes par neurone, pendant 1s de temps biologique.

simulation diminue en fonction du nombre de threads. Notamment avec 8 machines, le temps de simulation à partir de 8 threads est légèrement plus haut que celui avec 4 machines et 16 threads. Nous concluons que le calcul parallèle avec 8 machines et plus de 8 threads est inutile pour 10 000 neurones, car il ne permet pas de réduire le temps de simulation. Dans ce cas, le nombre de neurones traité par thread est d'environ 160 neurones, ce qui n'est pas très élevé, et nous pensons que, dans ce cas précis, les communications entre les machines deviennent trop importantes par rapport aux calculs effectués par les threads. Par conséquent, le calcul parallèle MTMM n'est pas intéressant si le nombre de neurones par thread n'est pas suffisamment élevé.

La Fig. 3.16 montre une accélération très modérée. D'ailleurs, en utilisant 8 machines l'accélération est même moins forte qu'avec 4 machines aussi observée précédemment.

L'efficacité du calcul parallèle chute très rapidement comme nous pouvons le voir en Fig. 3.17,

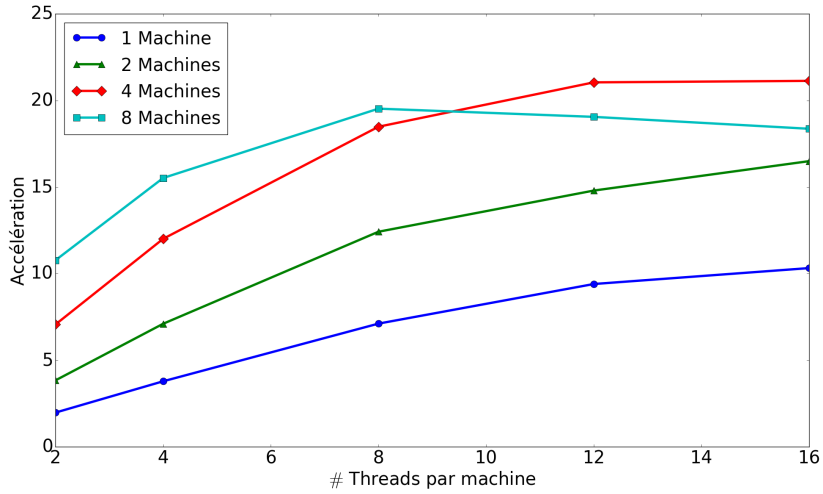

Fig. 3.16 – Accélération de la simulation en fonction du nombre de threads avec la méthode MTMM de 10 000 neurones MSN avec 504 connexions entrantes en 1s de temps biologique.

montrant encore une fois qu'il n'est pas intéressant d'utiliser cette méthode MTMM lorsqu'il y a trop peu de neurones par thread.

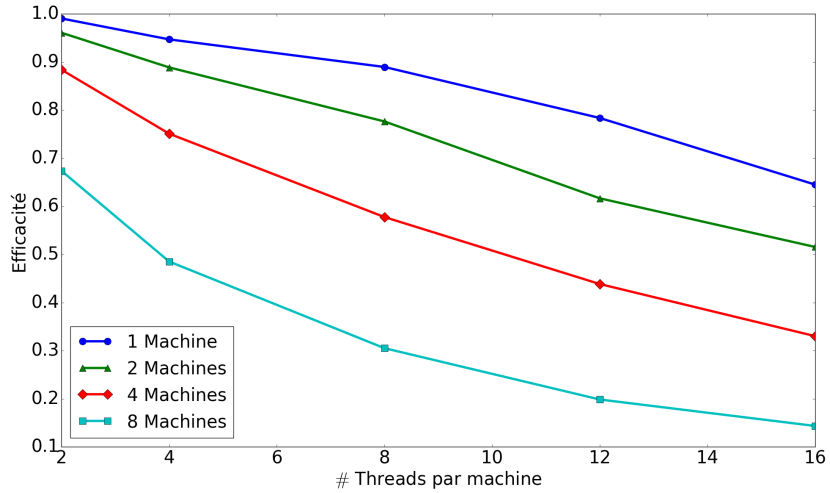

Fig. 3.17 – Efficacité de la simulation en fonction du nombre de threads avec la méthode MTMM de 10 000 neurones MSN avec 504 connexions entrantes en 1s de temps biologique.

Version MTMM : 100 000 neurones MSN

Dans la Fig. 3.18, la consommation de mémoire augmente lentement en fonction du nombre de threads, en restant très raisonnable vu que nous simulons 100 000 neurones. Entre la simulation de 10 000 neurones et celle de 100 000 neurones, l'allocation de mémoire est 3 fois plus important pour les 100 000 neurones. Cette augmentation reste tout à fait raisonnable puisque nous ne sauvegardons pas les connexions synaptiques.

La Fig. 3.19 montre que le temps de simulation diminue significativement en fonction du nombre de processus employés. Cependant, plus nous utilisons de machines avec des threads,

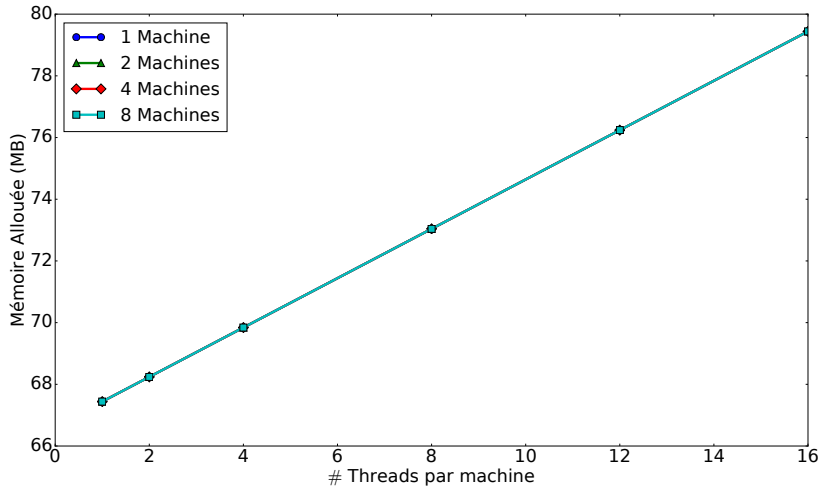

Fig. 3.18 – Allocation de mémoire en fonction du nombre de threads avec la méthode MTMM pour la simulation d'un réseau MSN avec 100 000 neurones, et 504 connexions entrantes par neurone, pendant 1s de temps biologique. Inspiré de l'article [13].

moins le temps de simulation diminue en fonction du nombre de threads. Par contre, nous n'observons plus l'augmentation de temps à partir de 8 machines et 8 threads comme dans les résultats précédents.

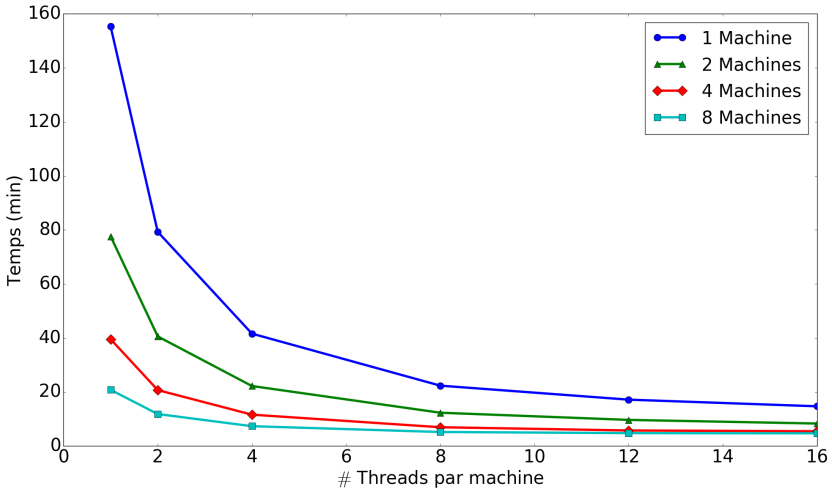

Fig. 3.19 – Temps d'exécution en fonction du nombre de threads avec la méthode MTMM pour la simulation d'un réseau MSN avec 100 000 neurones, et 504 connexions entrantes par neurone, pendant 1s de temps biologique.

La Fig. 3.20 montre une accélération très modérée. Cependant lorsque plusieurs machines sont utilisées, l'accélération est plus importante que celle observée dans le cas précédent. Ces résultats confirment que la méthode MTMM donne de meilleures performances lorsque la quantité de neurones par thread est plus importante.

L'efficacité de simulation avec le calcul parallèle chute très rapidement et plus particulièrement lorsque 8 machines sont utilisées. Avec cette courbe d'efficacité, nous comprenons que cette première version MTMM permet de traiter des réseaux neuronaux de grande taille, mais reste

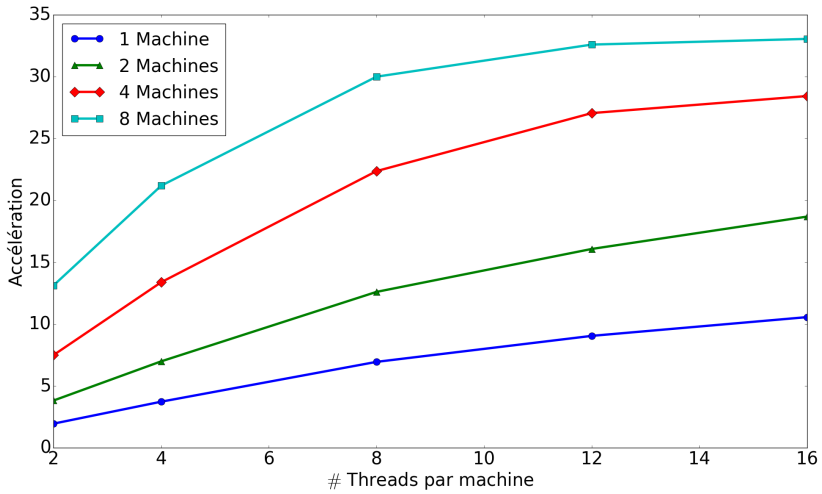

Fig. 3.20 – Accélération de la simulation en fonction du nombre de threads avec la méthode MTMM de 100 000 neurones MSN avec 504 connexions entrantes en 1s de temps biologique.

limitée en termes de performances, principalement à cause des communications importantes entre les machines.

D'autres simulations utilisant MTMM ont été effectué sur le réseau complet, voir l'annexe B.

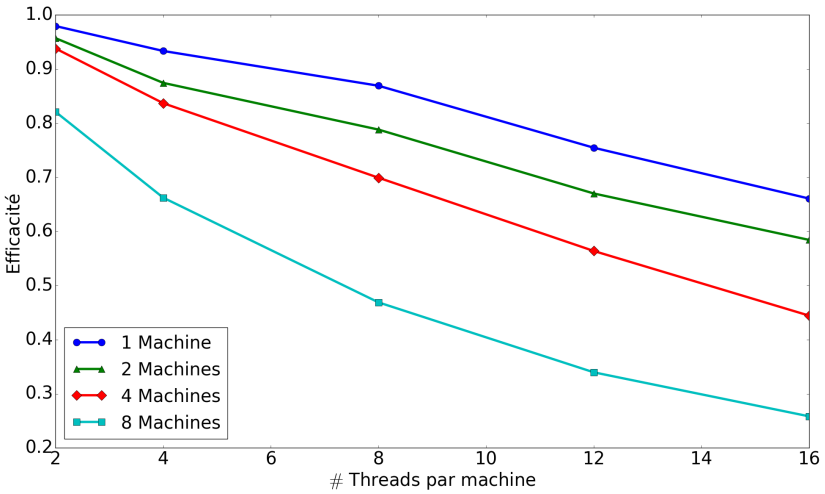

Fig. 3.21 – Efficacité de la simulation en fonction du nombre de threads avec la méthode MTMM de 100 000 neurones MSN avec 504 connexions entrantes en 1s de temps biologique.

3.4 Conclusion du chapitre

Le développement de SiReNe nous a permis de travailler sur une approche hybride, à pas de temps et événementielle, présentée dans le chapitre précédent. Ainsi, nous sommes maintenant capables de simuler des gros réseaux de neurones de type HH sans que l'allocation de la mémoire augmente trop rapidement. Nous avons réussi à mutualiser plusieurs paramètres que ce soit au niveau des neurones ou des synapses afin d'optimiser la mémoire allouée. Notre simulateur a ensuite pu être validé par un modèle publié par McCarthy et al. [72]. Une version OpenMP

fonctionnant correctement est disponible. Une première version multi-machines a également été développée, mais elle montre des limitations en terme d'efficacité, principalement dues aux importantes communications nécessaires entre les machines. Dans le chapitre suivant, le réseau complet des GB va être simulé avec notre simulateur SiReNe.

4

Modélisation et simulation des neurones prototypiques et arkypallidaux du GPe

Sommaire

Le globus pallidus extern, signifiant en latin «globe pâle» en lien avec son aspect, est un sous-ensemble des GB. Il se projette et inhibe le STN, le STR, le GPi et aussi la SN. De ce fait, il n'envoie pas d'axones inhibiteurs sur le thalamus. Cependant il est le principal régulateur du système des GB.

Dans la MP, les oscillations pathologiques, causant la rigidité et la bradykinésie, sont souvent associées au cortex ou à la boucle STN-GPe [84]. Les neurones du GPe se trouvent dans une position centrale et sont connectés à une majorité des noyaux des GB [70]. Ainsi ils peuvent

propager et synchroniser l'activité des GB. Dans l'article [70], Mallet et al. supposent que les neurones du GPe créent des synchronisations pathologiques lors de la MP qui par la suite se propagent sur le réseau des GB.

Au niveau cellulaire, nous avons fait le choix de distinguer les neurones du GPeP et du GPeA, puisque les caractéristiques cellulaires et les spécificités de connectivité sont différentes pour ces deux populations de neurones. Le GPe se projette sur deux structures bien distinctes, le STN pour le GPeP et le STR pour le GPeA et les deux structures du GPe présentent des fréquences de décharge différentes qui reflètent des propriétés cellulaires distinctes. Cette fréquence est 2 à 3 fois plus élevée pour le GPeP que le GPeA [7, 49].

Dans l'article [27], De la Crompe et al. montrent que l'activité du STN en condition saine et parkinsonienne est contrôlée par les neurones du GPeP. Ils supposent que la diminution de fréquence de décharge du GPeP dépend de l'inhibition du STR, puisque le STR est plus actif en condition parkinsonienne, ce qui aurait une influence directe sur l'hyperactivité du STN. Ces études nous montrent l'importance de faire la distinction entre les deux populations de neurones en condition saine et parkinsonienne.

Notre modèle du GPe se base sur le travail de modélisation de Fujita et al. [39] que nous avons simplifié (passage de 10 à 6 canaux ioniques) et où nous avons modifié certaines valeurs de canaux ioniques afin d'adapter notre modèle aux données expérimentales. Contrairement à Fujita et al., nous avons fait la distinction entre GPeA et GPeP. Les paramètres ont été ajustés à l'aide d'une approche d'optimisation multi-objectifs regroupant différents critères. L'approche utilisée est l'évolution différentielle qui est, à l'instar des algorithmes génétiques, un algorithme évolutionnaire. Toutes les simulations ont été réalisées avec le logiciel SiReNe.

Ce chapitre se compose de quatre sections principales. La première section illustre les premiers résultats biologiques sur la distinction des deux populations du GPe. La deuxième section décrit la modélisation du GPeA et du GPeP par le formalisme d'HH. La troisième section porte sur l'ajustement de paramètres aux données biologiques par une optimisation avec l'algorithme d'évolution différentielle et la dernière section présente les résultats de simulation de ces deux populations du GPe.

4.1 Résultats biologiques

En 2008, Mallet et al. avec l'article [70] mentionnent pour la première fois l'existence des deux populations de neurones du GPe avec la dénomination **GPe** de **T**ype **I** (GPe-TI ou GPeP) et **GPe** de **T**ype **A** (GPe-TA ou GPeA). Chez les rats affectés, 72% des neurones ont été classés GPe-TI, 17% ont été classés GPe-TA et 12% n'ont pas pu être classés.

Les deux types de populations déchargent à deux moments différents [70]. Les neurones du GPeP sont en antiphase avec le STN qu'ils inhibent. Par contre, les neurones du GPeA sont en phase avec le STN et inhibent le STR. Nous pouvons donc conclure que le GPeP et le GPeA émettent en antiphase. Les deux populations de neurones jouent certainement des rôles complémentaires dans les GB.

Abdi et al.[7] ont étudié l'électrophysiologie des deux populations et montrent que la fréquence de décharge des neurones du GPeP est plus forte (environ $21Hz$), et plus régulière que celle du GPeA (environ $6Hz$), voir Fig. 4.1. La fréquence de décharge des neurones du GPe dépend des canaux canaux sodiques "**p**ersistants", en anglais *persistent sodium channels* (NaP), mais aussi d'autres canaux comme des canaux activés par l'hyperpolarisation, modulés par les nucléotides cyclides et non sélectifs aux cations, en anglais ***H**yperpolarisation-actived **C**yclic-**N**ucleotid modulated cation non-selective channels* (HCN), des canaux potassiques à faible conductance activés

par le calcium, en anglais ***S**mall conductance calcium-activated **potassium** channels* (SK), (des canaux calciques à haut seuil d'activation, en anglais ***H**igh-**v**oltage-activated **calcium** channels* (Ca_{vH})), et des canaux K^+ dépendant du voltage, en anglais *voltage-gated K^+ channels* (K^+). D'ailleurs, d'après Schwab et al.[84], les propriétés intrinsèques des neurones du GPe sont déterminées par plus de 10 canaux ioniques différents, mais l'activité des neurones du GPe est plus dépendante des courants HCN, SK, canaux transitoires rapides de sodium, en anglais ***f**ast transient sodium channels* (NaF). Nous décrivons ci-après le rôle de chacun de ces courants dans la génération du signal.

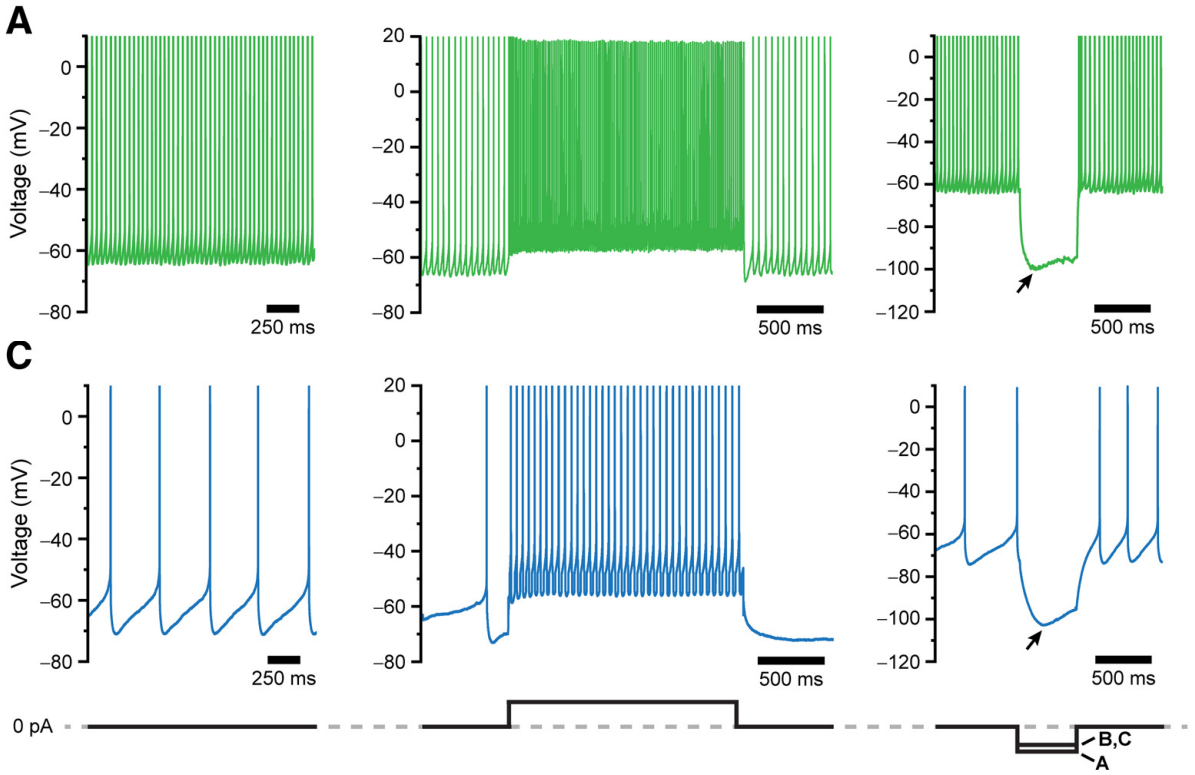

Fig. 4.1 – Potentiels membranaires des neurones du GPeA et GPeP *in vitro*. **A,** Activité d'un neurone du GPeP enregistré dans une tranche du cerveau d'un rat juvénile sain. $1^{ière}$ colonne : Potentiel sans courant injecté, $2^{ième}$ colonne : avec $100pA$ de courant injecté pendant $2s$ et $3^{ième}$ colonne : réponse à une impulsion de courant hyperpolarisante de $-100pA$ ou $-80pA$ pendant $500ms$, provoquant des dépassements de l'hyperpolarisation d'environ $-100mV$ appelé «sag» (signalé par une flèche). **C,** Même expérimentation pour un neurone du GPeA. Ces résultats sont tirés de l'article de Abdi et al. [7].

4.1.1 Courant NaP

Dans l'article [7], Abdi et al. montrent que les neurones du GPeP ont un taux de décharge plus élevé que ceux du GPeA, voir Fig. 4.1, puisqu'ils ont une densité de canaux sodiques plus élevée que les neurones du GPeA. Ainsi, ils ont réalisé une expérience dans laquelle ils ont isolé le courant des canaux NaP et ont stimulé les neurones par une rampe de potentiel lentement dépolarisante, voir Fig. 4.2 schémas du haut, pour ensuite, mesurer le courant NaP des deux types de neurones, voir Fig. 4.2 schémas du bas. Ils ont conclu que les neurones du GPeP ont un

courant presque deux fois plus important (environ $224pA$) que celui du GPeA (environ $102pA$), voir Fig. 4.2 schémas du bas.

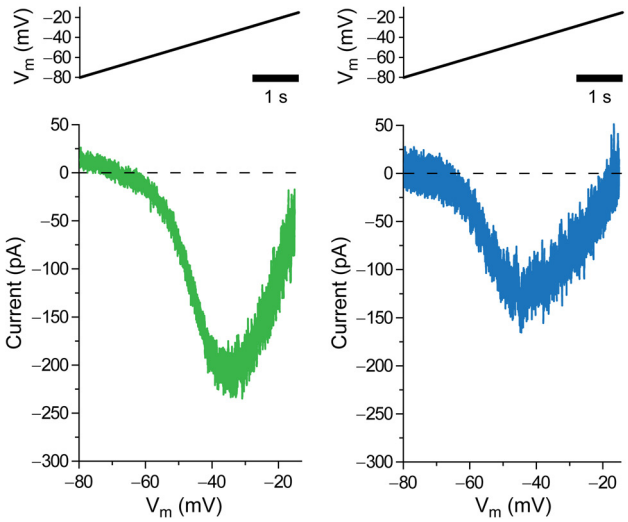

Fig. 4.2 – Amplitudes des courants NaP des neurones du GPeA et GPeP. **A,B,** Enregistrements *in vitro* représentatifs des courants persistants des canaux NaP dans un neurone du GPeP **(A)** et un neurone du GPeA **(B)** lors d'injections somatiques d'une rampe de potentiel (ligne noire dans le schéma du haut). V_m, potentiel membranaire. Ces résultats sont tirés de l'article de Abdi et al. [7].

4.1.2 Courant HCN

Un courant d'hyperpolarisation de $-100pA$ ou $-80pA$ est injecté à un neurone du GPeA et GPeP occasionnant une chute du potentiel de membrane de $100mV$, voir Fig 4.1, la dernière colonne. Abdi et al. [7] ont nommé «sag» cette déviation du potentiel (indiqué par une flèche sur la Fig. 4.1) et ont remarqué que son amplitude est similaire pour les deux populations. Les canaux HCN sont responsables de cette déviation du potentiel lorsqu'un courant hyperpolarisant est injecté à un neurone. Ainsi, la densité des canaux HCN est vraisemblablement la même entre les deux types de neurones.

4.1.3 Courant SK

Deister et al. [29] ont voulu tester si les canaux SK ont une influence sur la précision des PA dans les neurones du GPe. Cependant, ils ne faisaient pas encore la distinction entre les deux populations de neurones. En condition normale avec les canaux SK non bloqués, les PA sont reproductibles d'un essai à un autre (traces noires). Par contre, en condition sous apamine, bloqueur des canaux SK chez le mammifère, l'emission de PA devient imprécise, voir la Fig. 4.3. En conclusion, le courant des canaux SK contrôle le taux de décharge et améliore la précision de la décharge.

4.1.4 Courbe FI

Abdi et al. ont comparé la fréquence de décharge des deux populations de neurones en traçant la courbe entre le taux de décharge et le courant injecté, en anglais ***Firing rate and***

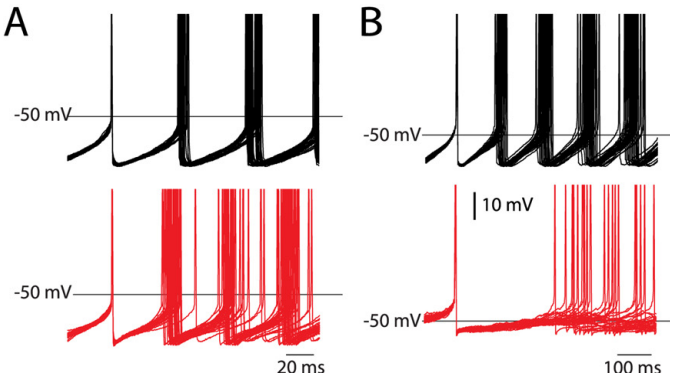

Fig. 4.3 – Trente répétitions de PA sont enregistrés, alignés et superposés. Les traces noires sont obtenues en condition contrôle. Les traces rouges sont obtenues sous apamine (bloqueur des canaux SKchez le mammifère). Les résultats sont tirés de l'article de Deister et al. [29].

***I**njected current* (FI). Dans la Fig. 4.4, les courbes FI montrent bien une différence entre les deux populations et plus précisément, nous voyons que la courbe du GPeA monte plus lentement. La Fig. 4.1 indique que le GPeP est plus excitable que le GPeA. La Fig. 4.4 expose deux types de GPeP, mais la proportion de «PV- prototypique» est 6 fois plus faible que celle de «PV+ prototypique». Nous allons ajuster notre modèle aux neurones «PV+ prototypique».

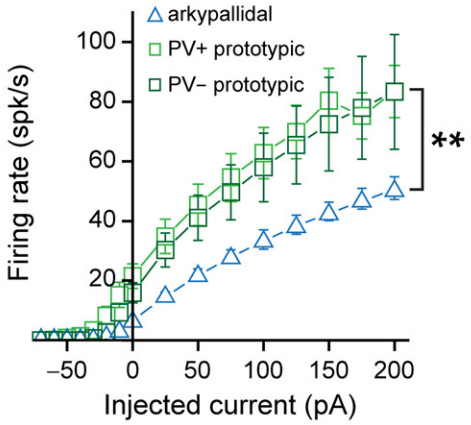

Fig. 4.4 – Courbe FI des neurones du GPeA («arkypallidal»), GPeP («PV+ prototypic»et «PV- prototypic»), tirés de l'article de Abdi et al. [7].

4.2 Modélisation des GPeA et GPeP par le formalisme d'HH

Notre modèle du GPe est basé sur le travail de modélisation de Fujita et al. [39]. Leur neurone du GPe, à compartiment unique, est défini par 10 courants ioniques différents, un courant de fuite, un courant appliqué et un courant synaptique.

Cependant le modèle de Fujita et al. [39] ne fait pas de distinction entre les neurones du GPeA et du GPeP jouant un rôle important et différent dans l'activité des GB. De plus, nous avons sélectionné 6 canaux ioniques principaux sur les 10 afin de simplifier notre modèle. D'après les articles [7] et [84], l'activité des neurones du GPeP et GPeA dépend des canaux NaP, HCN, SK (avec le calcium étant fourni par l'ouverture des Ca_{vH}) et K_v, mais aussi le courant NaF.

Ainsi, nous avons décidé de modéliser le courant des canaux NaF, NaP, K_v, Ca_{vH}, HCN et SK.

La dynamique du potentiel membranaire V est décrite par le formalisme de HH que nous avons défini à la sous-section 1.1.2. Les équations du formalisme de HH sont exactement identiques pour les deux types de neurones du GPe, seules les valeurs de quelques paramètres vont différer :

$$\begin{aligned} C_m \frac{dV}{dt} &= -I_{NaP}(V) - I_{NaF}(V) - I_{HCN}(V) - I_{Ca_{vH}}(V) - I_{K_v}(V) - I_{SK}(V) \\ &\quad -I_{leak}(V) - I_{syn}(V) + I_{app} \end{aligned} \tag{4.1}$$

où C_m est la capacité de la membrane. I_{syn} et I_{app} est le courant synaptique, resp. le courant appliqué au neurone. I_{leak} est le courant de fuite et les autres courants ont déjà été défini au paragraphe précédent. Ces courants sont représentés par les équations suivantes :

$$\begin{aligned} I_{leak} &= g_{leak}(V - E_{leak}) \\ I_{NaP} &= g_{NaP} m^3 hs (V - E_{Na}) \\ I_{NaF} &= g_{NaF} m^3 hs (V - E_{Na}) \\ I_{HCN} &= g_{HCN} m (V - E_{HCN}) \\ I_{Ca_{vH}} &= g_{Ca_{vH}} m (V - E_{Ca}) \\ I_{K_v} &= g_{K_v} m^4 h (V - E_K) \\ I_{SK} &= g_{SK} m (V - E_K) \end{aligned} \tag{4.2}$$

où g_X et E_X sont la conductance maximale et le potentiel d'inversion du canal ionique $X = \{leak, NaP, NaF, HCN, Ca_{vH}, K_v, SK\}$. m, h et s sont les variables d'activation et d'inactivation. Elles sont définies par les équations suivantes :

$$\frac{dx}{dt} = \frac{x_\infty(V) - x}{\tau_x(V)} \tag{4.3}$$

où x représente la variable m, h ou s. $x_\infty(V)$ et $\tau_x(V)$ sont les fonctions d'activation, ou d'inactivation, resp. la constante de temps. Elles sont définies par les équations suivantes :

$$\begin{aligned} x_\infty(V) &= x_{min} + \frac{1 - x_{min}}{1 + \exp\left[\dfrac{\theta_x - V}{k_x}\right]} \\ \tau_x(V) &= \tau_x^0 + \frac{\tau_x^1 - \tau_x^0}{\exp\left[\dfrac{\varphi_x - V}{\sigma_x^0}\right] + \exp\left[\dfrac{\varphi_x - V}{\sigma_x^1}\right]} \end{aligned} \tag{4.4}$$

Le courant SK et la constante de temps de la variable s du courant NaP sont caractérisés de manière différente des autres courants et une définition est donnée dans l'annexe C.

Nous avons adapté les paramètres du modèle de Fujita et al. afin de distinguer les deux types de neurones du GPe, voir annexe C. Avec notre modèle, nous avons reproduit le courant NaP et la courbe FI des données biologiques, vu précédemment sur les Figs. 4.2 et 4.4.

4.2.1 Courant NaP

D'après l'article [7], le courant NaP est deux fois plus important pour le GPeP que pour le GPeA. Nous avons simulé l'expérience *in vitro* de la Fig. 4.2 où le neurone est soumis à une rampe de tension lentement dépolarisante. Le courant NaP obtenu en simulation est comparé

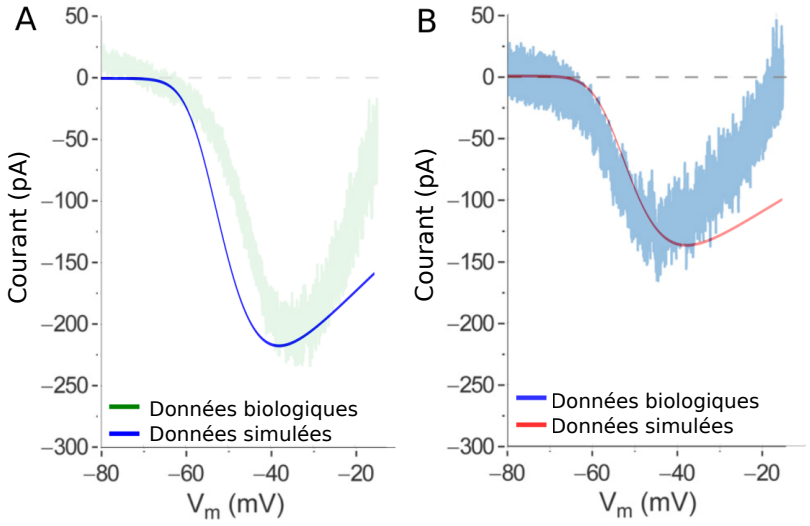

Fig. 4.5 – Comparaison du courant NaP simulé et expérimental avant la modification des paramètres $\theta_x, k_x, \tau_x^0, \tau_x^1$ afin de s'ajuster plus proprement à la courbe expérimentale. Les données biologiques sont tirées de l'article [7]. **A.** Le courant du GPeP et **B.** celui du GPeA.

avec les données biologiques en Fig. 4.5. Comme nous pouvons le voir, la courbure du courant ne correspond pas bien à celle des données biologiques.

Nous avons alors décidé d'analyser l'influence des paramètres $\theta_x, k_x, \tau_x^0, \tau_x^1$ du courant NaP sur la courbure du courant NaP et d'optimiser ces paramètres afin de s'approcher au mieux des résultats biologiques. Les résultats sont exposés en sous-section 4.4.2 et en Fig. 4.10.

4.2.2 Courant HCN

D'après l'article [7], le canal HCN est responsable de la déviation du potentiel lorsqu'un courant hyperpolarisant est injecté à un neurone du GPe. Nous avons alors cherché à savoir si notre canal HCN modélisait bien ce phénomène que Abdi et al. ont nommé un «sag ». Nous avons ajusté la conductance ionique HCN de façon à obtenir ce «sag », comme nous pouvons le voir dans la Fig 4.6(A) où un courant hyperpolarisant de $-200pA$, resp. $-180pA$, est injecté au neurone du GPeP (en bleu), resp. GPeA (en rouge). Ensuite, nous avons aussi bloqué le canal HCN, voir la Fig. 4.6(B), où aucun «sag » est visible. Ainsi, nous montrons que le courant HCN entraîne cette déviation du potentiel membranaire.

4.2.3 Courbe FI

Par la suite, nous avons adapté notre courbe FI simulée à la courbe FI de l'article [7]. Pour cela, nous avons décidé d'ajuster les conductances des différents courants ioniques, plus précisément, les courants NaF, Ca_{vH}, K_v et SK, puisque les courants NaP et HCN ont déjà été ajustés précédemment.

Étant donné la forte non linéarité des équations et l'interdépendance des paramètres, leur ajustement manuel empirique s'est révélé inadéquat. Nous avons donc eu recourt dans un second temps à un algorithme d'optimisation, plus précisément, un algorithme d'évolution différentielle, en anglais ***D**ifferential **E**volution* (DE) [20, 86]. Cet algorithme ressemble à un algorithme génétique sauf que l'algorithme de DE fait évoluer l'individu par une recombinaison des vecteurs

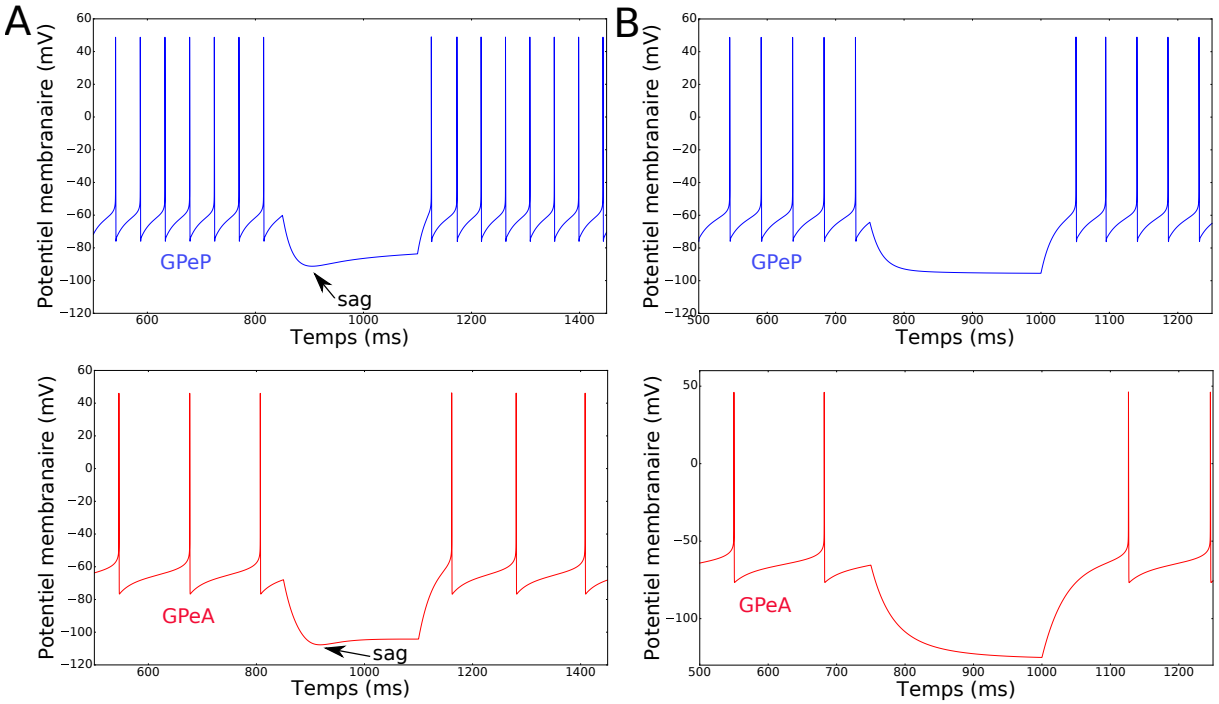

Fig. 4.6 – Comparaison du potentiel d'action avec et sans canal HCN. **A.** Le potentiel membranaire du GPeP (en bleu) et du GPeA (en rouge) avec le canal HCN et **B.** le canal HCN est bloqué.

d'individus et ensuite compare ces individus entre eux. Plus d'explication sur l'algorithme de DE est donné dans la prochaine section.

4.3 Optimisation des paramètres par l'algorithme d'évolution différentielle

L'algorithme de DE, introduit par STORN & PRICE [86], est un algorithme évolutionnaire comme aussi les algorithmes génétiques. Les algorithmes évolutionnaires s'inspirent de la théorie de l'évolution pour résoudre des problèmes, p.ex. d'optimisation de paramètres. L'idée principale de tels algorithmes est de faire évoluer un ensemble de solutions vers les meilleurs résultats possibles, i.e. d'obtenir une solution approchée à un problème d'optimisation lorsqu'il n'est pas possible de trouver une solution exacte.

4.3.1 Algorithme de DE

D'après la thèse de LAURE BUHRY [19], l'algorithme de DE est plus adapté à l'optimisation de paramètres de modèles de neurones du type HH que l'algorithme génétique et une solution est trouvée plus rapidement. L'algorithme de DE prend une population de solutions et crée des nouvelles solutions en fonction des solutions déjà existantes, ensuite les nouvelles solutions et les solutions existantes sont comparées afin de choisir les meilleures d'entre elles.

Concrètement, il existe une population de NP individus. Un individu est noté X_k^j, i.e. le $j^{ième}$ individu de la $k^{ième}$ génération dans cette population. Il existe D nombre de gènes dans un individu. D'un point de vue plus mathématique, nous parlerions d'un ensemble (*population*)

de NP vecteurs (*individus*) et dans un vecteur, il existe D nombre de paramètres (*gènes*). Un exemple illustrant cette notion est donné par la Fig. 4.7.

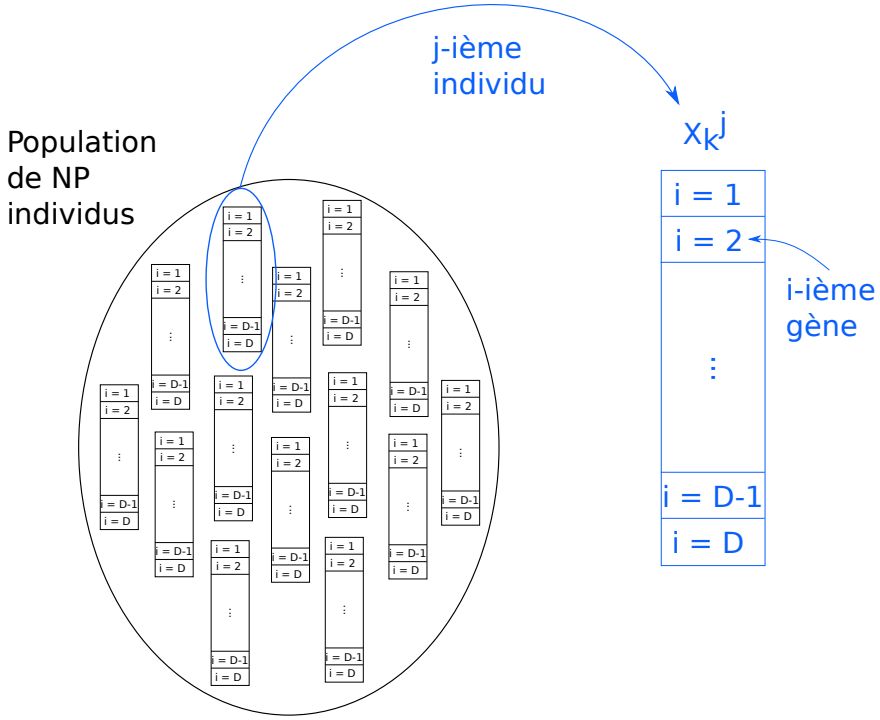

Fig. 4.7 – Exemple d'une population de NP individus à la $k^{ième}$ génération de l'algorithme de DE. Tirée de la soutenance de thèse de Laure BUHRY.

L'algorithme de DE s'opère en trois étapes principales pour une itération k.

1. La population d'individus X_k^j évolue par une recombinaison des individus, noté $X_{k,tmp}^j$. Cette étape est appelé la *différenciation*.

$$\forall j = 1, \ldots, NP \qquad X_{k,tmp}^j = X_k^{d_1} + F\left(X_k^{d_2} + X_k^{d_3}\right) \text{ avec } F \in [0;2] \tag{4.5}$$

2. Nous croisons les gènes des individus X_k^j avec ceux des individus $X_{k,tmp}^j$ afin de reformuler un nouvel individu $X_{k,mut}^j$. Cette étape est appelé la *recombinaison*.

$$\forall i = 1, \ldots, D \ , \forall j = 1, \ldots, NP \ , \ u = \mathcal{U}(0,1) \text{ et } CR \in [0,1]$$
$$X_{k,mut}^j(i) = \begin{cases} X_{k,tmp}^j(i) & \text{si} \quad u < CR \\ X_k^j(i) & \text{sinon} \end{cases} \tag{4.6}$$

3. Selon la valeur de la fonction coût, $F_{coût}$, l'individu X_k^j est remplacé, ou non, par l'individu $X_{k,mut}^j$. Cette étape est appelé la *sélection*.

$$X_{k+1}^j = \begin{cases} X_{k,mut}^j(i) & \text{si} \quad F_{coût}\left(X_{k,mut}^j\right) \leqslant F_{coût}\left(X_k^j\right) \\ X_k^j(i) & \text{sinon} \end{cases} \tag{4.7}$$

À la première étape, $X_k^{d_1}$ est le meilleur individu trouvé à partir de la fonction coût et $X_k^{d_2}$ et $X_k^{d_3}$ sont choisis parmi la population courante de manière aléatoire.

Remarque Contrairement à l'algorithme de DE, l'algorithme génétique n'a pas une évolution des individus par une recombinaison «géométrique» des vecteurs des individus et aussi il n'a pas de combinaison, à chaque itération, à un nouvel individu $X^j_{k,mut}$ qui pourrait éventuellement remplacer X^j_k à la génération suivante [19]. De plus, dans l'algorithme DE, il y a moins d'étapes et de paramètres internes à régler, permettant ainsi à cet algorithme d'être plus rapide (dans notre cas) [19].

La constante de différenciation F et de recombinaison CR joue un rôle très important dans la convergence vers une solution optimale. Lorsque F est trop petit, l'individu recombiné $X^j_{k,trial}$ resta très proche de l'individu X^j_k. En revanche, si F trop grand, le nouvel individu risque de sortir de l'intervalle imposé par l'utilisateur. CR, de son côté, doit être proche de 1 afin que nous puissions générer des nouveaux individus. D'après STORN & PRICE [86], les valeurs idéales seraient $F = 0,5$ et $CR = 0,9$.

L'algorithme 14 illustre l'algorithme DE utilisé pour l'optimisation des conductances ioniques. L'initialisation de la population commence avec la détermination des valeurs maximales et minimales des conductances ioniques, X_{Max} et X_{Min}. Ensuite, la première population d'individus X est fixée comme la combinaison des individus X_{Max} et X_{Min}. Avec la fonction coût, un score est calculé, i.e. une erreur est calculée. Cette étape est plus détaillée à la prochaine sous-section. Parmi la population d'individus, celui avec le meilleur score est choisi et est noté X_{Best}.

Lors d'une itération, nous retrouvons bien la première étape de **la différenciation**, où l'individu X_{Tmp} est combiné en fonction du meilleur individu X_{Best} et de deux individus tirés au hasard dans la population X. Ensuite, nous vérifions bien que l'individu X_{Tmp} se trouve bien dans le domaine de définition, i.e. entre X_{Max} et X_{Min}. Ensuite vient l'étape de **la recombinaison** où une nouvelle population d'individus est reconstruite à partir de X et X_{Tmp}. Pour finir, il y a l'étape de **la sélection**, où en fonction du score de X et X_{Mut}, les nouveaux individus de la population X sont sélectionnés.

Dans notre cas, les gènes, ou les paramètres, sont les conductances ioniques que nous souhaitons optimiser afin d'ajuster notre modèle aux données biologiques. Dans la Fig. 4.8, les différentes étapes de l'algorithme sont données schématiquement. Tout d'abord, l'algorithme DE calcule l'individu X_{Mut}, comme décrit dans l'algorithme 14. Cet individu X_{Mut} est donné en paramètre au logiciel SiReNe où le nombre de PA est calculé en fonction du courant injecté afin de tracer la courbe FI. Ensuite, la courbe FI est tracée et elle est comparée à la courbe FI expérimentale. La fonction $F_{coût}$, expliquée dans la sous-section suivante, calcule l'erreur quadratique entre les deux courbes FI et la valeur est envoyée à l'algorithme DE afin qu'il puisse régénérer un nouveau jeu de données. Ce processus est reproduit jusqu'au nombre d'itérations, $NbIter$ donné par l'utilisateur.

Il est important de bien choisir l'intervalle de définition de chaque paramètre. Si cet intervalle est trop grand, alors il sera plus difficile de trouver les bonnes valeurs du paramètres. Nous avons repris les valeurs du modèle du GPe et avons ajouté $\pm 70\%$ de cette valeur pour le maximum et le minimum. Lorsque la valeur trouvée à la fin de l'algorithme était trop près de la borne, alors nous reajustions les bornes pour ce paramètre.

4.3.2 Fonction coût multi-critère

Nous cherchons à ajuster les conductances synaptiques et pour cela, la courbe FI biologique est comparée avec la courbe simulée au moyen d'une fonction coût.

Algorithme 14 : Algorithme de DE

Entrée : NP - nombre d'individus
Entrée : D - nombre de gènes
Entrée : $NbIter$ - nombre d'itérations
Entrée : $X_{Max}[D]$ - tableau de valeurs maximales données au différents gènes
Entrée : $X_{Min}[D]$ - tableau de valeurs minimales données au différents gènes
Entrée : $X[D, NP]$ - tableau contenant la population des meilleurs individus de l'itération précédente
Entrée : $X_{Tmp}[D]$ - tableau temporaire
Entrée : $X_{Mut}[D, NP]$ - tableau avec des gènes recombinés
Entrée : $score_X[NP]$ - tableau de scores de la population X obtenue avec $F_{coût}$
Entrée : $score_{Mut}[NP]$ - tableau de scores de la population X_{Mut} obtenue avec $F_{coût}$
Sortie : $X_{Best}[D]$ - tableau du meilleur individu

$F \leftarrow 0,5$
$CR \leftarrow 0,9$
$X \leftarrow X_{Min} + Random(D, NP) \times (X_{Max} - X_{Min})$
`// Random(D,NP) renvoie une matrice (D,NP) composée de valeurs entre 0 et 1`
$F_{coût}$ calcule le score, $score_X$, de la population X
Choisie parmi la population d'individus X le meilleur score, X_{Best}
pour $k \leftarrow 1$ **à** $NbIter$ **faire**
  pour $j \leftarrow 0$ **à** NP **faire**
    `// Différenciation`
    $d_2 \leftarrow Randint(NP)$
    $d_3 \leftarrow Randint(NP)$
    `// Randint(NP) renvoie une valeur entière entre 0 et NP`
    $X_{Tmp} \leftarrow X_{Best} + F \times (X[d_2] - X[d_3])$
    si $X_{Tmp} > X_{Max}$ **alors**
      $X_{Tmp} \leftarrow X_{Min} + Random(D) \times (X_{Max} - X_{Min})$
    fin
    si $X_{Tmp} < X_{Min}$ **alors**
      $X_{Tmp} \leftarrow X_{Min} + Random(D) \times (X_{Max} - X_{Min})$
    fin
    `// Recombinaison`
    $mut \leftarrow Random(D)$
    si $mut < CR$ **alors**
      $X_{Mut}[j] \leftarrow X_{Tmp}$
    sinon
      $X_{Mut}[j] \leftarrow X$
    fin
  fin
  `// Sélection`
  $F_{coût}$ calcule le score, $score_{Mut}$, de la population X_{Mut}
  pour $j \leftarrow 0$ **à** NP **faire**
    si $scoreMut[j] \leqslant scoreX[j]$ **alors**
      $X[j] \leftarrow X_{Mut}[j]$
      $score_X[j] \leftarrow score_{Mut}[j]$
    fin
    Choisie parmi la population d'individus X les individus avec le meilleur score, X_{Best}
  fin
fin

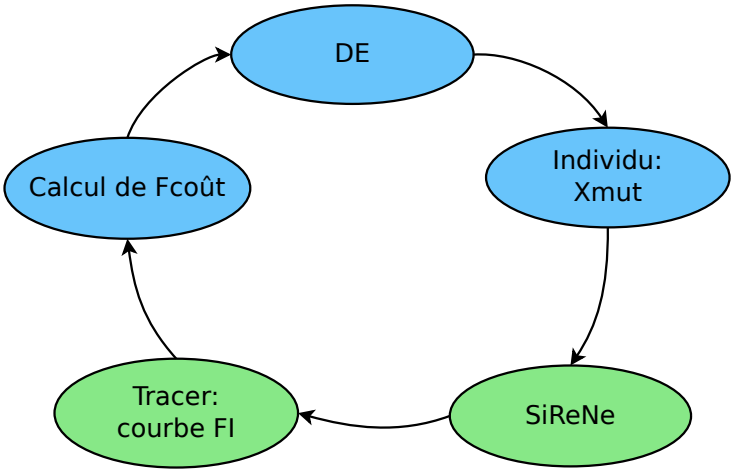

Fig. 4.8 – Schéma de l'algorithme DE. En bleu, ce sont des étapes effectuées par l'algorithme DE et en vert, ce sont les étapes effectuées par SiReNe.

Fonction coût : calcul de l'erreur

Afin de simplifier notre recherche de paramètres, nous avons décidé de calculer l'écart entre les données biologiques et simulées avec l'erreur quadratique. Ainsi, nous comparons les fréquences de décharge de la courbe FI simulée, noté f_{simul}, avec celles des données biologiques de Abdi et al., noté f_{ref}.

$$F_{coût} = \frac{1}{nbVal} \|f_{ref} - f_{simul}\|^2 \tag{4.8}$$

où $nbVal$ est le nombre de valeurs comparées.

Fonction coût : poids sur les points clés de la courbe FI

Puisque la courbe FI biologique a des écarts types, nous avons ajouté des conditions en fonction de l'écart type. Lorsque les fréquences de décharge de la courbe FI simulée, f_{simul}, ne se trouvent pas dans l'écart type, alors nous ajoutons un poids à l'erreur calculée, i.e. nous multiplions l'erreur par une valeur, e.g. 10.

Une deuxième condition a ensuite été ajoutée. Nous savons que le GPeP, resp. GPeA, peut décharger entre 0 et $40Hz$, ce qui correspond à des courants injectés entre $-50pA$ à $50pA$. Par conséquence, nous avons pondéré, avec un poids plus important, la courbe FI entre $-50pA$ à $50pA$, lorsque celle-ci ne se trouve pas dans l'écart type.

La valeur de la fonction coût peut alors vite augmenter lorsque ces différentes conditions sont réunies allant jusqu'à 5000, ou même plus, dans certains cas.

Constante de différenciation F

Comme dit précédemment, lorsque la constante de différenciation F est grande, la population sera probablement différente de la précédente. Étant donné que la fonction coût peut atteindre des valeurs très grandes, nous avons fait le choix de faire évoluer F de façon à ce que nous puissions chercher des nouvelles populations d'individus, lorsque l'erreur est trop importante.

La constante de différenciation est fixée à 1 à l'initialisation, ensuite nous l'avons fait varier en fonction de la valeur de la fonction coût. Si la fonction coût est plus grande que 3000 alors F

reste à 1, sinon si l'erreur est entre 3000 et 1000, alors F est fixée à $0,8$, sinon la constante est égale à $0,5$.

MPI

Afin de réduire le temps de recherche de l'algorithme DE, nous avons adapté l'algorithme au calcul parallèle en utilisant MPI. Nous avons une population d'individus et pour chaque individu, nous traçons une courbe FI, donc nous les lançons séparément sur SiReNe. Ainsi, la machine principale, la machine 0, distribue un groupe d'individus à chaque machine et les différents individus sont calculés parallèlement sur différentes machines. Les résultats sont ensuite transmis à la machine principale qui les transmet à l'algorithme DE. La Fig. 4.9 donne un exemple d'une telle distribution d'individus à différentes machines.

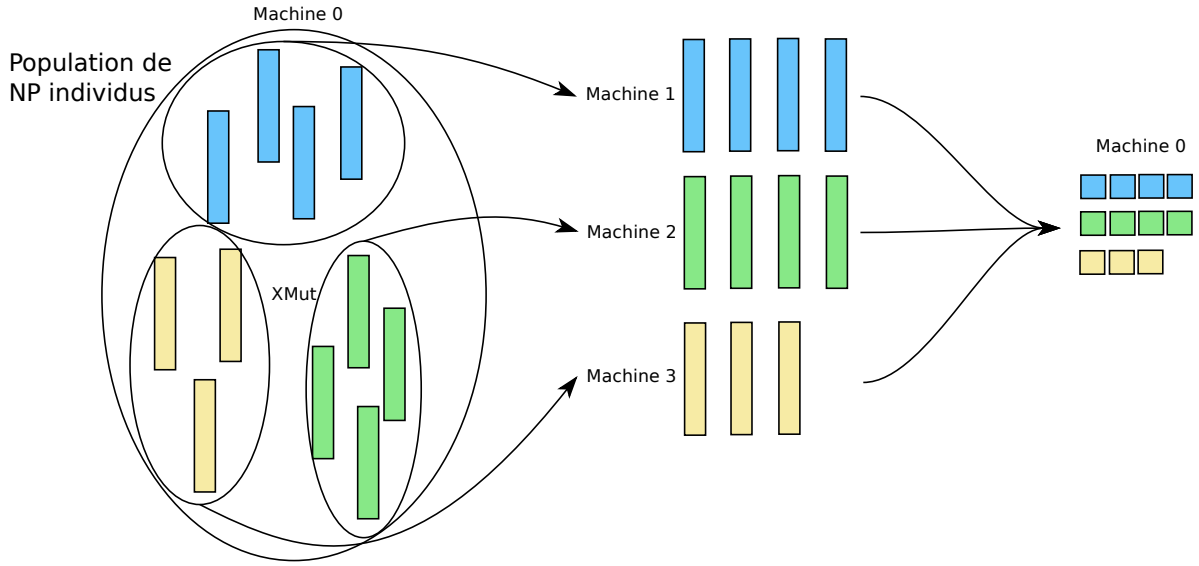

Fig. 4.9 – Exemple de distribution de la population d'individus sur plusieurs machines en MPI.

4.4 Résultats de simulation

Dans cette section, les résultats simulés sont comparés aux résultats expérimentaux.

4.4.1 Contexte expérimental

Les simulations présentées ci-dessous ont été réalisées sur `Grid'5000` sur le site de Nancy avec le cluster `grvingt`. `grvingt` est sous Linux Debian 4.19 amd64 avec 2 CPUs Intel Xeon Gold 6130 avec 16 coeurs chacun, et 192GB de RAM. Les temps ont été mesurés avec la fonction d'OpenMP, `omp_get_wtime()`, et le programme a été compilé avec `gcc` 8.3.0 et le niveau d'optimisation `-O3`.

Le Tab. 4.1 donne la liste des paramètres de simulation lorsque SiReNe est lancé. Le temps de simulation biologique est de 4s, mais les 2 premières secondes correspondent à un régime de transition et ne sont pas prises en compte dans l'analyse des résultats. Le Tab. 4.2 sont les paramètres de simulation de l'algorithme DE.

Paramètres de simulation		
Paramètres généraux	Type d'interpolation	Courbe de Bézier
	Méthode numérique	RK2
	Taille du réseau	1
	Temps biologique $[ms]$	4000
	Pas de temps $[ms]$	1e-2
	# thread(s)	18
Paramètres du courant appliqué	Type du courant appliqué	constant
	$I_{app}\,[\mu A]$	-50 à 150

Tab. 4.1 – Liste des paramètres de simulation lors du lancement de **SiReNe** avec l'algorithme de DE.

Paramètres de simulation		
Paramètres généraux	NP	100
	N_{iter}	100
	D	5
	CR	0,9
Domaine de définition des conductances	g_L	2 - 20
	g_{NaF}	1 000 - 30 000
	g_{Kv}	200 - 10 000
	g_{Ca}	0,8 - 50
	g_{SK}	0,8 - 20

Tab. 4.2 – Liste des paramètres de simulation de l'algorithme DE.

4.4.2 Courant NaP

Comme expliqué dans la sous-section 4.2.1, nous n'étions pas satisfaits de nos résultats, nous avons alors changé quelques valeurs des paramètres $\theta_x, k_x, \tau_x^0, \tau_x^1$ du courant NaP afin d'ajuster notre modèle aux données expérimentales. Nous observons en Fig. 4.5(A) que la courbe simulée du GPeP descend plus tôt et que la pente de la montée est plus lente que la courbe expérimentale. En revanche sur la Fig. 4.5(B), la courbe simulée du GPeA s'ajuste bien lors de la descente, mais la montée est bien plus lente que la courbe expérimentale.

Après avoir optimisé les valeurs des paramètres du courant NaP, nous avons obtenu les résultats de la Fig. 4.10. La courbe s'ajuste beaucoup mieux aux données expérimentales. Comme pour les résultats biologiques, le courant du GPeP (Fig. 4.10A.) est deux fois plus important que celui du GPeA (Fig. 4.10B).

Dans l'annexe C, les paramètres modifiés $\theta_x, k_x, \tau_x^0, \tau_x^1$ du courant NaP sont notés en rouge.

4.4.3 Courbe FI

Après 100 itérations et environ 2h de simulation, l'algorithme DE a trouvé les conductances données dans le Tab. 4.3. Comme attendu les valeurs sont différentes entre les deux populations de neurones. Les valeurs de conductances NaP et HCN n'ont cependant pas été trouvées avec l'algorithme. La conductance NaP a été déterminée avec le processus décrit à la sous-section 4.2.1

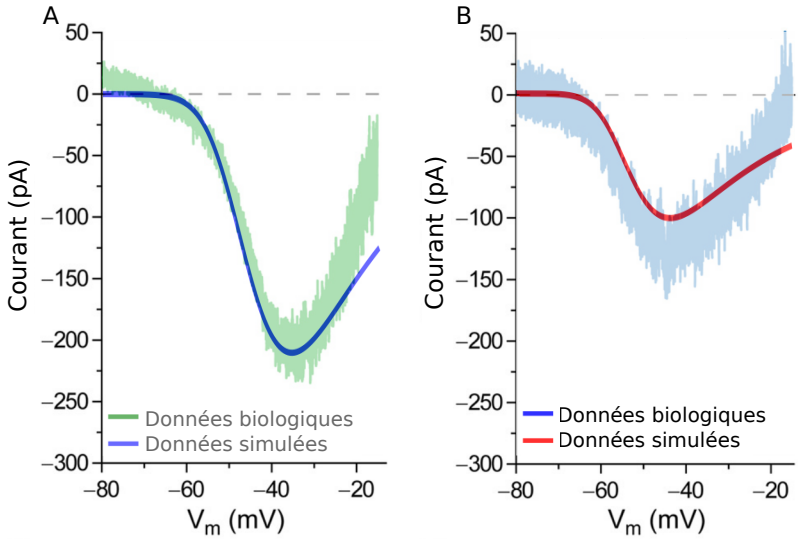

Fig. 4.10 – Comparaison du courant NaP simulé et expérimental de l'article [7]. A. le courant du GPeP et B. celui du GPeA.

et la conductance HCN a été ajustée à la main.

Conductances ioniques retenues (en nS)	g_L	g_{NaP}	g_{NaF}	g_{Kv}	g_{HCN}	g_{Ca}	g_{SK}
GPeA	2,731	6,0	8 698,126	9 330,831	0,79	1,911	19,968
GPeP	5,614	18,0	19 474,447	9 670,274	4,19	49,592	0,802

Tab. 4.3 – Liste des paramètres de simulation de l'algorithme DE.

La Fig. 4.11 donne l'évolution de la fonction de coût suivant les itérations. Nous remarquons que dans un premier temps la fonction de coût diminue drastiquement et que vers la fin, elle ne varie presque plus du tout. La plus petite valeur de $F_{coût}$ est 433,976 pour le GPeP et 134,43 pour le GPeA. La courbe FI du GPeP a été plus difficile à ajuster aux données expérimentales, comme nous pouvons le voir avec la Fig. 4.11.

La Fig. 4.12 représente la fréquence de décharge en fonction du courant injecté. Ces résultats sont fidèles aux données expérimentales et ont été trouvés avec l'algorithme DE décrit précédemment.

4.4.4 Potentiel membranaire

Dans la Fig. 4.13, le potentiel de membrane est tracé en fonction du temps pour les neurones du GPeP (A) et GPeA (B). Les résultats de la première colonne sont ceux où aucun courant n'est injecté au neurone, sur la deuxième colonne, un courant de 100pA est injecté pendant 1s et sur la dernière colonne, un courant de -180pA, resp. -200pA, est injecté pendant 250ms à un neurone du GPeP, resp. GPeA.

Nous remarquons que, pour les deux types de neurones, ils émettent sans qu'aucun courant ne soit injecté, cependant le GPeP émet plus que le GPeA. Lorsqu'un courant positif est injecté, le neurone émet beaucoup plus. Toutefois, quand un courant négatif est injecté le neurone

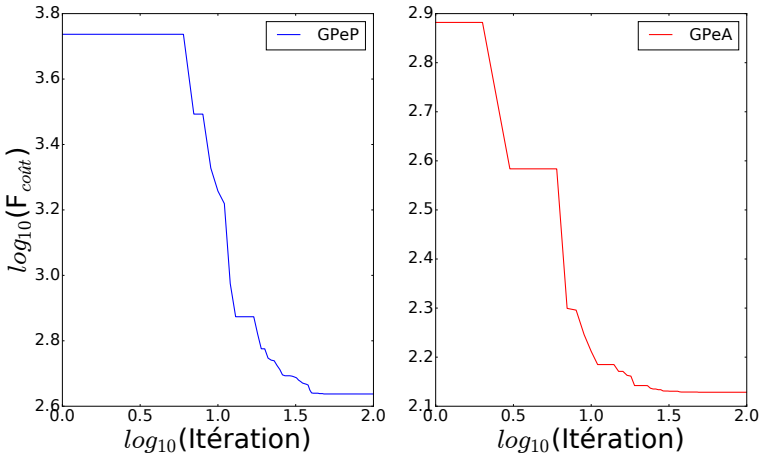

Fig. 4.11 – L'évolution de la fonction coût, $F_{coût}$, au fur des itérations.

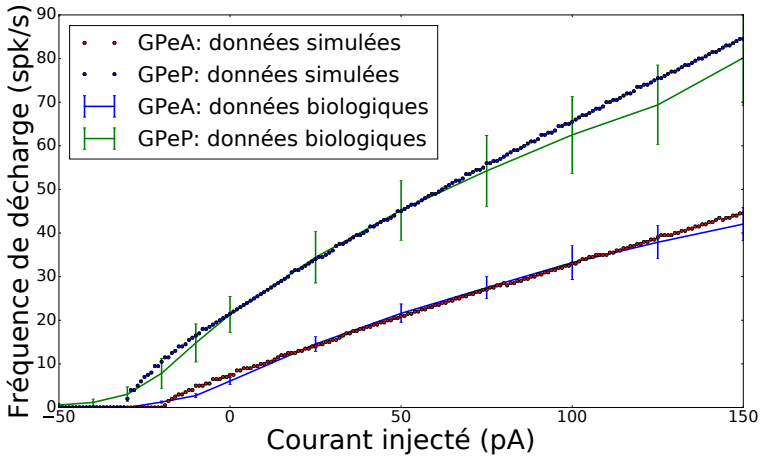

Fig. 4.12 – Comparaison de la courbe FI avec des données expérimentales de l'article [7] et simulées des neurones du GPeA et GPeP.

s'hyperpolarise et crée, ce que Abdi et al. [7] a nommé, un «sag».

Le comportement du neurone aux différents courants injectés est similaire aux résultats de l'article [7], voir Fig. 4.1. La seule grande différence, que nous n'avons pas réussi à améliorer, est l'hyperpolarisation, en anglais ***A**fter**H**yper**P**olarization* (AHP). La valeur de l'AHP en expérimentale est de $-67mV$ pour le GPeP et de $-73mV$ pour le GPeA. Toutefois pour notre modèle, elle est de $-76mV$ pour le GPeP et de $-77mV$ pour le GPeA. La différence de valeurs entre les deux types de neurones est très petite en simulation par rapport à l'expérimentale. De plus, pour le GPeP, nous avons presque $10mV$ de différence entre le modèle simulé et les données expérimentales.

Le «sag» est un peu moins visible sur nos simulations que sur les résultats expérimentaux de l'article [7]. Toutes ces petites différences avec les résultats expérimentaux restent tout à fait raisonnables et montrent la difficulté et les limites de la modélisation.

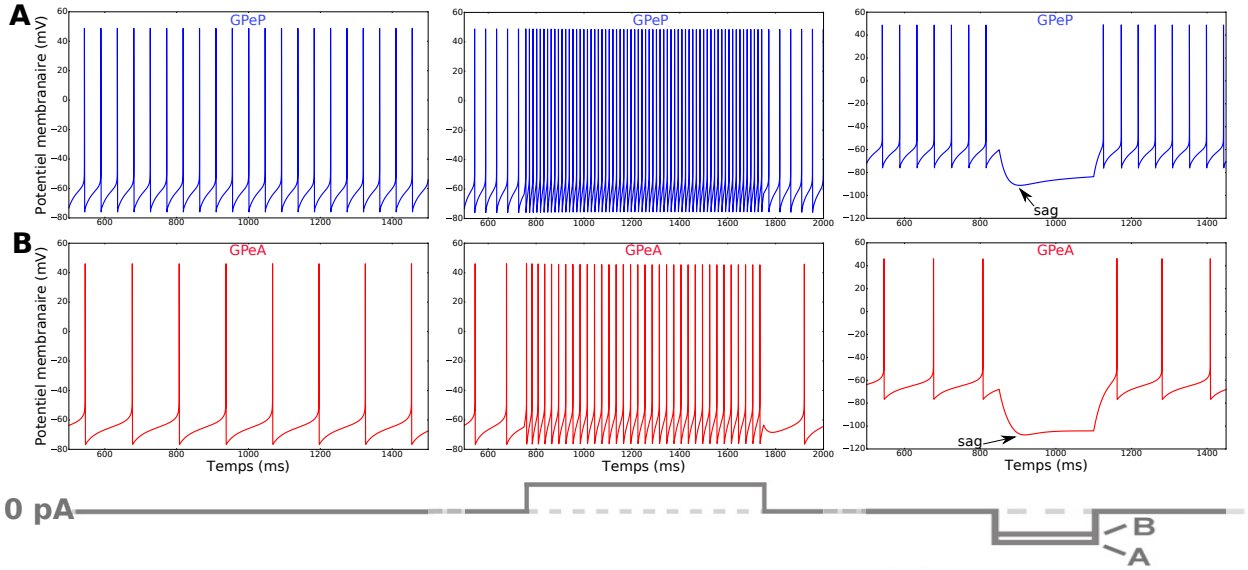

Fig. 4.13 – Potentiels membranaires simulés des neurones du GPeA et GPeP. **A,** activité d'un neurone du GPeP. $1^{ière}$ colonne : Potentiel sans courant injecté, $2^{ième}$ colonne : avec $100pA$ de courant injecté pendant $1s$ et $3^{ième}$ colonne : la réponse à une impulsion de courant hyperpolarisante de $-200pA$ ou $-180pA$ pendant $250ms$, provoquant un «sag». **B,** Même expérimentation pour un neurone du GPeA.

4.5 Conclusion

Dans cette section, nous avons modélisé les neurones du GPe. Cependant, Abdi et al. [7] et Mallet et al. [70] ont distingué deux types de neurones du GPe, le GPeP et le GPeA. Nous nous sommes basés sur le modèle de Fujita et al. [39], où la distinction des deux types n'était pas encore prise en compte, et l'avons personnalisé afin de pouvoir reproduire les données expérimentales de l'article [7] et distinguer les deux populations de neurones.

La modélisation du courant NaP et l'ajustement de la courbe FI nous ont permis d'affiner notre modèle. L'algorithme de DE nous a permis de chercher les meilleures conductances ioniques en comparant la courbe FI simulée à la courbe FI expérimentale. Les seuls résultats, que nous n'avons pas eu l'opportunité d'étudier et de simuler, sont ceux sur les canaux SK de l'article [29], mentionné à la section 4.1.3.

5

Modélisation et simulation du réseau STN-GPeA/P-MSN

Sommaire

La MP est caractérisée par des oscillations pathologiques au niveau des GB. Les circuits, les plus propices à une synchronisation pathologique, sont le circuit du STN avec le GPe [16, 82] et le circuit du GPeP avec le GPeA [69, 70]. De la Crompe et al. [27] et McCarthy et al. [72] suggèrent aussi que l'inhibition du STR induit une activité anormale dans les GPe et que celle-ci augmenterait potentiellement les synchronisations pathologiques.

Il existe trois types de modélisation de réseaux de neurones en général. Le premier type est la modélisation simplifiée d'une population de neurones, i.e. l'activité d'une population de neurones est résumée par une seule équation (l'évolution de la fréquence de décharge moyenne des neurones au sein d'une même population). Cette approche est plus analytique et mathématique,

et ne consiste qu'à étudier le comportement des populations entre elles. Cette modélisation ne prend pas en compte l'aspect biologique d'un neurone et simplifie beaucoup l'activité d'un réseau. Les articles [52, 83] se sont inspirés de cette approche, afin de modéliser les GB.

Le deuxième type est la modélisation des neurones par le modèle IF quadratique. Cette approche a déjà été expliquée dans la sous-section 1.1.2. L'activité d'un neurone est représentée par une équation. Cependant le modèle IF ne peut pas expliquer la biochimie et la biophysique des neurones, comme e.g. les canaux ioniques. Les articles [45, 64] présentent des modèles des GB avec ce type de modélisation. Les modèles IF sont faciles et rapides à simuler et permettent de simuler des millions de neurones en peu de temps. Par contre, l'analyse, mais aussi la nature et la contribution potentielle, des canaux ioniques n'est pas prise en compte et ça nous ramène à la troisième variante qui est le formalise de HH, expliqué à la sous-section 1.1.2.

Le formalisme de HH permet de tenir compte de la présence de différents types de canaux ioniques et de modéliser la présence de chaque type en considérant une distribution moyenne des canaux de ce type qui va notamment dépendre de la conductance maximale de l'ion considéré dans le modèle. Ainsi un neurone est représenté par plusieurs équations. Cette variante est celle qui se rapproche le plus de la réalité biologique. Cependant, elle est plus coûteuse en temps de simulation. Nous avons choisi de modéliser notre réseau des GB avec ce formalisme puisqu'il nous permet d'étudier les effets de certains canaux ioniques sur l'activité du réseau en condition saine, resp. parkinsonienne. Un exemple est celui du canal M du MSN-D2 dont le blocage engendre des synchronisations au niveau du réseau, comme le montre l'article [72] (voir sous-section 3.2.8). L'article [26] utilise ce type de modèle sauf qu'ils ne distinguent pas les deux populations de neurones du GPe et considèrent moins d'une centaine de neurones dans les simulations. Leur modèle ne respecte ni la proportion de neurones ni les connexions synaptiques des GB.

Dans ce chapitre, notre modèle des GB distingue les deux populations de neurones du GPe, ce qui est important d'après les articles [69, 70]. Nos neurones du GPeA et GPeP ont été validés par les données biologiques de l'article [7]. Le réseau est aussi composé des populations STN et MSN. Le modèle respecte les proportions de neurones entre chaque population et le nombre de connexions synaptiques. De plus, nous validons les connexions synaptiques de notre modèle avec des données expérimentales de l'article d'Aristieta et al.[10] où a été étudiée la connectivité du réseau des GB.

Ce chapitre se compose de trois sections principales. La première section illustre les résultats biologiques sur les connectivités entre les différentes populations de neurones des GB et sur quelques résultats de la MP. La deuxième section décrit la modélisation du réseau complet. La troisième section porte sur les résultats lorsque nous comparons notre modèle des GB avec les résultats expérimentaux et pour finir, nous simulons le réseau en conditions normale et parkinsonienne.

5.1 Résultats biologiques

L'article [10] étudie le schéma de connectivité synaptique lié aux neurones du GPeA et du GPeP. Les chercheurs ont analysé dans quelle proportion les deux populations de neurones du GPe sont connectées aux autres populations, i.e. du STN et du STR.

5.1.1 Boucle (MSN-D2)-GPe

La Fig. 5.1 montre l'impact des entrées striatales D2, MSN-D2, sur les neurones du GPeA et GPeP. Les neurones du D2–D ont été excités par optogénétique (impulsion de lumière bleue pendant 2 secondes) et l'impact de cette excitation sur le reste du réseau des GB a été étudié.

Les neurones du GPeP sont inhibés par la désinhibition du MSN-D2. Par contre, les neurones du GPeA, comme ceux du STN, sont très excités lors de cette manipulation.

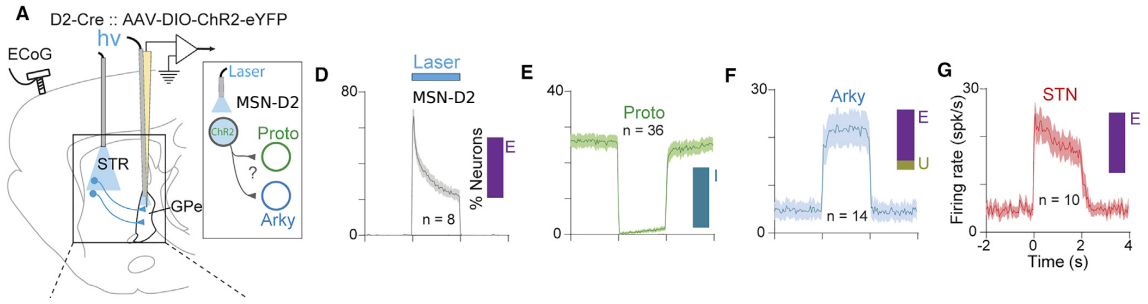

Fig. 5.1 – L'excitation des MSN-D2 entraîne des activités opposées entre le GPeA et le GPeP. (A) Schéma de l'expérimentation chez des souris où les neurones du MSN-D2 sont activés par optogénétique. (D-G) Fréquence de décharge en fonction du temps des différentes populations de neurones : (D) MSN-D2, (E) GPeP, (F) GPeA, (G) STN. Les barres de côté représentent le pourcentage de neurones excités (E), inhibés (I) et non affectés (U). Résultats tirés et modifiés de Aristieta et al. [10].

5.1.2 Boucle STN-GPe

Dans une deuxième expérience présentée dans l'article [10], les neurones du STN ont été inhibés (voir Fig. 5.2). Comme pour la première expérience, les populations de neurones du GPe réagissent de manières opposées. Les prototypiques sont inhibés et les arkypallidaux sont, quant à eux, excités. Les auteurs concluent que ces résultats montrent que les neurones du GPeA sont principalement contrôlés par la désinhibition des neurones du GPeP, avec une petite contribution excitatrice des neurones du STN.

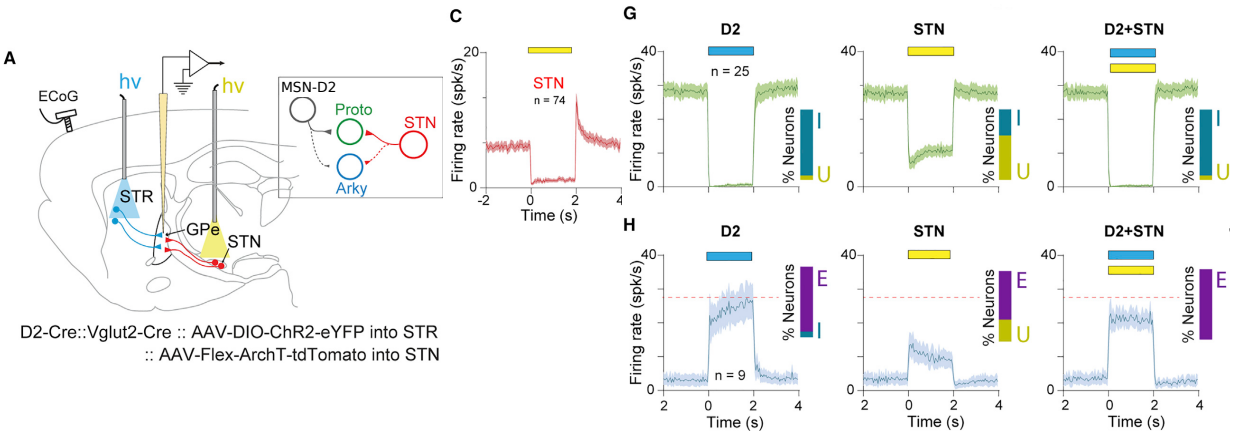

Fig. 5.2 – L'inhibition du STN entraîne l'excitation du GPeA. (A) Schéma de l'expérimentation chez les souris où les neurones du STN sont inhibés optogénétiquement. (C) Fréquence de décharge des neurones du STN. (G+H) Fréquence de décharge des neurones du GPeP (G) et du GPeA (H) en premier avec l'opto-activation du MSN-D2, en deuxième avec l'opto-inhibition du STN et en dernier avec les deux ensemble. Résultats tirés et modifiés de Aristieta et al. [10].

Dans une troisième expérience, Fig 5.3, les auteurs excitent les neurones du STN. Les neurones du GPeP sont fortement excités lors de cette expérience, alors que les neurones du GPeA sont

inhibés. Ces résultats montrent encore une fois que l'activité du GPeA est fortement contrôlée par les neurones du GPeP et moins par les neurones du STN.

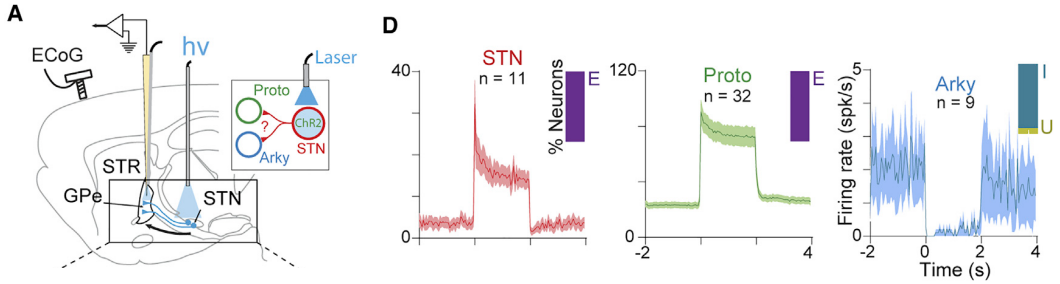

Fig. 5.3 – L'excitation des STN entraînant des activités opposées entre le GPeA et le GPeP. (A) Schéma de l'expérimentation chez des souris où les neurones du STN sont opto activés. (D) Fréquence de décharge de l'opto-activation des neurones du STN, mais aussi la fréquence de décharge des neurones du GPeP (au milieu) et du GPeA (à droite). Résultats tirés de Aristieta et al. [10].

5.1.3 Boucle GPeA-GPeP

Dans une quatrième expérience, Fig. 5.4, Aristieta et al. regardent l'activité entre le GPeP et le GPeA. Lorsque le GPeP est excité, voir Fig. 5.4 1^{ier} cadre de gauche, le GPeP inhibe très fortement le GPeA, consolidant les expériences précédentes. En revanche, lorsque les neurones du GPeA sont fortement excités, voir Fig. 5.4 $2^{ième}$ cadre de droite, les neurones du GPeP ne sont pas du tout affectés par l'activité du GPeA, montrant aussi qu'il n'existe probablement aucune connexion allant du GPeA vers le GPeP.

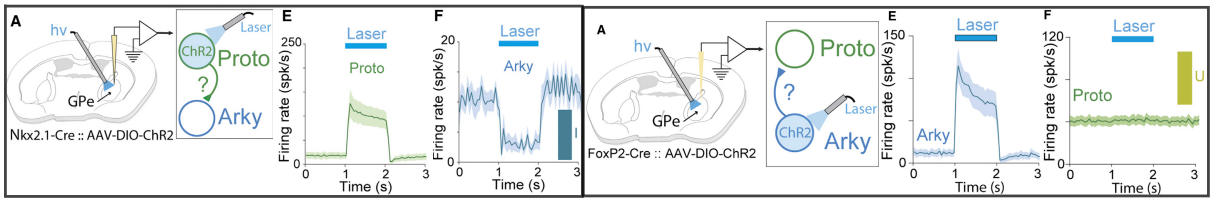

Fig. 5.4 – 1^{ier} cadre de gauche : Le GPeP inhibe fortement le GPeA. (A) Schéma des expériences *ex vivo* où le GPeP est opto activé. (E+F) Fréquence de décharge en fonction du temps du GPeP (E) et du GPeA (F). $2^{ième}$ cadre de droite : Le GPeA n'a aucun effet sur l'activité du GPeP. (A) Schéma des expériences *ex vivo* où le GPeA est opto activé. (E+F) Fréquence de décharge en fonction du temps du GPeP (E) et du GPeA (F). Résultats tirés de Aristieta et al. [10].

5.1.4 Conductances

Le Tab. 5.1 reprend toutes les conductances synaptiques estimées dans l'article [10] grâce aux expériences d'optogénétique présentées ci-dessus.

5.1.5 Effets de la maladie de Parkinson sur le réseau STN-GPeA-GPeP-MSN

Dans cette sous-section, nous allons présenter les différents résultats expérimentaux, montrant l'impact de quelques connexions synaptiques dans les GB lors de la MP.

Conductances synaptiques $g_{syn,XY}$ (en nS)		
X \ **Y**	**GPeA**	**GPeP**
STN	1,423 ± 0,45	5,429 ± 0,81
GPeA	-	-
GPeP	4,38 ± 1,07	-
MSN-D2	3,487 ± 1,08	24,28 ± 2,49

Tab. 5.1 – Conductances synaptiques du réseau des GB d'après l'article [10].

McCarthy et al. montrent dans leur article [72] que l'amplification de la dynamique du STR, et plus précisément MSN-D2, est responsable de l'augmentation des oscillations pathologiques dans la MP. En effet lors de la MP, le courant M, étant présent dans le MSN (voir l'annexe A), est partiellement bloqué [72], conduisant ainsi à l'amplification de la dynamique du STR. Toujours d'après l'article [72], les neurones du MSN s'inhibent mutuellement et ainsi ils pourraient avoir la propriété de générer des oscillations pathologiques. Ces résultats ont déjà été présentés dans la sous-section 3.2.8.

D'après De la Crompe et al.[27] et Nambu & Tachibana[75], les entrées corticales vers le STN et les interconnexions réciproques GPe-STN sont importantes pour la génération et l'amplification des oscillations pathologiques de la MP. Ces oscillations pathologiques, dans le STN, sont ensuite transmises au GPi et peuvent favoriser les symptômes de la maladie. Corbit et al. [26] avaient aussi soutenu, quelques années auparavant, que l'activité β dans le STN pourrait agir comme un synchroniseur du GPe et ensuite l'amplification des oscillations β dans la boucle GPe-STR pourrait renforcer les oscillations du STN.

Selon Glajch et al. [44], les connexions synaptiques GPe-STR sont renforcées lors de la MP. Les résultats de leurs recherches montrent que l'entrée renforcée GPe-STR peut participer aux oscillations β.

En conclusion, l'origine de ces oscillations pathologiques n'est toujours pas connue. D'un côté, une théorie considère que la boucle GPe-STN serait à l'origine des oscillations et celles-ci sont ensuite propagées au reste du réseau et d'un autre coté, une seconde théorie estime que la boucle MSN-MSN serait à l'origine des oscillations et ensuite celles-ci sont propagées par la voie STR-GPe.

5.2 Modélisation du réseau des GB

Le Tab. 5.2 a été repris de la sous-section 1.3.1 et décrit les différentes structures modélisées des GB. Pour chaque population de neurones, nous indiquons les articles sur lesquels nous nous sommes basés. Le STN est un modèle repris de l'article de Terman et al. [90]. Dans le chapitre précédent, nous avons présenté la modélisation des neurones du GPe et le modèle du MSN-D2 est celui de l'article de McCarthy et al. [72]. Le nombre de neurones par structure est tiré de l'article de Oorschot [79]. Les connexions synaptiques sont inspirées de l'article de Lindahl & Kotaleski [64].

La Fig. 5.5 représente le schéma du réseau des GB déduit, en partie, de l'article [10]. Le STN excite le GPeA et le GPeP. Le GPeA inhibe le STR. Le GPeP inhibe le STN et GPeA. Le STR, et plus précisément le MSN-D2, inhibe le GPeP et s'inhibe lui même aussi. Un courant I_{app} est appliqué au STN et au STR afin de simuler le courant excitateur arrivant du cortex.

Structures des GB et nombre de neurones chez le rat d'après [79]	**Courants respectifs**
STN [90] *(13 600 neurones)*	I_L, I_K, I_{Na}, I_T, I_{Ca}, I_{AHP}
GPe(P/A) [39] *(GPeA : 11 500 neurones et GPeP : 34 500 neurones)*	I_L, I_{NaF}, I_{NaP}, I_{Kv}, I_{HCN}, I_{SK}
MSN-D2 [72] *(1 300 000 neurones)*	I_L, I_{Na}, I_K, I_M

Tab. 5.2 – Les différentes populations des GB de notre modèle. Sur chaque population de neurones, nous retrouvons les articles sur lesquels nous nous sommes basés avec leurs nombres de neurones respectifs. Les courants ioniques modélisés sont donnés dans la deuxième colonne.

5.2.1 Boucle GPe-(MSN-D2)

Nous nous sommes appuyés sur les connexions étudiées lors des expérimentations de l'article [10]. Cependant, nous avons ajouté au modèle les connexions GPeA-(MSN-D2) et (MSN-D2)-(MSN-D2) et avons retiré la connexion (MSN-D2)-GPeA. En effet, d'après l'article [10], les auteurs ont aussi conclu que l'entrée MSN-D2 avait un impact plus fort sur les neurones du GPeP que pour celui du GPeA, i.e. les neurones du GPeA recevaient des projections plus faibles et moins nombreuses du MSN-D2 que le GPeP.

D'après Mallet et al. [69, 71] et Abdi et al [7], les neurones du GPeA se projettent massivement vers le STR. Mallet et al. [71] ont même montré que l'activité des neurones du GPeA augmenterait fortement lors d'un signal STOP à une souris où le STR est complètement inactif. Nous avons donc ajouté cette connexion entre le GPeA et le MSN-D2.

De plus, Tepper et al. [89] et McCarthy et al. [72] montrent, lors d'expériences, qu'il existe une connexion MSN-MSN. Cette connexion renforcerait les oscillations pathologiques sur le MSN quand il y a une perte en dopamine. Nous avons alors jugé important d'ajouter cette connexion à notre modèle de neurones.

5.2.2 Boucle STN-GPe

Selon Aristieta et al. [10], le STN excite préférentiellement le GPeP plutôt que le GPeA, mais le GPeP inhibe à son tour le GPeA. Ainsi, pour le GPeA, il est probable, d'après ces résultats, que l'inhibition du GPeP sur le GPeA l'emporte sur l'excitation du STN sur le GPeA. C'est pourquoi, sur la Fig. 5.5, nous avons noté que la synapse STN-GPeA est plus faible, ligne en pointillé, par rapport à la synapse GPeP-GPeA.

En ce qui concerne la connexion GPeP-STN, Abdi et al.[7] ont révélé que le GPeP inhibe fortement le STN. Cependant le GPeA a une très faible influence sur le STN (d'environ 20 fois moins importante que le GPeP). Ces résultats justifient le fait d'avoir modélisé la connexion GPeP-STN et pas celle de GPeA-STN.

Nous avons choisi les connexions à modéliser en fonction des données expérimentales, mais aussi il est important de savoir que plus nous modélisons de connexions synaptiques plus le temps

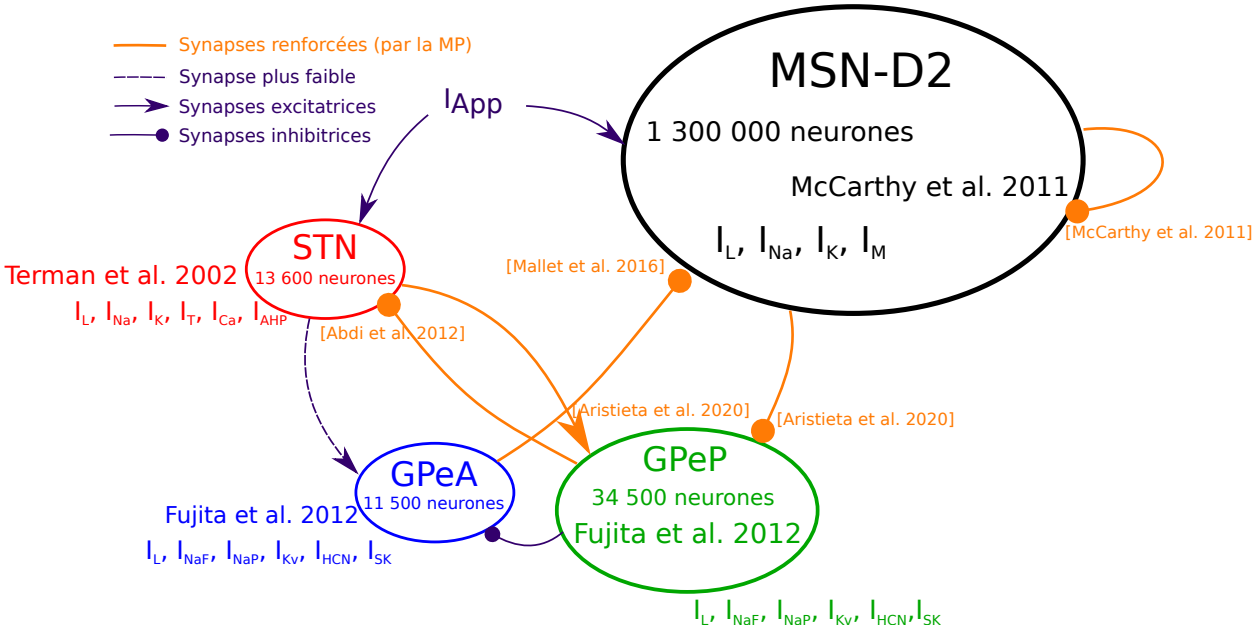

Fig. 5.5 – Schéma du modèle de réseau des GB. Chaque cercle représente une population de neurones ; rouge pour le STN, bleu pour GPeA, vert pour GPeP et le noir pour STR (MSN-D2). Le trait en pointillé représentent des connexions à plus faibles intensité, les traits en orange sont les connexions renforcées par la MP, les flèches avec un rond sont les synapses inhibitrices et celles avec une flèche sont les synapses excitatrices.

de simulation s'allonge, ainsi nous avons fait notre choix en fonction des données expérimentales. Ensuite, nous avons adapté les conductances données par l'article afin de reproduire les mêmes comportements que l'article [10], i.e. les Fig.5.1 à 5.4.

5.2.3 Modélisation des effets de la maladie de Parkinson sur le réseau STN-GPeA-GPeP-MSN

Dans la Fig. 5.5, les synapses renforcées par la MP sont en orange et en gras. Nous nous sommes inspirés des articles mentionnés précédemment à la sous-section 5.1.5 afin de choisir les bonnes connexions à renforcer. En effet, nous avons augmenté les connexions GPeP-STN, GPeA-(MSN-D2), (MSN-D2)-(MSN-D2) et (MSN-D2)-GPeP par une valeur donnée dans la prochaine sous-section. La conductance du courant M de la population MSN-D2 a été diminuée, de $1,34nS$ à $1,1nS$, comme dans l'article [72].

Nous avons testé des hypothèses sur les connexions synaptiques, et la propagation des oscillations pathologiques sur le réseau, et aussi sur les propriétés intrinsèques du MSN-D2. D'après McCarthy et al. [72], les oscillations pathologiques sont générées dans le MSN-D2 par les connexions (MSN-D2)-(MSN-D2) et le canal M. Nous allons retirer la connexion (MSN-D2)-(MSN-D2) dans un premier temps et dans un deuxième temps, nous modifierons le canal M afin de voir si les oscillations pathologiques sont toujours présentes dans le réseau. Ensuite, nous avons supprimé la connexion (MSN-D2)-GPeP afin de voir si cette voie propage les oscillations sur le reste du réseau ou si c'est la boucle STN-GPe qui est à l'origine de ces oscillations.

5.2.4 Conductances synaptiques

Les conductances synaptiques de l'article [10] ont dû être adaptées. Nous aurions pu les adapter avec l'algorithme DE, en changeant la fonction coût, afin de reproduire les données expérimentales. Mais nous avons commencé à ajuster empiriquement les conductances et nous avons réussi à reproduire les résultats expérimentaux. Par conséquent, la conductance synaptique GPeP-GPeA a été diminuée, sinon le GPeA était trop inhibé par le GPeP. Ensuite, il fallait que le GPeA soit plus influencé par le GPeP tout en étant impacté par l'excitation du STN. Ainsi, la conductance STN-GPeA a été réduite et celle de GPeP-GPeA a été augmentée afin de retrouver les deux comportements, des Figs. 5.2 et 5.3, dans nos simulations. Les conductances utilisées dans notre modèle sont représentées dans le Tab. 5.3.

Conductances synaptiques $g_{syn,XY}$ (en nS)				
X \ **Y**	**STN**	**GPeA**	**GPeP**	**MSN-D2**
STN	-	0,5	10,0	-
GPeA	-	-	-	0,3
GPeP	0,2	2,5	-	-
MSN-D2	-	-	9,0	0,03

Tab. 5.3 – Conductances synaptiques de notre modèle des GB.

Chaque conductance synaptique a été mise à l'échelle en divisant par le nombre de connexions synaptiques avant d'être appliquée au courant synaptique. Les conductances renforcées ont été choisies en fonction de la littérature, vu à la sous-section 5.1.5. Dans le Tab. D.5 de l'annexe, nous indiquons le nombre de connexions synaptiques entre chaque population de neurones.

Dans le but de simuler la condition parkinsonienne, des boucles ont été renforcées en multipliant certaines conductances synaptiques par un facteur entre 1.4 et 10 comme indiqué dans le Tab. 5.4. La Fig. 5.5 et le Tab. 5.4 montre les différentes boucles renforcées en orange. Les conductances les plus renforcées, pour la condition parkinsonienne, sont celles qui inhibent le MSN-D2.

Conductances synaptiques renforcées $g_{syn,XY}$ (en nS)				
X \ **Y**	**STN**	**GPeA**	**GPeP**	**MSN-D2**
STN	-	0,5	5 * 10,0	-
GPeA	-	-	-	20 * 0,3
GPeP	2 * 0,2	2,5	-	-
MSN-D2	-	-	1.4 * 9,0	20 * 0,03

Tab. 5.4 – Conductances synaptiques renforcées lors de la MP du modèle des GB.

5.3 Résultats

Cette section comporte deux parties. La première partie est consacrée à la reproduction des résultats de l'article de Aristieta et al. [10] et la deuxième porte sur la simulation du réseau

complet, à l'échelle du rat, soit environ 1 360 000 neurones, en conditions saine et parkinsonienne et sur la vérification de quelques hypothèses sur l'origine et la propagation des synchronisations pathologiques.

5.3.1 Contexte expérimental

Les simulations présentées ci-dessous ont été réalisées sur `Grid'5000` sur le site de Nancy avec le cluster `grvingt` avec le logiciel SiReNe. `grvingt` est sous Linux Debian 4.19 amd64 avec 2 CPUs Intel Xeon Gold 6130 avec 16 coeurs chacun, et 192GB de RAM. Les temps ont été mesurés avec la fonction d'OpenMP, `omp_get_wtime()`, et le programme a été compilé avec `gcc` 8.3.0 et le niveau d'optimisation `-O3`.

Paramètres de simulation			
Paramètres généraux	Type d'interpolation		Courbe de Bézier
	Méthode numérique		RK2
	Taille du réseau	STN	136
		GPeA	115
		GPeP	345
		MSN-D2	13 000
	Temps biologique $[ms]$		8000
	Pas de temps $[ms]$		5e-3
	# thread(s)		16

Tab. 5.5 – Liste des paramètres de simulation du réseau complet.

Le Tab. 5.5 donne la liste des paramètres de simulation lorsque nous avons reproduit les résultats de l'article [10]. Le temps de simulation biologique est de $8s$, mais les 2 premières secondes correspondent à un régime de transition et ne sont pas prises en compte dans l'analyse des résultats. Les paramètres d'initialisation du STN sont expliqués dans l'annexe D.1.1 et pour les autres populations de neurones, les paramètres ont été donnés dans les chapitres précédents. Une hétérogénéité de $\pm 10\%$ a été ajoutée à chaque paramètre d'initialisation afin que chaque neurone ait des initialisations différentes. Les différents courants appliqués sont donnés dans l'annexe D.1.2. Les paramètres synaptiques de chaque connexion sont présentés dans l'annexe D.1.3.

5.3.2 Reproduction des résultats de l'article [10]

La moyenne de PA d'une population de neurones a été calculée tous les $50ms$ et les fréquences de décharge ont été tracées. Les valeurs à droite des figures sont la moyenne des PA pour une population de neurones sur une durée de $2s$ avant la stimulation, resp. l'inhibition de la population, et après la stimulation, resp. l'inhibition de la population.

La Fig. 5.6 reproduit les résultats de la Fig. 5.1. Le MSN-D2 est excité par un courant extérieur excitateur entre 4 et 6 secondes. Nous observons que le GPeP est fortement inhibé par la désinhibition du MSN-D2. Comme le GPeP est inhibé, alors le STN et GPeA ne sont plus inhibés par le GPeP et sont donc excités. Ainsi, le GPeA et le GPeP montrent une activité opposée l'une à l'autre, comme vu dans l'article [10].

La Fig. 5.7 simule la Fig. 5.2 de l'article [10]. Le STN est inhibé par un courant extérieur inhibiteur entre 4 et 6 secondes. Nous constatons que le GPeP est légèrement inhibé puisqu'il

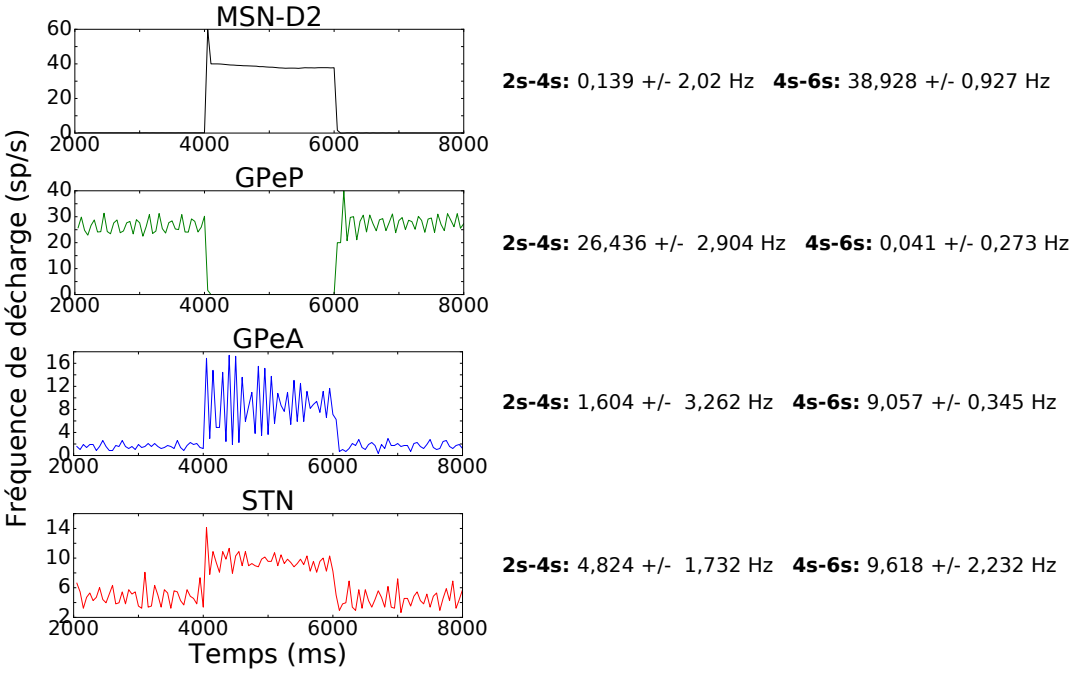

Fig. 5.6 – Fréquences de décharge du réseau lors d'une excitation du MSN-D2.

ne reçoit plus d'excitation du STN. Ainsi, le GPeA n'est plus inhibé par le GPeP et donc nous remarquons une légère hausse de la fréquence de décharge du GPeA.

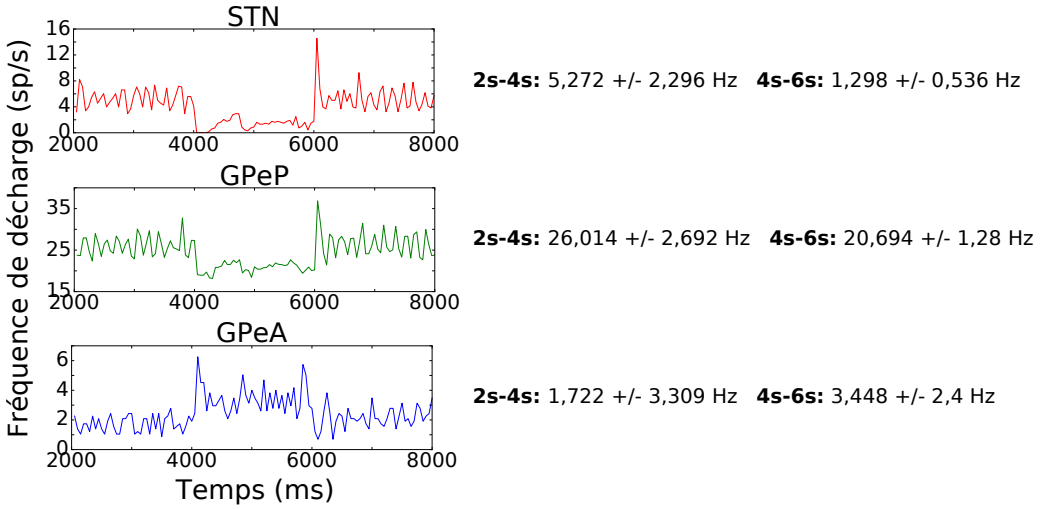

Fig. 5.7 – Fréquences de décharge du réseau lors d'une inhibition du STN.

La Fig. 5.8 reproduit les résultats de la Fig. 5.3 de l'article [10]. Nous appliquons un courant excitateur au STN entre 4 et 6 secondes. Le GPeP est fortement excité par le STN. Ainsi, le GPeA est à son tour fortement inhibé par le GPeP. Ainsi, le GPeA et le GPeP montrent une activité opposée l'un à l'autre, comme vu dans l'article [10].

Les valeurs des fréquences de décharge ne sont pas exactement les mêmes que sur les données expérimentales [10]. Cette différence s'explique par le fait que les conditions expérimentales et celles de simulation ne sont pas exactement les mêmes. D'un côté, nous ne modélisons pas tous

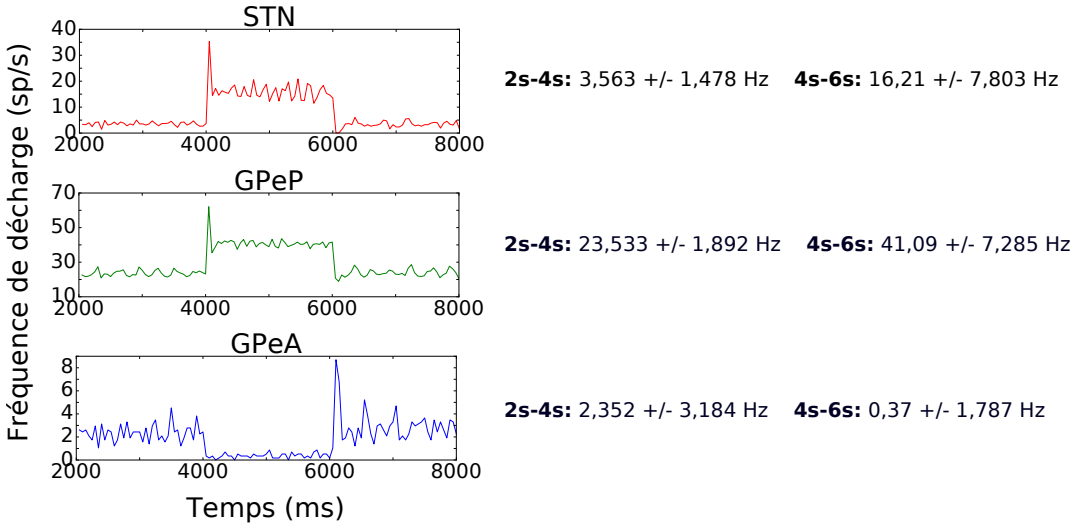

Fig. 5.8 – Fréquences de décharge du réseau lors d'une excitation du STN.

les canaux ioniques et les connexions entre chaque population, comme mentionné à la section 5.2, et de l'autre, les données de ces expériences ont été recueillies sur des tranches de cerveaux et inévitablement des connexions sont coupées et seulement une partie des neurones est étudiée. Nous, de notre côté, nous modélisons 1/100 des GB chez le rat, soit environ 13 600, avec le nombre de connexions tiré de l'article [64] en condition normale.

En conclusion, nous avons réussi à reproduire les mêmes comportements sur notre réseau que l'article [10]. Nous avons ainsi validé notre modèle de GB, et plus précisément les connexions synaptiques de notre modèle. Nous pouvons maintenant simuler les conditions saine et parkinsonienne à l'échelle une du rat, i.e. environ 1 360 000 neurones.

5.3.3 Simulation du réseau des ganglions de la base

Dans cette sous-section, les résultats du réseau complet des GB en conditions saine, et parkinsonienne, sont exposés.

Contexte expérimental

Les simulations présentées ci-dessous ont été réalisées sur `Grid'5000` sur le site de Nancy avec le cluster `gruss`. `gruss` est sous Linux Debian 4.19 amd64 avec 2 CPUs AMPD EPYC 7352 avec 24 coeurs chacun, et 256GB de RAM. Les temps ont été mesurés avec la fonction d'OpenMP, `omp_get_wtime()`, et le programme a été compilé avec `gcc` 8.3.0 et le niveau d'optimisation `-O3`.

Le Tab. 5.6 donne la liste des paramètres de simulation lorsque nous avons simulé le réseau complet des GB à l'échelle une du rat, i.e. environ 1 360 000. Le temps de simulation biologique est de $4s$, mais les 2 premières secondes ne sont pas prises en compte. Les paramètres d'initialisation du réseau sont les mêmes que pour les simulations précédentes. Les différents courants appliqués sont donnés dans l'annexe D.2.1 et les paramètres synaptiques de chaque connexion sont les mêmes que précédemment, voir l'annexe D.1.3, sauf pour la condition parkinsonienne où les conductances synaptiques sont renforcées, voir Tab. 5.4 de la section précédente.

Le temps de simulation est d'environ $4h30$ par seconde de temps biologique avec un pas de temps de $0,005ms$. En comparaison, pour notre article [13] sur la méthode hybride, nous avions un temps de simulation de 10h par seconde de temps biologique avec un pas de temps plus grand,

Paramètres de simulation			
Paramètres généraux	Type d'interpolation		Courbe de Bézier
	Méthode numérique		RK2
	Taille du réseau	STN	13600
		GPeA	11500
		GPeP	34500
		MSN-D2	1 300 000
	Temps biologique $[ms]$		4000
	Pas de temps $[ms]$		5e-3
	# thread(s)		32

Tab. 5.6 – Liste des paramètres de simulation du réseau complet.

i.e. $0,01ms$, et nous ne simulions que le STR, soit 1 300 000 neurones MSN-D2. Le cluster utilisé cette fois-ci est plus récent (2021), donc il a une vitesse de calcul plus importante, et quelques améliorations ont été apportées à SiReNe entre temps. La mémoire allouée est d'environ $2,2GB$.

Dans les Figs. 5.9 et 5.11, montrant les simulations entre $2s$ et $4s$, nous affichons uniquement 1400 neurones, i.e. 136 neurones du STN, 115 neurones du GPeA, 345 neurones du GPeP et le reste des neurones du MSN-D2.

Condition saine

En condition saine, Fig. 5.9, nous remarquons aucune synchronisation pathologique. Les fréquences de décharge sont affichées à droite de la Fig. 5.9.

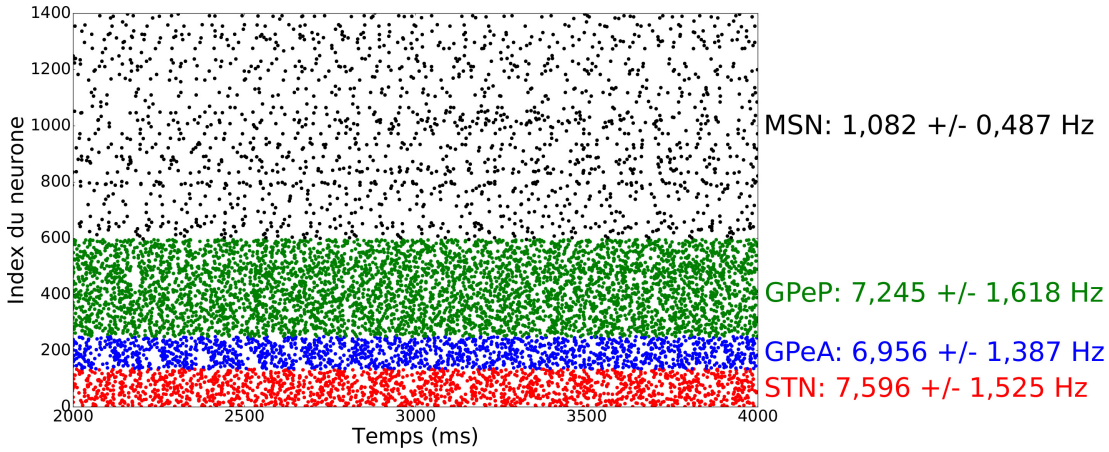

Fig. 5.9 – Tracé des temps des PA des neurones du réseau STN-GPeA-GPeP-(MSN-D2) à chaque pas de temps en condition saine. Simulation du réseau à l'échelle du rat, i.e. environ 1 360 000 de neurones, pendant $4s$ de temps biologique, affichant uniquement 1400 neurones, i.e. 136 neurones du STN, 115 neurones du GPeA, 345 neurones du GPeP et le reste des neurones du MSN-D2.

Dans la Fig. 5.10, la puissance spectrale du LFP a été tracée en fonction de la fréquence. En condition saine, l'analyse spectrale ne montre aucun pic significatif (le réseau n'est globalement pas synchronisé). Seul un petit pic est observé sur la population du STN et du GPeA. Le pic est observé à $10Hz$, indiquant un faible état de synchronisation. Sur les deux autres populations

aucun pic n'est vraiment visible.

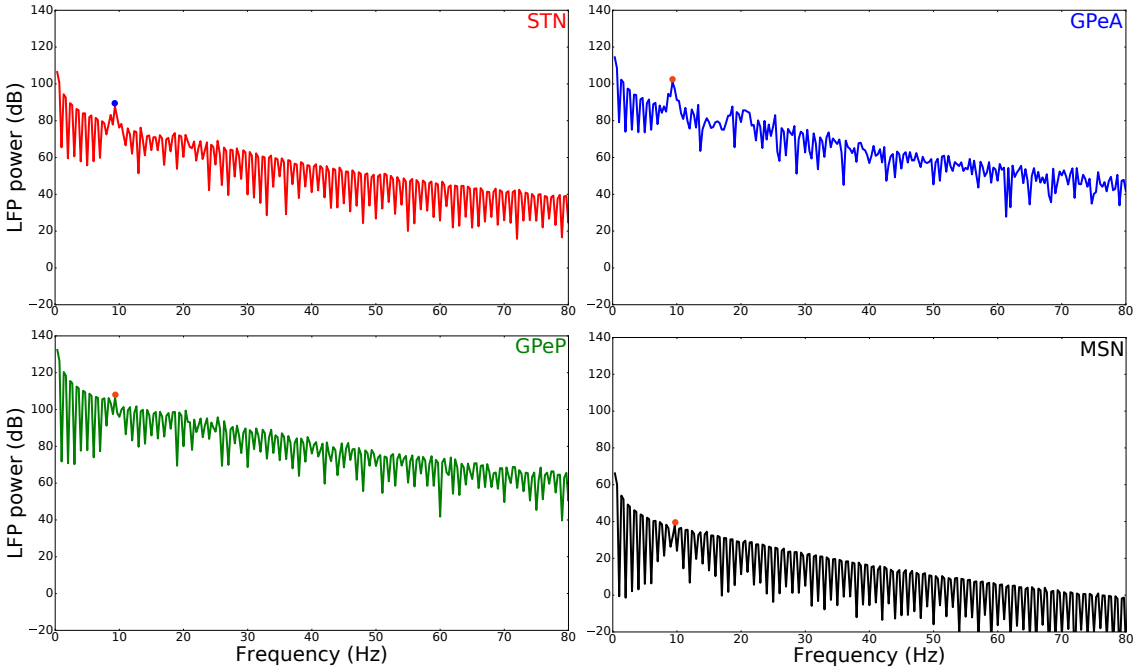

Fig. 5.10 – Puissance du LFP en fonction de la fréquence en condition saine lors de la simulation du réseau à l'échelle du rat, i.e. environ 1 360 000 de neurones, pendant $4s$ de temps biologique.

Condition parkinsonienne

En condition parkinsonienne (Fig. 5.11) nous remarquons une forte synchronisation du réseau. Cette synchronisation pathologique est moins marquée sur le GPeA. Les fréquences de décharge sont affichées à droite de la Fig. 5.11, qui ont fortement augmenté pour les populations du MSN-D2 et du GPeP, diminuant ainsi celles du GPeA et STN.

Dans la Fig. 5.12, la puissance spectrale du LFP a été tracée en fonction de la fréquence. En condition parkinsonienne, nous observons un pic pour toutes les populations de neurones à environ $22Hz$, ce qui représente bien les oscillations pathologiques parkinsoniennes, i.e. des fortes synchronisations neuronales par rapport aux PA, dans la bande β [26, 72].

Hypothèses

Afin de tester des hypothèses rapidement, nous avons simulé un plus petit réseau à l'échelle 1/100 du rat, i.e. environ 13 600 de neurones sur une durée de $4s$ de temps biologique.

Notre première hypothèse porte sur l'origine des synchronisations, donc à cet effet nous avons modifié le canal M du MSN-D2. Comme dit précédemment, le courant M peut être partiellement bloqué lors de la MP [72], conduisant ainsi à l'amplification de la dynamique du STR. Nous avons ainsi augmenté la conductance du courant M, passant de $1,1nS$ (condition parkinsonienne) à $3nS$ afin de montrer l'arrêt des synchronisations pathologiques du réseau MSN. La Fig. 5.13 montre que lorsque nous augmentons la conductance du courant M, nous n'observons quasiment aucune synchronisation pathologique du MSN-D2.

Dans la deuxième hypothèse, nous avons retiré la connexion (MSN-D2)-(MSN-D2), afin de voir, si le MSN-D2 est toujours synchronisé après cette suppression. Dans la Fig. 5.14, nous

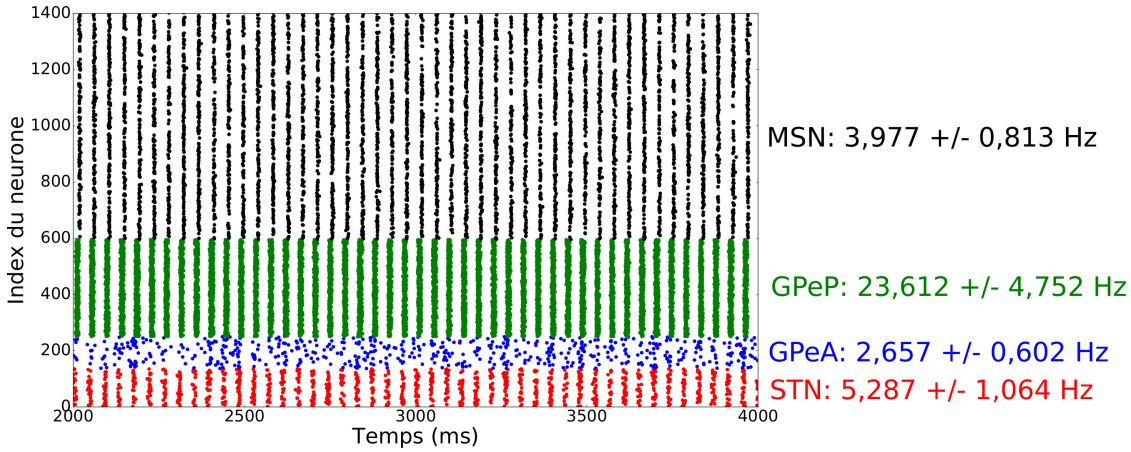

Fig. 5.11 – Tracé des temps des PA des neurones du réseau STN-GPeA-GPeP-(MSN-D2) à chaque pas de temps en condition parkinsonienne. Simulation du réseau à l'échelle du rat, i.e. environ 1 360 000 de neurones, pendant 4*s* de temps biologique, affichant uniquement 1400 neurones, i.e. 136 neurones du STN, 115 neurones du GPeA, 345 neurones du GPeP et le reste des neurones du MSN-D2.

constatons que le MSN-D2 n'est plus synchronisé. Cependant une légère synchronisation pathologique est observée à la fin de la simulation. Il faudrait creuser un peu plus à ce niveau afin de voir si les synchronisations pathologiques s'intensifient à plus long terme. Toutefois aucune synchronisation pathologique est visible au niveau du GPeA ou du STN.

La connexion (MSN-D2)-GPeP a été supprimée dans la troisième hypothèse afin de voir si les synchronisations pathologiques du MSN-D2 étaient propagées sur le reste du réseau ou si les synchronisations sont toujours présentes. Dans la Fig. 5.15, nous remarquons que le GPeP n'est quasiment plus synchronisé par contre nous retrouvons encore un peu de synchronisation au niveau du STN. Donc cela nous a amenés à notre derrière hypothèse.

Sur cette dernière hypothèse, nous avons enlevé la connexion GPeP-STN et lorsque celle-ci est retirée le STN n'est plus synchronisé.

Pour résumer, chaque connexion, que nous avons renforcé sur notre modèle, joue un rôle très important sur la synchronisation pathologique caractéristique de la MP. (MSN-D2)-(MSN-D2) permet au MSN-D2 de synchroniser, comme le démontre aussi McCarthy et al. [72]. Toutefois, la boucle STN-GPeP joue aussi un rôle sur les synchronisations pathologiques, mais celle-ci a une influence plus importante sur la structure du STN et du GPe. La connexion (MSN-D2)-GPeP propage ces oscillations vers le reste du réseau, et plus particulièrement vers le GPeP. La connexion GPeP-STN propage ces synchronisations vers le STN et donc les synchronisations au niveau du STN s'intensifie. D'après notre modèle en condition parkinsonienne, le réseau MSN-D2 est à l'origine des synchronisations pathologiques qui ensuite sont propagées sur le reste du réseau par le GPeP. Néanmoins, la boucle STN-GPeP est aussi propice aux synchronisations pathologiques, mais moins que le réseau MSN-D2.

Par manque de temps, nous n'avons pas pu étudier le rôle du GPeA dans ce réseau. Il est probable que la connexion (MSN-D2)-(MSN-D2) est un peu trop renforcé, ce qui nous ne permet pas de voir si la connexion GPeA-(MSN-D2) amplifie les oscillations pathologiques lors de la MP. De plus, une étude du LFP nous aurait permis une meilleur compréhension de nos résultats, mais malheureusement par manque de temps aussi, cette étude n'a pas été possible.

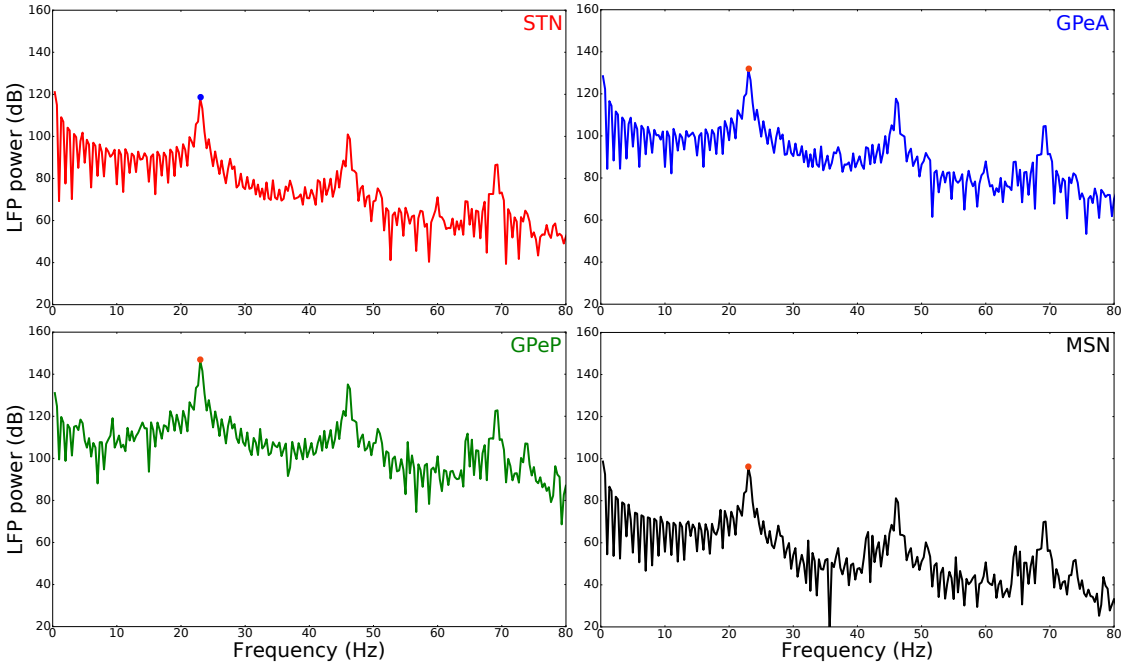

Fig. 5.12 – Puissance du LFP en fonction de la fréquence en condition parkinsonienne lors de la simulation du réseau à l'échelle du rat, i.e. environ 1 360 000 de neurones, pendant 4s de temps biologique.

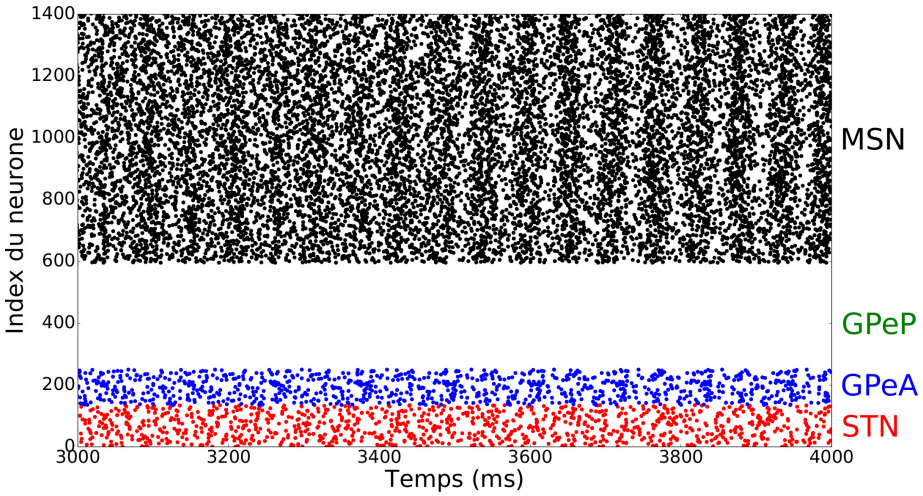

Fig. 5.13 – Tracé des temps des PA des neurones du réseau STN-GPeA-GPeP-(MSN-D2) à chaque pas de temps en condition parkinsonienne avec modification du canal M du MSN-D2. Simulation du réseau à l'échelle 1/100 du rat, i.e. environ 13 600 de neurones, pendant 4s de temps biologique, affichant uniquement 1400 neurones, i.e. 136 neurones du STN, 115 neurones du GPeA, 345 neurones du GPeP et le reste des neurones du MSN-D2.

5.4 Conclusion

Dans cette section, nous avons modélisé le réseau complet des GB avec la distinction des deux populations du GPe. La force de notre modèle, par rapport à des modèles déjà existants,

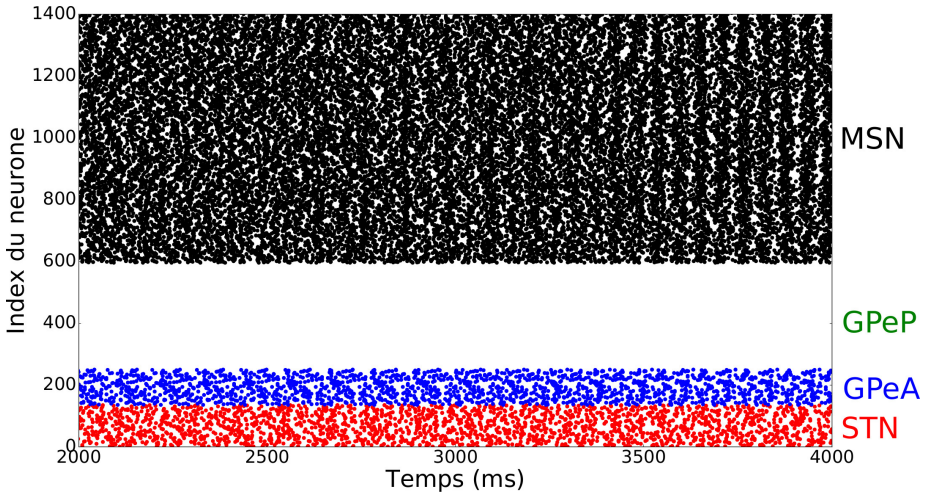

Fig. 5.14 – Tracé des temps des PA des neurones du réseau STN-GPeA-GPeP-(MSN-D2) à chaque pas de temps en condition parkinsonienne sans la connexion (MSN-D2)-(MSN-D2). Simulation du réseau à l'échelle 1/100 du rat, i.e. environ 13 600 de neurones, pendant 4s de temps biologique, affichant uniquement 1400 neurones, i.e. 136 neurones du STN, 115 neurones du GPeA, 345 neurones du GPeP et le reste des neurones du MSN-D2.

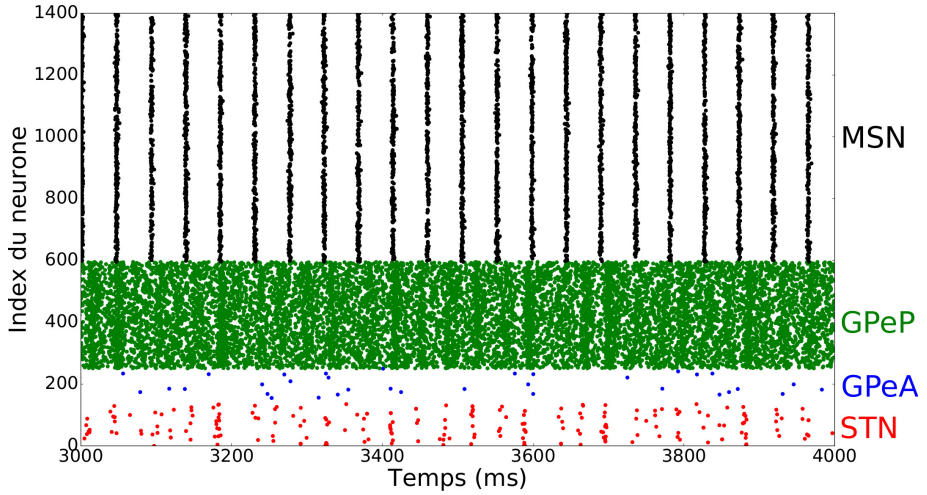

Fig. 5.15 – Tracé des temps des PA des neurones du réseau STN-GPeA-GPeP-(MSN-D2) à chaque pas de temps en condition parkinsonienne sans la connexion (MSN-D2)-(GPeP). Simulation du réseau à l'échelle 1/100 du rat, i.e. environ 13 600 de neurones, pendant 4s de temps biologique, affichant uniquement 1400 neurones, i.e. 136 neurones du STN, 115 neurones du GPeA, 345 neurones du GPeP et le reste des neurones du MSN-D2.

est que non seulement nous avons respecté les proportions de neurones entre chaque population, mais aussi le nombre de connexions synaptiques. La modélisation par le formalisme d'HH nous a permis de valider notre modèle par des données expérimentales, mais aussi nous permettra de tester des hypothèses intrinsèques aux neurones, comme le rôle de certains canaux ioniques sur la synchronisation pathologique lors de la MP.

L'ajustement de notre modèle aux données expérimentales de Aristieta et al. [10], nous a permis d'affiner notre modèle. Nos hypothèses en condition parkinsonienne, nous ont aidé à

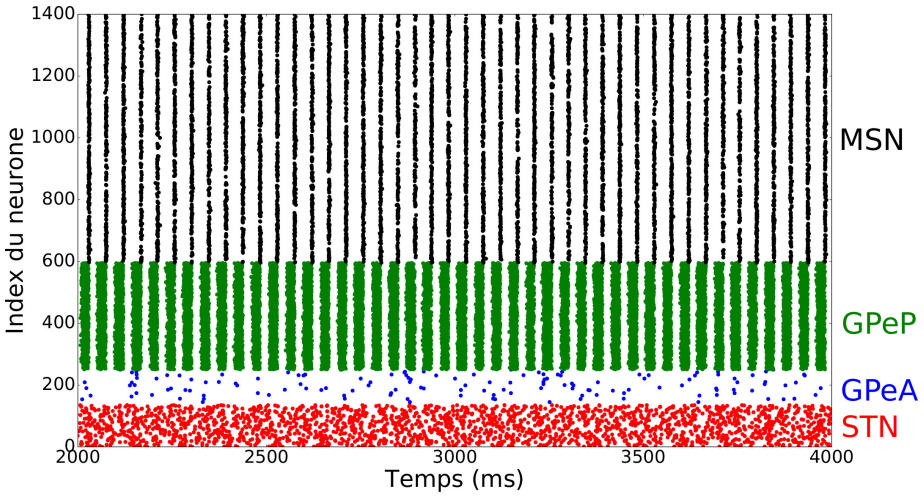

Fig. 5.16 – Tracé des temps des PA des neurones du réseau STN-GPeA-GPeP-(MSN-D2) à chaque pas de temps en condition parkinsonienne sans la connexion (MSN-D2)-(STN). Simulation du réseau à l'échelle 1/100 du rat, i.e. environ 13 600 de neurones, pendant $4s$ de temps biologique, affichant uniquement 1400 neurones, i.e. 136 neurones du STN, 115 neurones du GPeA, 345 neurones du GPeP et le reste des neurones du MSN-D2.

comprendre que potentiellement le MSN-D2 est à l'origine des synchronisations pathologiques, comme le soutient aussi McCarthy et al. [72], mais aussi que les boucles renforcées par la maladie sont importantes pour la propagation des oscillations pathologiques.

Conclusion

Cette thèse porte sur deux axes majeurs. Le premier axe est le développement d'un logiciel de simulation rapide, utilisant des méthodes de simulation précises, capable de simuler des grands réseaux de neurones de type Hodgkin-Huxley. Le deuxième axe est la modélisation des ganglions de la base chez le rat et sa validation par des données expérimentales en conditions saine et parkinsonienne. Le but est, à l'avenir, de pouvoir tester et valider en simulation des hypothèses ou/et des traitements, e.g. sur la maladie de Parkinson. Cette section présente les conclusions générales de la thèse.

Simulateur SiReNe

Nous nous sommes appuyés sur le formalisme d'Hodgkin-Huxley, afin de modéliser les différents canaux ioniques de neurones impliqués dans les ganglions de la base affectés lors de la maladie de Parkinson. Ce choix nous a permis de simuler un réseau de neurones proche de la dynamique neuronale biologique. Toutefois, nous avons vu que la quantité de mémoire nécessaire au stockage explicite de la connectivité d'un réseau de neurones augmente quasiment de manière quadratique avec le nombre de neurones. De plus, le temps de simulation est conséquent, compte tenu du choix du formalisme d'Hodgkin-Huxley (grand nombre d'équations différentielles ordinaires par neurones). Le choix de la méthode numérique est aussi important afin de suivre le système d'équations, lent-rapide, décrivant la dynamique neuronale. La précision des temps des potentiels d'action est essentielle de manière à bien se rendre compte des synchronisations pathologiques, caractéristiques de la maladie de Parkinson.

Ainsi, dans le chapitre 2, une nouvelle approche de simulation de réseaux de neurones de très grandes tailles a été présentée. Nous avons proposé une approche évitant le stockage de la connectivité, afin de réduire considérablement la consommation en mémoire lors des simulations. Les neurones postsynaptiques sont générés à chaque nouvel événement, i.e. lorsqu'un neurone émet un potentiel d'action, et leurs courants synaptiques respectifs sont actualisés. Par conséquent, nous évitons le stockage de la connectivité, car nous la générons pseudo aléatoirement à chaque événement. Ainsi, l'allocation de la mémoire est fortement réduite, ce qui permet de simuler des réseaux de neurones de très grandes tailles.

Cette nouvelle approche a été implémentée dans **SiReNe**, un logiciel développé dans le laboratoire LORIA-INRIA, présenté au chapitre 3. Nous avons aussi intégré le calcul parallèle et plus particulièrement le multi-threading avec OpenMP et une version multi-machines avec MPI. Avec OpenMP, nous avons réussi à réduire de manière significative le temps de simulation. Cependant, cette accélération est nécessairement limitée par le nombre de cœurs disponibles dans la machine utilisée. Avec MPI, la grande quantité de communications entre les machines, induite par les connexions synaptiques entre les neurones, représente un obstacle à l'obtention d'une efficacité optimale. Cependant, au-delà des performances absolues, cette approche multi-machines permet la simulation de réseaux de très grande taille pour lesquels la mémoire d'une seule machine ne suffirait pas.

Modélisation des ganglions de la base

Dans la littérature, plusieurs modèles des ganglions de la base ont été proposés. Ces modèles considèrent uniquement une population de neurones inhibiteurs du globus pallidus externe et ne font pas la distinction entre les neurones arkypallidaux et prototypiques. Dans les chapitres 4 et 5, nous montrons l'importance de distinguer les deux populations de neurones du globus pallidus externe. D'un côté, ces deux populations de neurones, se projettent sur des structures différentes et, de l'autre, elles présentent des fréquences de décharge différentes en présence de courants injectés. De plus, elles ont une activité opposée lorsque nous analysons l'ensemble du réseau des ganglions de la base.

La simulation de plus d'un million de neurones du type Hodgkin-Huxley est coûteuse en temps de calcul. Cela nous a contraint à valider notre modèle avec environ 13 600 de neurones avant de passer à la simulation du réseau complet. Les modèles présentés en [26, 72] ont été validés avec une centaine de neurones en ne respectant ni la proportion de neurones ni les connexions synaptiques. Nous avons fait le choix de valider notre modèle sur des données biologiques [7, 10] en respectant les proportions de neurones et la connectivité synaptique.

Par la suite, les hypothèses, que nous avons testées en simulations, montrent que le striatum (MSN-D2) est à l'origine des synchronisations pathologiques qui sont ensuite propagées à tout le réseau via les boucles (MSN-D2)-GPeP et GPeP-STN. Pour finir, dans le chapitre 5, nous avons simulé le réseau à l'échelle une du rat, i.e. environ 1 360 000 neurones, en conditions saine et parkinsonienne. En condition parkinsonienne, nous observons une forte synchronisation du réseau à environ 22Hz (bande β). En condition saine, aucune synchronisation est observé.

Perspectives

Concernant **SiReNe**, nous avons vu que la complexité des interconnexions entre neurones reste une contrainte forte pour l'obtention d'une efficacité élevée de la version multi-machines utilisant MPI. Une seconde version multi-machines est en cours de développement. Son principe est d'une part de réduire les volumes des communications, et d'autres part de recouvrir une partie des communications par des calculs. Les premiers tests sont prometteurs et montrent une amélioration significative des performances.

Un autre objectif est de mettre **SiReNe** à la portée de la communauté des neuro-sciences. Ainsi, il faudrait approfondir l'étude sur la définition d'un langage de description non seulement adapté à **SiReNe**, mais aussi aux autres simulateurs.

Maintenant que le modèle a été validé par des données expérimentales, l'objectif est de tester des résultats expérimentaux, il est possible d'étudier le rôle du GPeA dans le réseau et aussi de tester des hypothèses à propos des propriétés intrinsèques des neurones comme par exemple le rôle du canal SK dans la régularité des potentiel d'action [29] et/ou le rôle du canal M sur les synchronisations pathologiques dans le MSN-D2.

Les MSN-D1 et MSN-D2 sont impliqués respectivement dans l'exécution et l'inhibition du mouvement [71]. Ainsi, avec notre modèle nous aurons la possibilité de comparer plusieurs schémas de connectivité des GPeA sur les MSN, i.e. le GPeA connecté de façon indifférencié au MSN, ou seulement au MSN-D1 [10] ou seulement au MSN-D2.

D'autres idées pourront être exploitées comme le rôle des populations MSN-D1 et MSN-D2 dans la génération et la propagation des oscillations pathologiques suivant les voies directes et indirectes. Cette étude permettrait notamment d'identifier l'origine des rythmes pathologiques afin de cibler précisément la, ou les, structure(s) qui serait(ent) pertinente(s) de stimuler pour développer un modèle de stimulation profonde en boucle fermée. Plus précisément, nous pour-

rions étudier la connectivité du GPeA vers les neurones dopaminergique D1 et savoir si cette connexion est renforcée en conditions pathologiques, ce qui pourrait expliquer au moins en partie les mécanismes de «freezing» observés dans la maladie de Parkinson [10].

Un autre objectif de notre modèle est de développer de nouvelles approches de stimulation cérébrale profonde en boucle fermée. Pour cela, il est nécessaire d'étendre notre modèle en prenant en compte l'aspect spatial, e.g. la forme ellipsoïdale des ganglions [35] et de considérer la distance entre l'électrode de stimulation et chaque neurone en suivant les principes de modélisation proposés en [12].

Bilan

Le développement du logiciel **SiReNe**, et la validation du modèle des ganglions de la base ont pris plus de temps que prévu. La recherche d'une méthode, permettant de garder la précision du calcul numérique sans que le temps de simulation soit trop impacté, n'était pas aussi facile à trouver que ce que nous pensions. Nous avons cherché plusieurs méthodes d'interpolation pour le calcul du temps du potentiel d'action, mais il n'était pas facile de trouver une méthode qui garde la précision de la méthode numérique de Runge-Kutta 2. De plus, la méthode devait définir une courbe avec comme seules informations 2 points et 2 dérivées, voir le deuxième chapitre.

Ensuite, la validation du modèle est passée par trois étapes ; la sélection des canaux ioniques à modéliser, la recherche des données expérimentales validant le modèle et l'optimisation des paramètres. Ces étapes n'ont pas toujours été faciles à réaliser et ont nécessité plusieurs mois de travail non négligeable, mais qui nous ont permis d'obtenir un modèle très satisfaisant. Il n'est pas facile de trouver des articles fournissant les données expérimentales que nous cherchons, comme les courbes FI, les courbes des différents courants ioniques ou des courbes de LFP. Quant à l'optimisation des paramètres, il a fallu réadapter plusieurs fois les intervalles de domaine, afin d'ajuster au mieux la courbe FI des deux populations du GPe. Cependant, la collaboration avec Jérôme Baufreton, neurophysiologiste, nous a permis de nous orienter vers des solutions biologiquement plus réalistes lorsque nous n'arrivions pas à ajuster nos paramètres.

Notre modèle est maintenant validé par des résultats expérimentaux et peut être utilisé à l'avenir pour tester des hypothèses. Le logiciel **SiReNe** est quant à lui disponible à tout le monde et opérationnel pour la simulation de grands réseaux de neurones.

La modélisation de phénomènes réels est un atout pour l'avenir. Néanmoins, je trouve que nous ne donnons pas encore assez d'importance à ce type de recherche qui est tout autant méritant que d'autres approches, notamment celles basées sur l'intelligence artificielle. L'avantage avec la modélisation mathématique est que nous pouvons étudier, ou tester, des hypothèses, mais aussi comprendre les mécanismes internes du fonctionnement des réseaux de neurones, sans pour autant avoir des données expérimentales.

Je suis convaincue qu'une collaboration plus étroite, entre chercheurs en physiologie et en modélisation, permettrait de limiter les expérimentations animales et de respecter les actuelles normes européennes. Pendant ma thèse, j'ai eu l'opportunité de travailler avec Jérôme Baufreton, directeur de recherche CNRS à l'IMN de Bordeaux, et notre collaboration m'a permis de comprendre que d'un côté, son point de vue d'expérimentateur m'a permis d'améliorer mon modèle, et d'un autre côté, un modèle, comme le nôtre leur permettra de tester des hypothèses avant de passer à l'expérimentation. En effet, certaines expérimentations ne sont pas très faciles à réaliser, ou trop longue à valider, et nécessitent plusieurs mois de travail avant d'obtenir des résultats concrets. La modélisation permettrait de tester des idées et de les valider avant de les réaliser expérimentalement.

A

Modèle de neurone simulé : striatum

Afin de valider notre nouvelle approche présentée dans le chapitre 2, nous avons simulé le modèle de McCarthy et al. [72]. Le potentiel de membrane V est décrit par l'équation suivante :

$$\begin{aligned} C_m \frac{dV}{dt} = & - \underbrace{\bar{g}_K n^4 (V - E_K)}_{I_K} - \underbrace{\bar{g}_{Na} m^3 h (V - E_{Na})}_{I_{Na}} - \underbrace{\bar{g}_M p (V - E_K)}_{I_M} \\ & - \underbrace{\bar{g}_L (V - E_L)}_{I_L} - I_{syn} + I_{app} \end{aligned} \tag{A.1}$$

où C_m est la capacité de la membrane, $\bar{g}_X$ est la conductance maximale de l'ion X, les valeurs h, m, n et p représentent les «gating variables» des différents canaux ioniques (activation/inactivation) et la variable E_X est le potentiel d'inversion du canal ionique X. Le courant rapide du potassium I_K possède quatre «gating variables» d'activation et aucune d'inactivation. Le courant de sodium I_{Na} a trois «gating variables» d'activation et une d'inactivation. Le courant M est un courant de potassium possédant une «gating variable» d'activation et aucune d'inactivation. Le courant de fuite est désigné par I_L.

Le courant synaptique I_{syn} est un courant inhibiteur du GABAa donné par l'équation (2.1) avec $E_{jk} = -80mV$ et $\tau_{syn,jk} = 12ms$. Le courant appliqué I_{app} est, soit un courant constant, soit un courant en escalier avec bruit. Le courant en escalier est un courant négatif dans les premières secondes et ensuite positif.

Les valeurs d'initialisation sont les mêmes pour toutes les expériences en ajoutant un pourcentage d'hétérogénéité à ces valeurs. Le seul paramètre changeant entre les simulation est celui de la conductance du courant M, g_m, puisqu'il existe la conductance à l'état sain et celui à l'état parkinsonien.

Connectivité entre les neurones du MSN

D'après l'article de Lindahl et Hellgren Kotaleski [64], un neurone MSN est entouré d'environ 2800 autres neurones MSN et la densité de connexion dans son voisinage est de 18% [88], i.e. un neurone est connecté avec environ 504 neurones post-synaptique.

Paramètres de simulation		
Paramètres généraux	Type d'interpolation	Courbe de Bézier
	Méthode numérique	RK2
	Seuil du PA $[mV]$	0
	Pas de temps $[ms]$	0,005
	# Threads	1
Paramètres d'initialisation des neurones	V $[mV]$	-63,826
	m	0,027781
	h	0,990915
	n	0,962045
	p	0,022788
	Hétérogénéité [%]	10
Paramètres du neurone	$C_m\,[pF]$	1,0
	$g_L\,[mS]$	0,1
	$E_L\,[mV]$	-67,0
	$g_{Na}\,[mS]$	100,0
	$E_{Na}\,[mV]$	50,0
	$g_K\,[mS]$	80,0
	$E_K\,[mV]$	-100,0
	$g_m\,[mS]$ (normal)	1,34
	$g_m\,[mS]$ (Parkinson)	1,1

Tab. A.1 – Liste des paramètres de simulation du modèle MSN.

B

Résultats de la méthode MPI/openMP

B.1 Version MTMM : ∼ 136 000 neurones STN-GPeA-GPeP-MSN

Nous avons simulé le réseau des GB, i.e. STN-GPeA-GPeP-MSN, à l'échelle 1/10 du rat. Les paramètres de cette simulation sont les mêmes que dans l'annexe A, par contre les paramètres des modèles seront donnés aux chapitres 4 et 5.

Dans la Fig. B.1, la consommation de mémoire augmente lentement en fonction du nombre de threads, en restant très raisonnable puisque nous simulons ∼136 000 neurones. Entre la simulation de 100 000 neurones de MSN et celle des ∼ 136 000 neurones des GB, l'allocation de mémoire est 2 fois plus importante pour le réseau des GB, puisque les différents modèles de neurones sauvegardent plus de variables par neurone que le modèle de MSN.

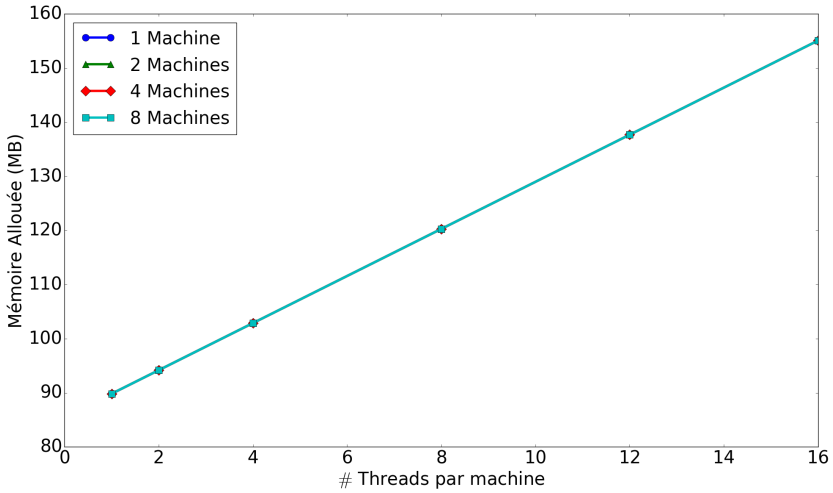

Fig. B.1 – Allocation de mémoire en fonction du nombre de threads avec la méthode MTMM pour la simulation du réseau STN-GPeA-GPeP-MSN avec ∼ 136 000, pendant 1s de temps biologique. Inspiré de l'article [13].

La Fig. B.2 nous montre que le temps de simulation diminue significativement en fonction

du nombre de processus employés. Cependant, plus nous utilisons de machines avec des threads, moins le temps de simulation diminue en fonction du nombre de threads. En particulier, avec 4 machines et 8-16 threads, le temps de simulation augmente lorsque nous ajoutons des threads. Une telle augmentation peut venir de la quantité de calcul par thread, qui devient trop petite pour permettre un gain de performance. En effet, plus nous augmentons le nombre de machines et/ou le nombre de threads, plus la quantité de neurones traitée par thread diminue. Or, il faut une quantité suffisante de travail par thread pour que leur utilisation apporte un gain de performance.

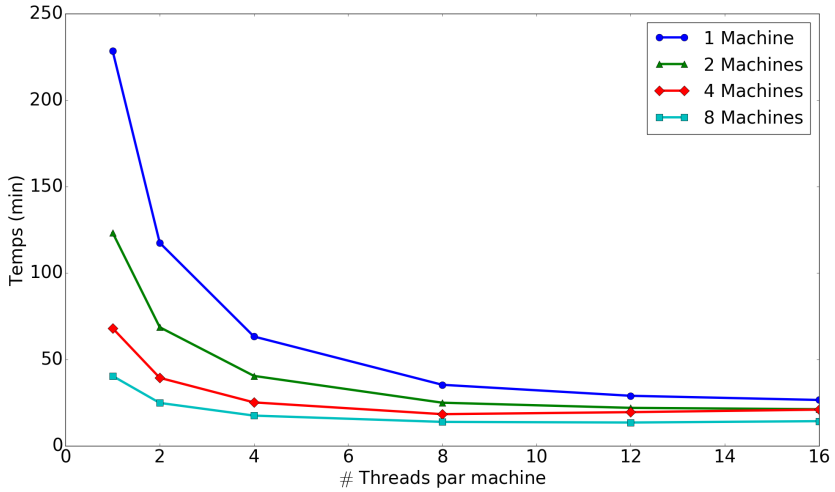

Fig. B.2 – Temps d'exécution en fonction du nombre de threads avec la méthode MTMM pour la simulation du réseau STN-GPeA-GPeP-MSN avec $\sim$ 136 000, pendant 1s de temps biologique.

La Fig. B.3 montre une accélération très modérée et moins bonne que dans la Fig. 3.20. Comme pour le résultat précédent, l'accélération avec 4 et 8 machines est mauvaise au-delà de 8 threads.

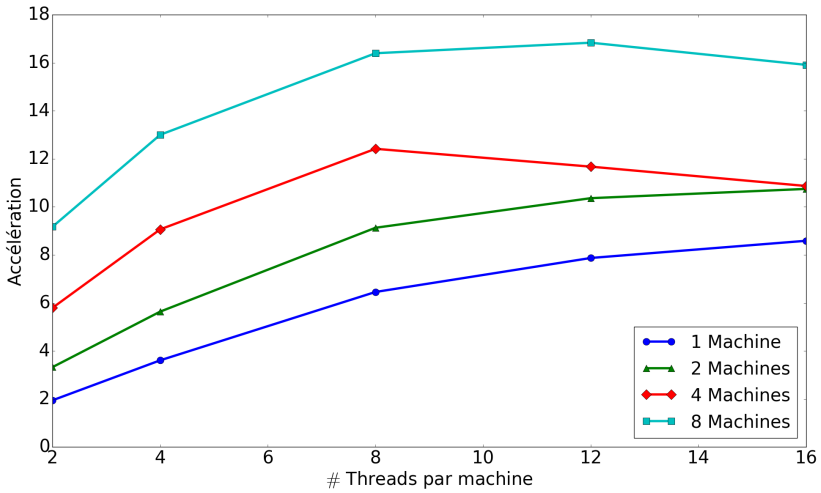

Fig. B.3 – Accélération de la simulation en fonction du nombre de threads avec la méthode MTMM du STN-GPeA-GPeP-MSN avec $\sim$ 136 000 neurones, pendant 1s de temps biologique.

L'efficacité de simulation avec le calcul parallèle chute très rapidement. Avec cette courbe d'efficacité, nous comprenons que la méthode est pénalisée par les communications entre les machines qui deviennent trop importantes par rapport au temps de calcul.

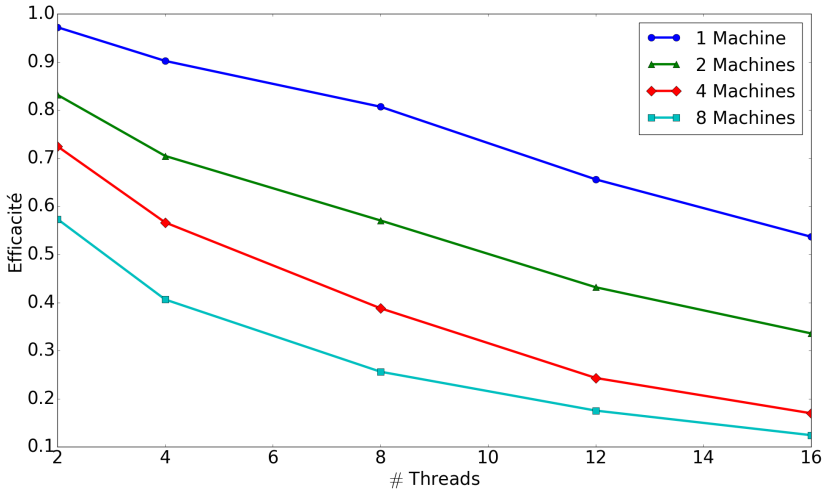

Fig. B.4 – Efficacité de la simulation en fonction du nombre de threads avec la méthode MTMM du STN-GPeA-GPeP-MSN avec ∼ 136 000 neurones, pendant 1s de temps biologique.

B.2 Version MTMM : ∼ 1 360 000 neurones STN-GPeA-GPeP-MSN

Nous avons voulu simuler le réseau complet à l'échelle une du rat, i.e. 1 360 000 neurones, avec la première version MTMM. Afin d'estimer le temps d'exécution, une première simulation de référence a été lancée sur une machine avec 16 threads. Cette simulation a duré 5 heures pour une seconde de temps biologique et a alloué 1,5 GB. Le temps pour les autres simulations avec au moins 16 processus a été estimé à partir de cette référence. Malheureusement, la simulation était plus lente que prévu. Au bout d'un moment, le cluster arrêtait prématurément notre simulation, car nous avions dépassé le temps alloué à la simulation dans le cluster. Lorsque nous avons constaté ce problème, nous avons arrêté le reste des simulations.

Nous avons quand même voulu étudier ces résultats, donc nous avons comparé le temps de simulation lorsque 20% de la simulation étaient exécutés, i.e. 20ms de temps biologique au lieu de 1s. Cependant nous n'avons aucun résultat utilisant plus de 8 machines.

La Fig. B.5 nous montre que les temps de simulation ne sont pas très bons et que plus nous utilisons de machines, moins le temps de simulation diminue en fonction du nombre de threads. En particulier, avec 4 machines et 8-16 threads, le temps de simulation augmente.

La Fig. B.6 montre une accélération très modérée avec 4 machines entre 12 et 16 threads. Nous pouvons conclure que cette première méthode MTMM n'est pas très adaptée à de gros réseaux (plus de 1 million de neurones) à cause des volumes de données échangées entre les machines. En effet, les communications entre machines deviennent prépondérantes.

L'efficacité de simulation avec le calcul parallèle chute rapidement. Ce résultat nous montre que la méthode reste à améliorer et que les communications entre les machines représentent un frein important à la vitesse de notre simulation. C'est pourquoi nous avons commencé l'étude

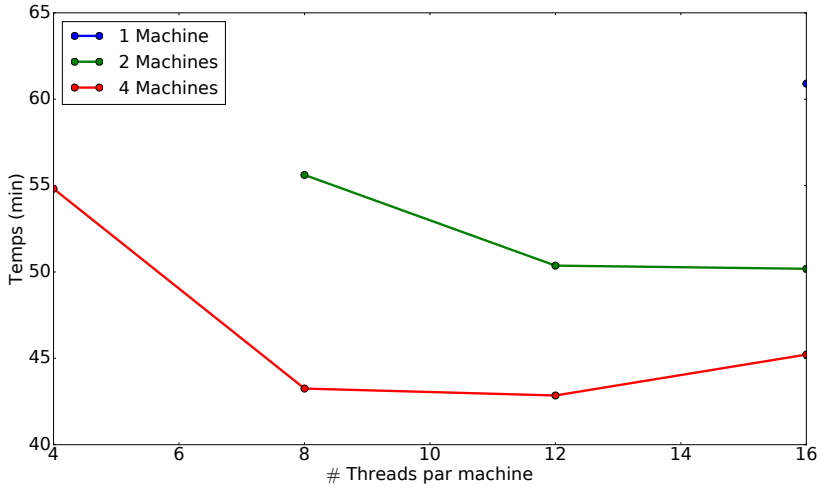

Fig. B.5 – Temps d'exécution en fonction du nombre de threads avec la méthode MTMM pour la simulation du réseau STN-GPeA-GPeP-MSN à l'échelle une du rat, soit $\sim$ 1,36 million de neurones, pendant $20ms$ de temps biologique.

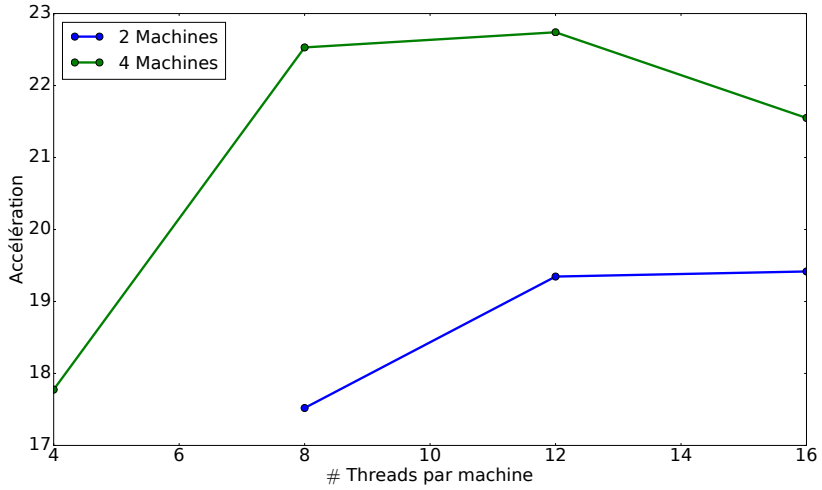

Fig. B.6 – Accélération de la simulation en fonction du nombre de threads avec la méthode MTMM du réseau STN-GPeA-GPeP-MSN à l'échelle une du rat, soit $\sim$ 1,36 million de neurones, en $20ms$ de temps biologique.

d'une version qui réduit les volumes de données échangées entre machines et qui fait appel à un recouvrement temporel d'une partie des communications par les calculs.

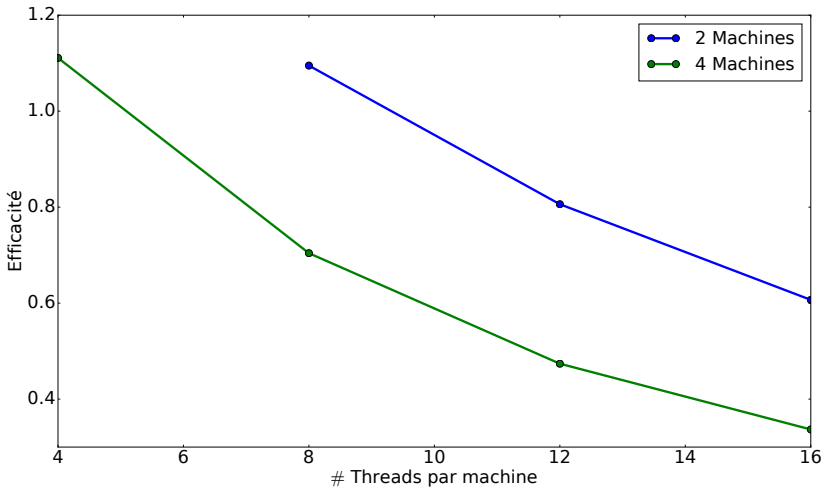

Fig. B.7 – Efficacité de la simulation en fonction du nombre de threads avec la méthode MTMM du réseau STN-GPeA-GPeP-MSN à l'échelle une du rat, soit ∼ 1,36 million de neurones, en $20ms$ de temps biologique.

C

Modèle du GPeA et GPeP

C.1 Paramètres d'initialisation des neurones

Le Tab. C.1 donne la liste des paramètres de simulation avec les paramètres d'initialisation du modèle GPe sans aucune distinction entre les deux populations de neurones.

Paramètres de simulation					
Paramètres d'initialisation des neurones	V $[mV]$	-74.66884	**Paramètres du neurone**	$C_m\,[pF]$	100,0
	m_{Na_P}	0,02070		$E_L\,[mV]$	-60,0
	h_{Na_P}	0,96212		$E_{Na}\,[mV]$	50,0
	s_{Na_P}	0,41		$E_K\,[mV]$	-80,0
	m_{Ca}	0,03364		$E_{cat}\,[mV]$	-30,0
	m_{HCN}	0,00621		$E_{Ca}\,[mV]$	130,0
	m_{Kv}	0,00101			
	h_{Kv}	0,9957			
	m_{NaF}	0,00029			
	h_{NaF}	0,99999			
	s_{NaF}	0,94518			
	q_{SK}	0,11778			
	m_{Ca}	0,00020			

Tab. C.1 – Liste des paramètres de simulation avec **SiReNe**.

C.2 Paramètres des courants ioniques

Les paramètres des courants des canaux ioniques sont tirés de l'article [39] et donnés dans le Tab C.2. Les emplacements avec une étoile sont des valeurs qui sont différentes pour le GPeA et GPeP. La valeur θ_m vaut -58,7 pour le GPeA et -52,0 pour le GPeP et la valeur k_m vaut 7,0 pour le GPeA et 5,7 pour le GPeP. Les valeurs en rouge sont des valeurs que nous avons ajustées

par rapport au modèle de Fujita et al. [39], afin que nous puissions adapter notre modèle aux données biologiques. Ces valeurs ont été trouvées avec un algorithme d'optimisation, expliqué dans la section 4.3.

Gate variables \ Courants ioniques		I_{NaP}	I_{NaF}	I_{K_v}	I_{HCN}	$I_{Ca_{vH}}$
m	θ_m	*	-39,0	-26,0	-76,4	-20,0
	k_m	*	5,0	7,8	-3,3	7,0
	τ_m^0	0,03	0,028	0,1	0,0	0,2
	τ_m^1	0,146	0,028	14,0	3625,0	0,2
	φ_m	-42,6	/	-26,0	-76,4	/
	σ_m^0	14,4	/	13,0	6,56	/
	σ_m^1	-14,4	/	-12,0	-7,48	/
h	θ_h	-57,0	-48,0	-20,0	/	/
	k_h	15,0	-2,8	-10,0	/	/
	h_{min}	0,154	/	0,6	/	/
	τ_h^0	1,5	0,25	7,0	/	/
	τ_h^1	2,5	4,0	33,0	/	/
	φ_h	-34,0	-43,0	0,0	/	/
	σ_h^0	26,0	10,0	10,0	/	/
	σ_h^1	-31,9	-5,0	-10,0	/	/
s	θ_s	-10,0	-40,0	/	/	/
	k_s	-4,9	-5,4	/	/	/
	s_{min}	/	0,15	/	/	/
	τ_s^0	/	10,0	/	/	/
	τ_s^1	/	1000,0	/	/	/
	φ_s	/	-40,0	/	/	/
	σ_s^0	/	18,3	/	/	/
	σ_s^1	/	-10,0	/	/	/

Tab. C.2 – Liste des paramètres de simulation du modèle GPe.

C.3 Courant NaP

D'après l'article de Fujita et al. [39], la constante de temps du «gate variable» s du courant NaP est définie différemment des autres «gate variables». Les valeurs des paramètres sont données dans le Tab. C.2.

$$\alpha_s(V) = \frac{A_s^\alpha V + B_s^\alpha}{1 - \exp\left[\frac{V + B_s^\alpha}{\frac{A_s^\alpha}{K_s^\alpha}}\right]} \quad \text{et} \quad \tau_s(V) = \frac{1}{\alpha_s(V) + \beta_s(V)} \tag{C.1}$$

$$\beta_s(V) = \frac{A_s^\beta V + B_s^\beta}{1 - \exp\left[\frac{V + B_s^\beta}{\frac{A_s^\beta}{K_s^\beta}}\right]}$$

Gate variable s du courant NaP	A_s^α	B_s^α	K_s^α	A_s^β	B_s^β	K_s^β
	-2,88e-6	-4,9e-5	4,63	6,94e-6	4,47e-4	-2,63

Tab. C.3 – Paramètres «gate variable» s du courant NaP.

C.4 Courant SK

Le courant SK est activé par le calcium qui est fourni par l'ouverture des canaux Ca_{vH}. Ainsi, la concentration de calcium intracellulaire est déterminée par l'équation suivante :

$$\frac{d\left[Ca^{2+}\right]_i}{dt} = -\frac{\gamma}{ZF} Ca_{vH} - K_{Ca}\left(\left[Ca^{2+}\right]_i - \left[Ca^{2+}\right]_{i0}\right) \tag{C.2}$$

où $Z = 2$ est la valence de l'ion Ca^{2+}, F est la constante de Faraday, $\left[Ca^{2+}\right]_{i0} = 0,01\mu M$ est la valeur de base de $\left[Ca^{2+}\right]_i$, $K_{Ca} = 0,4ms$ est le taux d'élimination du Ca^{2+}, et γ est le ratio entre la surface et le volume. Fujita et al. a fixé γ à $30000cm^{-1}$ afin que le transitoire du calcium atteigne $0,2 - 0,7\mu M$, lorsqu'un PA est généré.

Le courant SK est défini par l'équation de Hill :

$$m_\infty\left(\left[Ca^{2+}\right]_i\right) = \frac{\left[Ca^{2+}\right]_i^{H_{coeff}}}{(C_{50})^{H_{coeff}} + \left[Ca^{2+}\right]_i^{H_{coeff}}} \tag{C.3}$$

où le coefficient de Hill H_{coeff} vaut 4,6 et $C_{50} = 0.35\mu M$ est la concentration produisant un demi effet. La constante de temps τ_m est définie comme suit :

$$\tau_m = \begin{cases} \tau^1 - \dfrac{\left(\tau^1 - \tau^0\right)\left[Ca^{2+}\right]_i}{\left[Ca^{2+}\right]_{sat}} & \text{si} \quad \left[Ca^{2+}\right]_i < \left[Ca^{2+}\right]_{sat} = 5\mu M \\ \Delta t & \text{sinon} \end{cases} \tag{C.4}$$

où $\tau^0 = 4ms$ et $\tau^1 = 76ms$. Δt est le pas de temps fixé pour la simulation du modèle.

D

Modélisation et simulation du réseau STN-GPeA/P-MSN

D.1 Reproduction des résultats de l'article [10]

D.1.1 Paramètres d'initialisation des neurones

Le Tab. D.1 donne la liste des paramètres d'initialisation du modèle STN.

Paramètres d'initialisation	
V $[mV]$	-57,345760
h_{Na}	0,039472
n_K	0,506184
r_T	0,037866
Ca	0,042714

Tab. D.1 – Liste des paramètres d'initialisation du STN.

D.1.2 Paramètres des courants appliqués

Cette section expose tous les paramètres des courant appliqués aux populations MSN-D2 et STN. Aucun courant n'a été appliqué aux neurones du GPeA et GPeP.

Les paramètres des courants appliqués donnés par les Tab. D.2, resp. D.3 et D.4 sont relatifs aux Figs. 5.6, resp. 5.7 et 5.8.

D.1.3 Paramètres synaptiques

Cette section expose tous les paramètres synaptiques de chaque connexion synaptique.

Paramètres des courants appliqués		
STN	Type du courant appliqué	constant avec bruit
	$I_{app}\,[\mu A]$	7,5 +/- 10%
MSN-D2	Type du courant appliqué	à impulsion avec bruit
	moment où le courant change $[ms]$	4000
	durée de l'impulsion	2000
	$I_{app}\,[\mu A]$ à la 1^{ire} partie	0,01
	$I_{app}\,[\mu A]$ à la 2^{ime} partie	2,5
	Amplitude du bruit	50%

Tab. D.2 – Liste des paramètres des courants appliqués lors de l'excitation du MSN-D2.

Paramètres des courants appliqués		
STN	Type du courant appliqué	à impulsion avec bruit
	moment où le courant change $[ms]$	4000
	durée de l'impulsion	2000
	$I_{app}\,[\mu A]$ à la 1^{ire} partie	6,0
	$I_{app}\,[\mu A]$ à la 2^{ime} partie	0,0001
	Amplitude du bruit	50%
MSN-D2	Type du courant appliqué	constant avec bruit
	$I_{app}\,[\mu A]$	0,001 +/- 0,1%

Tab. D.3 – Liste des paramètres des courants appliqués lors de l'inhibition du STN.

Paramètres des courants appliqués		
STN	Type du courant appliqué	à impulsion avec bruit
	moment où le courant change $[ms]$	4000
	durée de l'impulsion	2000
	$I_{app}\,[\mu A]$ à la 1^{ire} partie	3,5
	$I_{app}\,[\mu A]$ à la 2^{ime} partie	11,0
	Amplitude du bruit	50%
MSN-D2	Type du courant appliqué	constant avec bruit
	$I_{app}\,[\mu A]$	0,001 +/- 0,1%

Tab. D.4 – Liste des paramètres des courants appliqués lors de l'excitation du STN.

D.2 Simulation du réseau des GB

D.2.1 Paramètres des courants appliqués

Le Tab. D.6 expose tous les paramètres des courants appliqués aux populations MSN-D2 et STN. Aucun courant n'a été appliqué aux neurones du GPeA et GPeP. Un courant en escalier est appliqué aux neurones du MSN-D2. La deuxième partie du courant, i.e. $1,267\mu A$ est appliqué à $100ms$, cependant, nous avons ajouté une hétérogénéité, i.e. pour chaque neurone la deuxième partie du courant ne commence pas à précisément $100ms$, mais est décalée. À cet effet, nous

Paramètres synaptiques		
STN-GPeP	# de connexions	76
	$\tau[ms]$	12
	$E_{syn}[mV]$	0
	$g_{syn}[nS]/\#connections$	10/# connections
STN-GPeA	# de connexions	25
	$\tau[ms]$	7
	$E_{syn}[mV]$	0
	$g_{syn}[nS]/\#connections$	0,5/# connections
GPeA-(MSN-D2)	# de connexions	1130
	$\tau[ms]$	10
	$E_{syn}[mV]$	-85
	$g_{syn}[nS]/\#connections$	0,3/# connections
GPeP-STN	# de connexions	11
	$\tau[ms]$	12
	$E_{syn}[mV]$	-85
	$g_{syn}[nS]/\#connections$	0,2/# connections
GPeP-GPeA	# de connexions	8
	$\tau[ms]$	5
	$E_{syn}[mV]$	-85
	$g_{syn}[nS]/\#connections$	2,5/# connections
(MSN-D2)-GPeP	# de connexions	13
	$\tau[ms]$	6
	$E_{syn}[mV]$	-85
	$g_{syn}[nS]/\#connections$	0,9/# connections
(MSN-D2)-(MSN-D2)	# de connexions	500
	$\tau[ms]$	12
	$E_{syn}[mV]$	-85
	$g_{syn}[nS]/\#connections$	0,03/# connections

Tab. D.5 – Liste des paramètres synaptiques du réseau complet.

avons multiplié les $100ms$ par une valeur entre 0 et 1.

Paramètres des courants appliqués		
STN	Type du courant appliqué	constant avec bruit
	$I_{app}\,[\mu A]$	6,5 +/- 20%
MSN-D2	Type du courant appliqué	escalier avec bruit
	moment où le courant change $[ms]$	100 $\times[0,1]$
	$I_{app}\,[\mu A]$ à la 1^{ire} partie	-10,0
	$I_{app}\,[\mu A]$ à la 2^{ime} partie	1,267
	Amplitude du bruit	70%

Tab. D.6 – Liste des paramètres des courants appliqués pour la simulation du réseau complet.

Bibliographie

[1] Mpi. https://www.mpi-forum.org/docs/.

[2] Openmp. https://www.openmp.org/.

[3] Site de l'association france parkinson. https://www.franceparkinson.fr/.

[4] Générateur de nombres pseudo-aléatoires, 2021. https://fr.wikipedia.org/w/index.php?title=G%C3%A9n%C3%A9rateur_de_nombres_pseudo-al%C3%A9atoires&oldid=182145573.

[5] Message passing interface, 2021. https://fr.wikipedia.org/w/index.php?title=Message_Passing_Interface&oldid=180967726.

[6] Openmp, 2021. https://en.wikipedia.org/w/index.php?title=OpenMP&oldid=1051890197.

[7] A. Abdi, N. P. Mallet, F. Y. Mohamed, A. Sharott, P. D. Dodson, K. C. Nakamura, S. Suri, S. V. Avery, J. T. Larvin, F. N. Garas, S. N. Garas, F. Vinciati, S. Morin, E. Bezard, J. Baufreton, and P. J. Magill. Prototypic and arkypallidal neurons in the dopamine-intact external globus pallidus. *J. Neurosci.*, 35(17) :6667–6688, 2015. doi : 10.1523/JNEUROSCI.4662-14.2015.

[8] R. L. Albin, A. B. Young, and J. B. Penney. The functional anatomy of basal ganglia disorders. *Trends in Neurosci.*, 12(10) :366–375, 1989. doi : 10.1016/0166-2236(89)90074-X.

[9] H. An, M. A. Ehsan, Z. Zhou, and Y. Yi. Electrical modeling and analysis of 3d synaptic array using vertical RRAM structure. In *18th International Symposium ISQED*, pages 1–6, 2017. doi : 10.1109/ISQED.2017.7918283.

[10] A. Aristieta, M. Barresi, S. Azizpour Lindi, G. Barrière, G. Courtand, B. de la Crompe, L. Guilhemsang, S. Gauthier, S. Fioramonti, J. Baufreton, and N. P. Mallet. A disynaptic circuit in the globus pallidus controls locomotion inhibition. *Curr. Biol.*, 31(4) :707–721.e7, 2021. doi : 10.1016/j.cub.2020.11.019.

[11] Assemblée Nationale, M. C. Villani, and F. Lassarade. Rapport de l'office parlementaire d'évaluation des choix scientifiques et technologiques établi au nom de cet office, sur l'utilisation des animaux en recherche et alternatives à l'expérimentation animale : état des lieux et perspectives (M. Cédric Villani et Mme Florence Lassarade), 2019. https://www.assemblee-nationale.fr/dyn/15/rapports/ots/l15b1794_rapport-information.

[12] A. Aussel, L. Buhry, L. Tyvaert, and R. Ranta. A detailed anatomical and mathematical model of the hippocampal formation for the generation of sharp-wave ripples and theta-nested gamma oscillations. *J. Comput. Neurosci.*, 45(3) :207–221, 2018. doi : 10.1007/s10827-018-0704-x.

[13] N. Azevedo Carvalho, S. Contassot-Vivier, L. Buhry, and D. Martinez. Simulation of large scale neural models with event-driven connectivity generation. *Front. Neuroinform.*, 2020. doi : 10.3389/fninf.2020.522000.

[14] M. F. Bear, B. W. Connors, and M. A. Paradiso. *Neurosciences : à la découverte du cerveau.* Pradel, 2002.

[15] J. Benkert, S. Hess, S. Roy, D. Beccano-Kelly, N. Wiederspohn, J. Duda, C. Simons, K. Patil, A. Gaifullina, N. Mannal, E. Dragicevic, D. Spaich, S. Müller, J. Nemeth, H. Hollmann, N. Deuter, Y. Mousba, C. Kubisch, C. Poetschke, J. Striessnig, O. Pongs, T. Schneider, R. Wade-Martins, S. Patel, R. Parlato, T. Frank, P. Kloppenburg, and B. Liss. Cav2.3 channels contribute to dopaminergic neuron loss in a model of parkinson's disease. *Nat. Commun.*, 10(1) :5094, 2019. doi : 10.1038/s41467-019-12834-x.

[16] M. D. Bevan, P. J. Magill, D. Terman, J. P. Bolam, and C. J. Wilson. Move to the rhythm : oscillations in the subthalamic nucleus–external globus pallidus network. *Trends Neurosci.*, 25(10) :525–531, 2002. doi : 10.1016/S0166-2236(02)02235-X.

[17] I. Blundell, R. Brette, T. A. Cleland, T. G. Close, D. Coca, A. P. Davison, S. Diaz-Pier, C. Fernandez Musoles, P. Gleeson, D. F. M. Goodman, M. Hines, M. W. Hopkins, P. Kumbhar, D. R. Lester, B. Marin, A. Morrison, E. Müller, T. Nowotny, A. Peyser, D. Plotnikov, P. Richmond, A. Rowley, B. Rumpe, M. Stimberg, A. B. Stokes, A. Tomkins, G. Trensch, M. Woodman, and J. M. Eppler. Code generation in computational neuroscience : A review of tools and techniques. *Front. Neuroinform.*, 2018. doi : 10.3389/fninf.2018.00068.

[18] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris, M. Zirpe, T. Natschläger, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. El Boustani, and A. Destexhe. Simulation of networks of spiking neurons : A review of tools and strategies. *J. Comput. Neurosci.*, 23(3) :349–398, 2007. doi : 10.1007/s10827-007-0038-6.

[19] L. Buhry. Estimation de paramètres de modèles de neurones biologiques sur une plateforme de SNN (spiking neural network) implantés "in silico". url : http://www.theses.fr/2010BOR14057, 2010.

[20] L. Buhry, S. SaÏghi, A. Giremus, E. Grivel, and S. Renaud. Automated tuning of analog neuromimetic integrated circuits. In *Conference BioCAS*, pages 13–16, 2009.

[21] J. Carr. Tremor in parkinson's disease. *Parkinsonism & Related Disorders*, 8(4) :223–234, 2002. doi : 10.1016/S1353-8020(01)00037-2.

[22] C. S. Chan, J. N. Guzman, E. Ilijic, J. N. Mercer, C. Rick, T. Tkatch, G. E. Meredith, and D. J. Surmeier. 'rejuvenation' protects neurons in mouse models of parkinson's disease. *Nature*, 447(7148) :1081–1086, 2007. doi : 10.1038/nature05865.

[23] G. Chatzikonstantis, H. Sidiropoulos, C. Strydis, M. Negrello, G. Smaragdos, C. I. De Zeeuw, and D. J. Soudris. Multinode implementation of an extended hodgkin–huxley simulator. *Neurocomputing*, 329 :370–383, 2019. doi : 10.1016/j.neucom.2018.10.062.

[24] K. R. Chaudhuri and A. H. V. Schapira. Non-motor symptoms of parkinson's disease : dopaminergic pathophysiology and treatment. *Lancet Neurol.*, 8(5) :464–474, 2009. doi : 10.1016/S1474-4422(09)70068-7.

[25] F. Chersi, M. Mirolli, G. Pezzulo, and G. Baldassarre. A spiking neuron model of the cortico-basal ganglia circuits for goal-directed and habitual action learning. *Neural Netw.*, 41 :212–224, 2013.

[26] V. L. Corbit, T. C. Whalen, K. T. Zitelli, S. Y. Crilly, J. E. Rubin, and A. H. Gittis. Pallidostriatal projections promote β oscillations in a dopamine-depleted biophysical network model. *J. Neurosci.*, 36(20) :5556–5571, 2016. doi : 10.1523/JNEUROSCI.0339-16.2016.

[27] B. de la Crompe, A. Aristieta, A. Leblois, S. Elsherbiny, T. Boraud, and N. P. Mallet. The globus pallidus orchestrates abnormal network dynamics in a model of parkinsonism. *Nat. Commun.*, 11(1) :1570, 2020. doi : 10.1038/s41467-020-15352-3.

[28] M. Crouzeix and A. L. Mignot. *Analyse numérique des équations différentielles*. Collection Mathématiques appliquées pour la maîtrise. Masson, 1989.

[29] C. A. Deister, C. S. Chan, D. J. Surmeier, and C. J. Wilson. Calcium-activated SK channels influence voltage-gated ion channels to determine the precision of firing in globus pallidus neurons. *J. Neurosci.*, 29(26) :8452–8461, 2009. doi : 10.1523/JNEUROSCI.0576-09.2009.

[30] J.-P. Demailly. *Analyse numérique et équations différentielles*. EDP Sciences, 2006. doi : 10.1051/978-2-7598-2004-7.

[31] L. Devroye and C. Yuan. Inversion with correction for the computer generation of discrete random variable. Technical report, McGill University, 1986.

[32] D. T. Dexter and P. Jenner. Parkinson disease : from pathology to molecular disease mechanisms. *Free Radic. Biol. Med.*, 62 :132–144, 2013. doi : 10.1016/j.freeradbiomed.2013.01.018.

[33] M. Diesmann and M.-O. Gewaltig. NEST : An environment for neural systems. *Forschung und Wisschenschaftliches Rechnen Beiträge zum Heinz-Billing-Preis*, 58, 2002.

[34] E. Dragicevic, J. Schiemann, and B. Liss. Dopamine midbrain neurons in health and parkinson's disease : Emerging roles of voltage-gated calcium channels and ATP-sensitive potassium channels. *Neuroscience*, 284 :798–814, 2015. doi : 10.1016/j.neuroscience.2014.10.037.

[35] M. Ebert, C. Hauptmann, and P. A. Tass. Coordinated reset stimulation in a large-scale model of the STN-GPe circuit. *Front. Comput. Neurosci.*, 8, 2014. doi : 10.3389/fncom.2014.00154.

[36] C. Eliasmith and O. Trujillo. The use and abuse of large-scale brain models. *Curr. Opin. Neurobiol.*, 25 :1–6, 2014. doi : 10.1016/j.conb.2013.09.009.

[37] K. Y. Fan, J. Baufreton, D. J. Surmeier, C. S. Chan, and M. D. Bevan. Proliferation of external globus pallidus-subthalamic nucleus synapses following degeneration of midbrain dopamine neurons. *J. Neurosci.*, 32(40) :13718–13728, 2012. doi : 10.1523/JNEUROSCI.5750-11.2012.

[38] Z. Fountas and M. Shanahan. The role of cortical oscillations in a spiking neural network model of the basal ganglia. *PLoS One*, 12(12) :e0189109, 2017. doi : 10.1371/journal.pone.0189109.

[39] T. Fujita, T. Fukai, and K. Kitano. Influences of membrane properties on phase response curve and synchronization stability in a model globus pallidus neuron. *J. Comput. Neurosci.*, 32(3) :539–553, 2012. doi : 10.1007/s10827-011-0368-2.

[40] C. R. Gerfen, T. M. Engber, L. C. Mahan, Z. Susel, T. N. Chase, F. J. Jr. Monsma, and D. R. Sibley. D1 and d2 dopamine receptor-regulated gene expression of striatonigral and striatopallidal neurons. *Science*, 1990. doi : 10.1126/science.2147780.

[41] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal Dynamics : From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014. doi : 10.1017/CBO9781107447615.

[42] A. Gillies, 2005. https://commons.wikimedia.org/wiki/File:Basal-ganglia-coronal-sections-large.png#/media/File:Basal-ganglia-coronal-sections-large.png.

[43] B. Girard, J. Lienard, C. E. Gutierrez, B. Delord, and K. Doya. A biologically constrained spiking neural network model of the primate basal ganglia with overlapping pathways exhibits action selection. *Eur. J. Neurosci.*, 53(7) :2254–2277, 2021. doi : 10.1111/ejn.14869.

[44] K. E. Glajch, D. A. Kelver, D. J. Hegeman, Q. Cui, H. S. Xenias, E. C. Augustine, V. M. Hernández, N. Verma, T. Y. Huang, M. Luo, N. J. Justice, and C. S. Chan. Npas1+ pallidal neurons target striatal projection neurons. *J. Neurosci.*, 36(20) :5472–5488, 2016. doi : 10.1523/JNEUROSCI.1720-15.2016.

[45] L. Goenner, O. Maith, I. Koulouri, J. Baladron, and F. H. Hamker. A spiking model of basal ganglia dynamics in stopping behavior supported by arkypallidal neurons. *Eur. J. Neurosci.*, 53(7) :2296–2321, 2021. doi : 10.1111/ejn.15082.

[46] D. F. M. Goodman. Code generation : A strategy for neural network simulators. *Neuroinform.*, 8(3) :183–196, 2010. doi : 10.1007/s12021-010-9082-x.

[47] D. F. M. Goodman and R. Brette. Brian : a simulator for spiking neural networks in python. *Front. Neuroinform.*, 2, 2008. doi : 10.3389/neuro.11.005.2008.

[48] D. Hansel, G. Mato, C. Meunier, and L. Neltner. On numerical simulations of integrate-and-fire neural networks. *Neural Comput.*, 10 :467–83, 1998. doi : 10.1162/089976698300017845.

[49] V. M. Hernández, D. J. Hegeman, Q. Cui, D. A. Kelver, M. P. Fiske, K. E. Glajch, J. E. Pitt, T. Y. Huang, N. J. Justice, and C. S. Chan. Parvalbumin+ neurons and npas1+ neurons are distinct neuron classes in the mouse external globus pallidus. *J. Neurosci.*, 35(34) :11830–11847, 2015. doi : 10.1523/JNEUROSCI.4672-14.2015.

[50] M. Hines. NEURON — a program for simulation of nerve equations. In Frank H. Eeckman, editor, *Neural Systems : Analysis and Modeling*, pages 127–136. Springer US, 1993. doi : 10.1007/978-1-4615-3560-7_11.

[51] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.*, 117(4) :500–544, 1952. doi : 10.1113/jphysiol.1952.sp004764.

[52] B. Hu, M. Xu, L. Zhu, J. Lin, Z. Wang, D. Wang, and D. Zhang. A bidirectional hopf bifurcation analysis of parkinson's oscillation in a simplified basal ganglia model. *J. Theor. Biol.*, 536 :110979, 2022. doi : 10.1016/j.jtbi.2021.110979.

[53] M. D. Humphries, R. D. Stewart, and K. N. Gurney. A physiologically plausible model of action selection and oscillatory activity in the basal ganglia. *J. Neurosci.*, 26(50) :12921–12942, 2006. doi : 10.1523/JNEUROSCI.3486-06.2006.

[54] E. M. Izhikevich. *Dynamical systems in neuroscience : the geometry of excitability and bursting*. Computational neuroscience. MIT Press, 2007.

[55] E. M. Izhikevich and G. M. Edelman. Large-scale model of mammalian thalamocortical systems. *Proc. Natl. Acad. Sci. U.S.A.*, 105(9) :3593–3598, 2008. doi : 10.1073/pnas.0712231105.

[56] E.M. Izhikevich. Simple model of spiking neurons. *IEEE Trans. Neural Netw.*, 14(6) :1569–1572, 2003. doi : 10.1109/TNN.2003.820440.

[57] J. Jankovic. Parkinson's disease : clinical features and diagnosis. *J. Neurol. Neurosurg. Psychiatry*, 79(4) :368–376, 2008. doi : 10.1136/jnnp.2007.131045.

[58] L. J. Kish, M. R. Palmer, and G. A. Gerhardt. Multiple single-unit recordings in the striatum of freely moving animals : effects of apomorphine and d-amphetamine in normal and unilateral 6-hydroxydopamine-lesioned rats. *Brain Research*, 833(1) :58–70, 1999. doi : 10.1016/S0006-8993(99)01496-1.

[59] J. C. Knight and T. Nowotny. Larger GPU-accelerated brain simulations with procedural connectivity. *Nat. Comput. Sci.*, 1(2) :136–142, 2021. doi : 10.1038/s43588-020-00022-7.

[60] A. V. Kravitz, B. S. Freeze, P. R. L. Parker, K. Kay, M. T. Thwin, K. Deisseroth, and A. C. Kreitzer. Regulation of parkinsonian motor behaviours by optogenetic control of basal ganglia circuitry. *Nature*, 466(7306) :622–626, 2010. doi : 10.1038/nature09159.

[61] J. W. Langston. The parkinson's complex : Parkinsonism is just the tip of the iceberg. *Ann. of Neurol.*, 59(4) :591–596, 2006. doi : 10.1002/ana.20834.

[62] L. E. Lapicque. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarization. *J. Physiol. Pathol. Gen.*, 9 :620–635, 1907.

[63] D. H. Lehmer. Mathematical methods in large-scale computing units. In *Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery, 1949*, pages 141–146. Harvard University Press, Cambridge, Mass., 1951.

[64] M. Lindahl and J. Hellgren Kotaleski. Untangling basal ganglia network dynamics and function : Role of dopamine depletion and inhibition investigated in a spiking network model. *eNeuro*, 3(6), 2017. doi : 10.1523/ENEURO.0156-16.2016.

[65] B. Liss, O. Haeckel, J. Wildmann, T. Miki, S. Seino, and J. Roeper. K-ATP channels promote the differential degeneration of dopaminergic midbrain neurons. *Nat. Neurosci.*, 8(12) :1742–1751, 2005. doi : 10.1038/nn1570.

[66] W. W. Lytton, A. Omurtag, S. A. Neymotin, and M. L. Hines. Just in time connectivity for large spiking networks. *Neural Comput.*, 20(11) :2745–2756, 2008. doi : 10.1162/neco.2008.10-07-622.

[67] O. Maith, F. V. Escudero, H. Ü. Dinkelbach, J. Baladron, A. Horn, F. Irmen, A. A. Kühn, and F. H. Hamker. A computational model-based analysis of basal ganglia pathway changes in parkinson's disease inferred from resting-state fMRI. *Eur. J. Neurosci.*, 53(7) :2278–2295, 2021. doi : 10.1111/ejn.14868.

[68] N. P. Mallet, L. Delgado, M. Chazalon, C. Miguelez, and J. Baufreton. Cellular and synaptic dysfunctions in parkinson's disease : Stepping out of the striatum. *Cells*, 8(9), 2019. doi : 10.3390/cells8091005.

[69] N. P. Mallet, B. R. Micklem, P. Henny, M. T. Brown, C. Williams, J. P. Bolam, K. C. Nakamura, and P. J. Magill. Dichotomous organization of the external globus pallidus. *Neuron*, 74(6) :1075–1086, 2012. doi : 10.1016/j.neuron.2012.04.027.

[70] N. P. Mallet, A. Pogosyan, L. F. Márton, J. P. Bolam, P. Brown, and P. J. Magill. Parkinsonian beta oscillations in the external globus pallidus and their relationship with subthalamic nucleus activity. *J. Neurosci.*, 28(52) :14245–14258, 2008. doi : 10.1523/JNEUROSCI.4199-08.2008.

[71] N. P. Mallet, R. Schmidt, D. Leventhal, F. Chen, N. Amer, T. Boraud, and J. D. Berke. Arkypallidal cells send a stop signal to striatum. *Neuron*, 89(2) :308–316, 2016. doi : 10.1016/j.neuron.2015.12.017.

[72] M. M. McCarthy, C. Moore-Kochlacs, X. Gu, E. S. Boyden, X. Han, and N. Kopell. Striatal origin of the pathologic beta oscillations in parkinson's disease. *Proc. Natl. Acad. Sci. U.S.A.*, 108(28) :11620–11625, 2011. doi : 10.1073/pnas.1107748108.

[73] J. W. Moore and F. Ramon. On numerical integration of the hodgkin and huxley equations for a membrane action potential. *J. Theor. Biol.*, 45(1) :249–273, 1974. doi : 10.1016/0022-5193(74)90054-X.

[74] M. Muthuraman, N. Koirala, D. Ciolac, B. Pintea, M. Glaser, S. Groppa, G. Tamás, and S. Groppa. Deep brain stimulation and l-DOPA therapy : Concepts of action and clinical applications in parkinson's disease. *Front. Neurol.*, 9 :711, 2018. doi : 10.3389/fneur.2018.00711.

[75] A. Nambu and Y. Tachibana. Mechanism of parkinsonian neuronal oscillations in the primate basal ganglia : some considerations based on our recent work. *Front. Syst. Neurosci.*, 8 :74, 2014. doi : 10.3389/fnsys.2014.00074.

[76] Eva M. Navarro-López, Utku Çelikok, and Neslihan S. Şengör. A dynamical model for the basal ganglia-thalamo-cortical oscillatory activity and its implications in parkinson's disease. *Cogn. Neurodyn.*, 15(4) :693–720, 2021. doi : 10.1007/s11571-020-09653-y.

[77] A. J. Nevado-Holgado, N. Mallet, P. J. Magill, and R. Bogacz. Effective connectivity of the subthalamic nucleus-globus pallidus network during parkinsonian oscillations. *J. Physiol.*, 592(7) :1429–1455, 2014. doi : 10.1113/jphysiol.2013.259721.

[78] T. Nowotny. Flexible neuronal network simulation framework using code generation for NVidia® CUDA™. *BMC Neuroscience*, 12(1) :P239, 2011. doi : 10.1186/1471-2202-12-S1-P239.

[79] D. E. Oorschot. Total number of neurons in the neostriatal, pallidal, subthalamic, and substantia nigral nuclei of the rat basal ganglia : A stereological study using the cavalieri and optical disector methods. *J. Comp. Neurol.*, 366(4) :580–599, 1996. doi : 10.1002/(SICI)1096-9861(19960318)366:4<580::AID-CNE3>3.0.CO;2-0.

[80] J. Parkinson. An essay on the shaking palsy. *JNP*, 14(2) :223–236, 2002-05-01. doi : 10.1176/jnp.14.2.223.

[81] L. Perestelo-Pérez, A. Rivero-Santana, J. Pérez-Ramos, P. Serrano-Pérez, J. Panetta, and P. Hilarion. Deep brain stimulation in parkinson's disease : meta-analysis of randomized controlled trials. *J. Neurol.*, 261(11) :2051–2060, 2014. doi : 10.1007/s00415-014-7254-6.

[82] D. Plenz and S. T. Kital. A basal ganglia pacemaker formed by the subthalamic nucleus and external globus pallidus. *Nature*, 400(6745) :677–682, 1999. doi : 10.1038/23281.

[83] A. Sanzeni, A. Celani, G. Tiana, and M. Vergassola. Theory of feedback controlled brain stimulations for parkinson's disease. *Physica A : Statistical Mechanics and its Applications*, 441 :121–130, 2016. doi : 10.1016/j.physa.2015.08.019.

[84] B. Schwab, T. Heida, Y. Zhao, E. Marani, S. van Gils, and R. Van Wezel. Synchrony in parkinson's disease : importance of intrinsic properties of the external globus pallidus. *Front. in Syst. Neurosci.*, 7 :60, 2013. doi : 10.3389/fnsys.2013.00060.

[85] R. B. Stein. Some models of neuronal variability. *Biophys. J.*, 7(1) :37–68, 1967. doi : 10.1016/S0006-3495(67)86574-3.

[86] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.*, 11(4) :341–359, 1997. doi : 10.1023/A:1008202821328.

[87] S. M. Suryanarayana, J. Hellgren Kotaleski, S. Grillner, and K. N. Gurney. Roles for globus pallidus externa revealed in a computational model of action selection in the basal ganglia. *Neural Netw.*, 109 :113–136, 2019. doi : 10.1016/j.neunet.2018.10.003.

[88] S. Taverna, E. Ilijic, and D. J. Surmeier. Recurrent collateral connections of striatal medium spiny neurons are disrupted in models of parkinson's disease. *J. Neurosci.*, 28(21) :5504–5512, 2008. doi : 10.1523/JNEUROSCI.5493-07.2008.

[89] J. M. Tepper, C. J. Wilson, and T. Koós. Feedforward and feedback inhibition in neostriatal GABAergic spiny neurons. *Brain Res. Rev.*, 58(2) :272–281, 2008. doi : 10.1016/j.brainresrev.2007.10.008.

[90] D. Terman, J. E. Rubin, A. C. Yew, and C. J. Wilson. Activity patterns in a model for the subthalamopallidal network of the basal ganglia. *J. Neurosci.*, 22(7) :2963–2976, 2002. doi : 10.1523/JNEUROSCI.22-07-02963.2002.

[91] D. Ting. Simple, optimal algorithms for random sampling without replacement. *CoRR*, 2021. doi : 10.48550/arXiv.2104.05091.

[92] J. S. Vitter. An efficient algorithm for sequential random sampling. *ACM Trans. Math. Softw.*, 13(1) :58–67, 1987. doi : https://doi.org/10.1145/23002.23003.

[93] B. L. Walter and J. L Vitek. Surgical treatment for parkinson's disease. *Lancet Neurol.*, 3(12) :719–728, 2004. doi : 10.1016/S1474-4422(04)00934-2.

[94] G. Young. Note on excitation theories. *Psychometrika*, 2(2) :103–106, 1937. doi : 10.1007/BF02288064.

Liste des publications

Journaux

Azevedo Carvalho N., Contassot-Vivier S., Buhry L. and Martinez D. Simulation of large scale neural models with event-driven connectivity generation. *Front. Neuroinform.*, 2020. doi :10.3389/fninf.2020.522000

Conférences (Poster)

Azevedo Carvalho N., Buhry L., Contassot-Vivier S., Baufreton J. and Martinez D. A computational model of GPe prototypic and arkypallidal neurons with automated paramter fitting. *NeuroFrance 2021*, 2021. url : https://hal.inria.fr/hal-03437767

Azevedo Carvalho N., Buhry L., Contassot-Vivier S., Baufreton J. and Martinez D. A physiologically realistic computational model of the basal ganglia network. *CNS 2021*, 2021. url : https://hal.inria.fr/hal-03437778

Résumé

Ma thèse s'articule autour de trois axes : développer un modèle biophysique, proposer un outil de simulation et exploiter le modèle en simulation.

Développer un modèle biophysique de la structure neuronale impliquée dans la maladie de Parkinson, les ganglions de la base. Nous simulons des neurones physiologiquement réalistes à l'aide du formalisme d'Hodgkin-Huxley permettant d'intégrer des canaux ioniques spécifiques. Dans les différentes populations de neurones des GB, dont les arkypallidaux et les prototypiques du GPe, ainsi que les neurones dopaminergiques D1 et D2 du striatum, les propriétés cellulaires et la connectivité semblent être des facteurs prépondérants dans le comportement oscillatoire du réseau. Notre modèle est validé par des données expérimentales en condition saine, recueillies par nos collaborateurs biologistes, chez le rat.

Proposer un outil de simulation de réseaux de neurones impulsionnels permettant des simulations réalistes à grande échelle, $>$ 1 million de neurones, sur machine parallèle i.e plateformes Grid'5000, Explor, etc... SiReNe est un simulateur de réseaux de neurones développé en langage C. Ce logiciel s'appuie sur une approche de simulation combinant une intégration numérique, Runge-Kutta 2, de la dynamique neuronale, et d'une génération événementielle de la connectivité du réseau lors des émissions des potentiels d'action. Cette approche hybride, développée pendant la thèse, permet de simuler des grands réseaux de neurones très détaillés du type Hodgkin-Huxley.

Dans le futur, notre modèle pourra être exploité en simulation pour tester certaines hypothèses sur la synchronisation pathologique observée dans la maladie de Parkinson. Les hypothèses que nous envisageons d'étudier portent, principalement, sur le rôle des connexions synaptiques GABAergiques, et des propriétés neuronales, intrinsèques, des canaux SK qui contrôlent la précision de la décharge neuronale. La simulation de modèles à grande échelle pourra, notamment, être utilisée pour limiter la synchronisation pathologique, et les troubles moteurs, par le biais de nouvelles méthodes de neurostimulation, comme la stimulation cérébrale profonde.

Mots-clés: Développement logiciel, Modèle biophysique, Maladie de Parkinson.

Abstract

My thesis is based on three axes : to develop a biophysical model, to propose a simulation tool, and exploit the model in simulation.

We develop a biophysical model of the neuronal structure involved in Parkinson's disease, the basal ganglia. We simulate physiologically realistic neurons using the Hodgkin-Huxley formalism to incorporate specific ion channels present in different populations of GB neurons, including arkypallidal and prototypic GPe, as well as dopaminergic D1 and D2 neurons in the striatum, whose cellular properties and connectivity appear to be prominent factors in the oscillatory behavior of the network. Our model is validated by experimental data in healthy conditions of the rat, collected by our biologist collaborators.

We propose a simulation tool for impulse neural networks allowing realistic simulations on a large scale, $>$ 1 million neurons, on a parallel machine i.e. Grid'5000, Explor, etc... SiReNe is a neural network simulator developed in the C language. This software is based on a hybrid simulation approach. It combines a numerical integration, Runge-Kutta 2, of the neuronal dynamics and an event-driven generation of the network connectivity during action potentials emissions. This approach, developed during the thesis, allows the simulation of large and very detailed neural networks of the Hodgkin-Huxley type.

In the future, our model could be used in simulation to test some hypotheses on the pathological synchronization observed in Parkinson's disease. Like the role of GABAergic synaptic connections and the intrinsic neuronal properties of SK channels which control the precision of neuronal discharge. The simulation of large-scale models could be used to limit pathological synchronization and motor disorders through new neurostimulation methods, such as deep brain stimulation.

Keywords: Software development, biophysical model, Parkinson's disease.