Rapport soumis aux rapporteurs, dans le but de sanctionner le dossier pour l'obtention du grade de
Docteur en Informatique
de
l'Université des Antilles et de la Guyane

Approche distribuée pour la simulation événementielle de réseaux de neurones impulsionnels.

Application au contrôle des saccades oculaires.

Anthony Mouraud, le 25 Mai 2009

Mots-clés : Réseaux de neurones impulsionnels. Simulations événementielles. Simulation événementielles distribuées (DDES). Mutithreading. Neurosciences computationnelles. Système saccadique.

" Du manque de mémoire naquît l'intelligence"

Table des matières

Notations

BSP	Le modèle *Bulk Synchronous Parallel* est un modèle de simulation parallèle
cMRF	central Mesancephalic Reticular Formation. La formation réticulée mésancéphalique centrale est une partie du tronc cérébral impliquée dans les saccades oculaires.
CS	Colliculus Supérieur. Structure corticale impliquée dans les saccades oculaires.
DAMNED	*Distributed And Multithreaded Neural Event Driven simulation framework*. Un simulateur événementiel multithreadé et distribué.
DDES	*Distributed Discrete Event Simulation*. Simulation distribuée à événements discrets.
DES	*Discrete Event Simulation*. Simulations par événements discrets.
EBN	Excitatory Burst Neuron. Neurones excitateurs impliqués dans les saccades oculaires.
EPSC	*Excitatory Post Synaptic Current*. Courant post-synaptique excitateur induit par un PA pré-synaptique sur la membrane post-synaptique.
ESN	*Echo State Network*. Modèle de réseau de neurones.
FIFO	First In First Out. Premier arrivé, premier sorti. Garantit la conservation de l'ordre chronologique des éléments entrants
LAN	Local Area Network. Un réseau local. C'est un petit résau de machines interconnectées et isolées de l'extérieur.
LFP	*Local Field Potential*. Un potentiel de champ local est enregistré à l'aide d'une micro-électrode extracellulaire. Elle enregistre l'ensemble de l'activité « synaptique » dans une petite zone de cortex ($< 400\ \mu$m).
LP	*Logical Process*. Processus logique.
LSM	*Liquid State Machine*. Modèle de réseau de neurones
LTP - LTD	*Long Term Potentiation* et *Long Term Depression*. Potentialisation et dépression à long terme sont des phénomènes synaptiques, de modulation des potentiels post-synaptiques. LTP et LTD sont observées en électrophysiologie et sont utilisées dans les réseaux de neurones impulsionnels.
MIMD	*Multiple Instruction flows and Multiple Data flows*. Architecture de machine parallèle où différentes instructions sont exécutées en parallèle sur des données différentes.
MISD	*Multiple Instruction flows and Single Data flow*.

	Architecture de machine parallèle où différentes instructions sont exécutées en parallèle sur un seul jeu de données.
MN	Motoneurones. Neurones commandant les contractions musculaires.
MPI	*Message Passing Interface*. Il s'agit d'une norme définissant des protocoles d'échange de messages entre machines.
OPN	OmniPause Neurons. Les neurones omnipause sont impliqués dans les saccades oculaires.
PA	Potentiel d'Action. Signal électrique émis par un neurone vers les neurones auxquels il est connecté.
PDES	*Parallel Discrete Event Simulation*. Simulations événementielles parallèles.
PPS	Un Potentiel Post Synaptique correspond à l'impact d'un PA sur le potentiel de membrane du neurone post-synaptique.
RNA	Réseau de Neurones Artificiel.
SIMD	*Single Instruction flow and Multiple Data flows.* architecture de machine parallèle où un seul jeu d'instructions est exécuté en parallèle sur différentes données.
SISD	*Single Instruction flow and Single Data flow*. Architecture de machine parallèle où un seul jeu d'instruction est exécuté sur un seul jeu de données.
SNN	*Spiking Neurons Networks*. Réseaux de neurones impulsionnels.
SOM	*Self Organizing Map*. Modèle de réseau de neurones proposé par (Kohonen 1982).
Spike	Un spike est un potentiel d'action (ou PA). Le signal émit par un neurone lorsque celui-ci est suffisamment stimulé.
SPMD	Single Program and Multiple data flows. Architecture de machine parallèle où un seul programme est exécuté sur différentes données.
SRM	Le neurone *Spike Response Model* (Gerstner & Van Hemmen 1992).
STDP	*Spike Timing Dependent Plasticity*. Règle d'apprentissage non supervisé, reposant sur la plasticité synaptique. La modification des poids dépend des temps d'arrivée des PA pré et post synaptiques.
Thread	Processus léger, sous-catégorie de processus s'exécutant sur un système informatique.
TN	*Tonic neurons*. Neurones dont l'activité est tonique.

Liste des figures

Liste des Algorithmes

Introduction

Dans le domaine des neurosciences computationnelles, l'utilisation de modèles de neurones impulsionnels s'est intensifiée depuis quelques années. En effet les apports computationnels de ces modèles ainsi que leur conception proche des neurones biologiques font des neurones impulsionnels, dans leur forme mathématique ou informatique, le nouvel outil de prédilection des neurosciences computationnelles.

Les quantités ressources matérielles nécessaires à la simulation de réseaux dont les tailles permettent des comparaisons à des activités de réseaux biologiques sont importantes. Effectuer des simulations séquentielles de tels réseaux n'est pas très réaliste.

L'objectif de cette thèse est de montrer qu'il est possible d'exploiter des supports matériels parallèles pour effectuer des simulations de grands réseaux de neurones, dans des temps raisonnables. La réalisation d'un simulateur distribué et multithreadé pour effectuer ce type de simulations a constitué l'essentiel de ce travail de thèse. L'utilisation d'une méthode événementielle pour la simulation du réseau, d'un support matériel parallèle, du multithreading associés à une méthode de gestion du temps entièrement décentralisée n'avait pas encore été étudiée.

Nous montrons dans ce travail que le simulateur proposé permet d'exploiter un support matériel parallèle pour augmenter la complexité (taille du réseau, connectivité, dynamique) des simulations effectuées. Nous montrons également que la gestion des ressources matérielles distribuées par une stratégie événementielle couplée à une gestion distribuée du temps virtuel permet au simulateur réalisé d'accélérer les simulations et d'en augmenter la taille, d'autant plus que la complexité des réseaux est grande.

Nous montrons enfin comment utiliser le simulateur afin de réaliser une étude d'un modèle de réseau de neurones d'inspiration biologique. Un modèle entièrement connexionniste de contrôle des saccades oculaires par le tronc cérébral est proposé. Son exécution à l'aide du simulateur réalisé permet de montrer la plausibilité des hypothèses émises et de confirmer la possibilité de réaliser, à l'aide du simulateur, des expériences basées sur des données biologiques réelles.

Le *premier chapitre* introduit les simulations de réseaux de neurones impulsionnels, les différents modèles de neurones et de synapses, ainsi que quelques-unes des méthodes d'apprentissage existantes. Le *deuxième chapitre* présente les simulations par stratégie événementielle de réseaux de neurones impulsionnels. Le *troisième chapitre* introduit les concepts de simulations parallèles distribuées et présente certaines méthodes existantes permettant d'assurer le déroulement correct des simulations. Le *quatrième chapitre* détaille la composition et le fonctionnement du simulateur DAMNED développé au cours de cette thèse. Le *cinquième chapitre* évalue les performances

du simulateur DAMNED dans le cadre d'exécutions de réseaux de neurones de différentes tailles, avec différentes dynamiques et différentes connectivités, sur supports matériels distribués. Dans le *sixième chapitre*, un modèle entièrement connexionniste de contrôle oculomoteur par le tronc cérébral pour la génération de saccades oculaires est proposé. Ce modèle est étudié à l'aide du simulateur DAMNED.

Nous conclurons cette thèse dans un dernier chapitre, avant de fournir, en annexes, des précisions sur les détails des études présentées.

Réseaux de neurones impulsionnels

1

Le fonctionnement du cerveau a inspiré l'informatique dès l'origine de cette science. Il est profondément inscrit dans la mémoire commune que la reproduction artificielle du fonctionnement du cerveau humain est fondamentale pour plusieurs raisons. Tout d'abord, au-delà de l'imitation, cela pourrait permettre la démultiplication des capacités de calcul ainsi que l'évolution des méthodes de travail ou l'étude plus approfondie de phénomènes humains tels que la pensée ou la conscience. De même, les capacités de mémoire, de planification ou de raisonnement du cerveau animal, et des mammifères en particulier, sont telles que la reproduction artificielle des mécanismes sous-jacents pourrait apporter beaucoup à l'Homme. Le cerveau est un organe impliqué dans toutes ces fonctions animales et humaines. Il est constitué d'un réseau complexe et extrêmement étendu de cellules de différents types. Parmi elles, les neurones représentent la majeure partie. Dans un cerveau humain moyen se trouvent environ 100 milliards de neurones interconnectés. Un cerveau humain moyen contient environ 10^{15} connexions, avec en moyenne 10^4 connexions par neurone. Ce gigantesque réseau de neurones est en perpétuelle activité et c'est sur lui que reposent les capacités du cerveau.

Après avoir introduit les sources biologiques et computationnelles de modèles de neurones impulsionnels (sections 1.1), les définitions et fonctionnements des modèles les plus utilisés seront détaillés (section 1.2 à 1.6). Les modèles de réponses synaptiques les plus courants (section 1.7) et la prise en compte des délais de transmission des potentiels d'action entre les neurones (section 1.8) seront également étudiés au cours de ce chapitre avant de s'intéresser aux méthodes d'apprentissage dans les réseaux de neurones impulsionnels (section 1.9 et 1.10). Enfin nous présenterons quelques modèles de réseaux parmi les plus courants (section 1.11).

1.1 Neurosciences Computationnelles

Les Neurosciences Computationnelles s'intéressent aux apports de l'étude des réseaux de neurones artificiels pour les neurosciences et réciproquement.

1.1.1 Sources biologiques

En neurosciences, l'étude du comportement neuronal est, d'une manière générale, basée sur l'observation des variations du taux de décharge des neurones dans le

cadre de stimulations contrôlées expérimentalement. Les comportements neuronaux peuvent être recueillis par plusieurs moyens :

- par des enregistrements unitaires où les Potentiels d'Action (PA) émis par un neurone sont directement récupérées par une électrode intracellulaire ;
- par des potentiels de champ locaux (LFP, « Local Field Potential »), où les variations de potentiel des neurones d'une zone restreinte du cerveau sont enregistrés grâce à une électrode extracellulaire ;
- par des électro-encéphalogrammes (EEG), où des électrodes posées sur le scalp enregistrent les variations de potentiels dans de larges zones du cerveau ;
- ou encore par l'Imagerie par Résonance Magnétique fonctionnelle (IRMf), où l'observation des variations de la consommation de sang dans le cerveau renseigne sur les variations d'activité électrique des zones corticales sous-jacentes. Celles-ci sont principalement dues aux échanges de Potentiels d'Action (PA) entre les neurones.

Les travaux de Hubel & Wiesel (1962) montrèrent notamment l'importance de ces variations du taux de décharge des neurones dans le traitement visuel. Les premières modélisations informatiques ont donc légitimement considéré le taux de décharge des neurones comme porteur de l'information échangée par les neurones.

1.1.2 Sources computationnelles

Dans le même temps, dans la lignée des travaux de Turing (Turing 1936), McCulloch & Pitts (1943) proposèrent un modèle de neurone tentant de prouver que les neurones étaient capable de traiter des fonctions logiques de base. À cette époque, on pense alors que les capacités fonctionnelles complexes observées dans le cerveau, telles que l'attention ou la conscience (par exemple), peuvent émerger d'un agencement adéquat d'un grand nombre de neurones effectuant chacun un traitement correspondant à une porte logique de base. Il s'agit de neurones binaires qui peuvent être « actifs » ou « inactifs ». L'état du neurone est calculé en faisant la somme des états de tous les neurones x_i qui sont connectés en entrée de ce neurone (dendrites). Chaque connexion est pondérée par un poids w_i (voir figure 1.1).

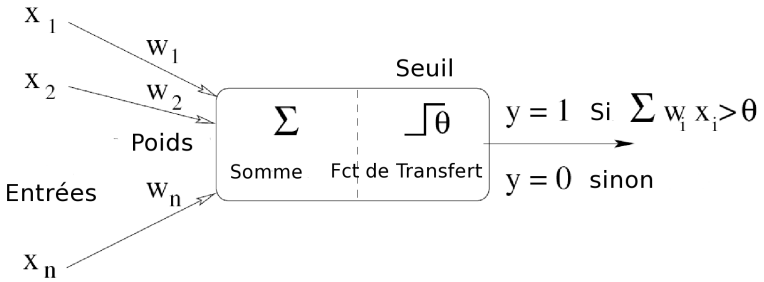

FIG. 1.1 – *Le modèle de neurone à seuil. Si la somme des entrées x_i, pondérées par w_i, dépasse le seuil Θ, alors la sortie y du neurone est différente de 0.*

Si la somme calculée dépasse le seuil θ fixé, alors le neurone passe à l'état « actif ». Par la suite, des modèles munis de fonctions de transfert linéaires par morceaux ou de forme sigmoïdale ont été développés. Pour ces modèles, les valeurs de sorties ne sont plus booléennes mais entières, ou réelles.

De nombreuses fonctions mathématiques, appliquables aux vecteurs d'entrées, peuvent être réalisées par de tels réseaux de neurones artificiels. Pour cela différentes méthodes d'apprentissage ont été mises au point (voir section 1.9).

D'un point de vue neuroscientifique, on peut associer les valeurs de sorties réelles à des fréquences de décharge instantanées. Cependant, la diversité et la complexité des traitements neuronaux ne peut être réduite à la seule notion de fréquence de décharge.

1.1.3 Importance de la temporalité dans le code neuronal

Plus récemment, d'autres travaux ont montré que certains traitements corticaux ne pouvaient s'effectuer uniquement sur la base d'un taux de décharge (Thorpe & Imbert 1989, Gray & Singer 1989) mais nécessitaient la prise en compte des instants d'émission des Potentiels d'Action (PA) individuels (Bohte 2004, pour une revue). Un PA est caractérisé par une variation brutale du potentiel de membrane du neurone. C'est une « impulsion » stéréotypée qui est ensuite propagée vers les neurones connectés. Cette propagation se fait par l'intermédiaire de canaux ioniques situés sur la membrane du neurone (en majorité potassium (K^+), sodium (Na^+) et chlore (Cl^-)). Ces canaux permettent la circulation d'ions entre le milieu intérieur du neurone et le milieu extérieur.

Un neurone est caractérisé par différentes zones ou compartiments illustrés sur la figure 1.2 :

1. Le soma, qui contient le noyau et donc le code génétique du neurone. Cette zone permet au neurone de synthétiser toutes les protéines nécessaires à son fonctionnement.
2. Les dendrites, qui supportent les stimulations arrivant sur le neurone. Les influx arrivant d'autres neurones faisant des connexions sur les dendrites génèrent une libération de neurotransmetteurs qui commandent les canaux ioniques responsables de la propagation d'un influx électrique vers le corps cellulaire du neurone.
3. L'axone est la voie d'expression du neurone. Les PA émis par ce neurone parcourent l'axone jusqu'aux terminaisons pré-synaptiques formant des synapses avec les dendrites d'autres neurones. De la myéline est présente sur certains axones afin d'accélérer la progression du PA le long de l'axone.
4. Le cône d'émergence, situé à la naissance de l'axone, est le siège de la génération d'un PA. L'intégration des entrées dendritiques modifie le potentiel de membrane du neurone. Lorsque le potentiel dépasse un seuil donné, il y a émission d'un PA. Le PA se propage dans l'axone et provoque la rétro-propagation d'une impulsion de faible amplitude dans les dendrites (Stuart & Sakmann 1994).

Lorsqu'un PA est émis par un neurone et que celui-ci arrive aux extrémités de l'axone, le passage de l'influx nerveux de la membrane pré-synaptique (du neurone émetteur) vers la membrane post-synaptique (du neurone récepteur) se fait via une *synapse* (figure 1.2). Un PA arrivant sur une synapse génère un Potentiel Post Synaptique (PPS). Cet impact peut être excitateur (PPSE) ou inhibiteur (PPSI) en fonction du type de neurotransmetteur de la synapse. Un neurotransmetteur excitateur génère un potentiel post synaptique dépolarisant tandis qu'un neurotransmetteur inhibiteur

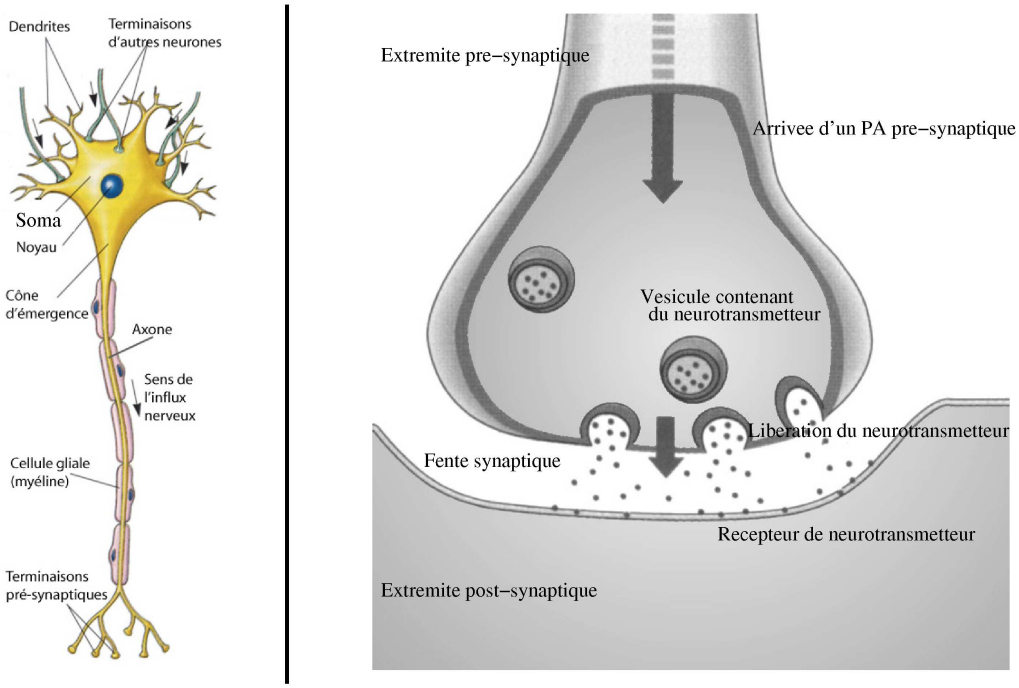

FIG. 1.2 – *Schéma d'un neurone représentant son aspect général. La forme choisie ici n'est qu'à titre indicatif. Les annotations désignent les zones communes à la majorité des neurones. Le noyau du neurone contient l'ADN et commande la fabrication des protéines assurant la survie du neurone. Les dendrites reçoivent l'influx venant d'autres neurones. Cet influx est intégré dans le soma. Les PA sont émis depuis le cône d'émergence et parcourent tout le neurone. La myéline permet d'accélérer la propagation dans l'axone. (Modifié de www.apteronote.com/revue/neurone/printer_77.shtml.) À droite, schéma d'une synapse. L'influx nerveux arrive par la voie pré-synaptique et provoque sur la membrane post-synaptique la génération d'un potentiel qui sera propagé jusqu'au soma du neurone post-synaptique. Le potentiel généré dépend de la quantité de neurotransmetteurs libérés dans la fente synaptique et du nombre de récepteurs recrutés pour la récupération des neurotransmetteurs. (modifié de http ://wwwppeda.free.fr/progressions/3/synapse-petit.gif).*

génère un potentiel post synaptique hyperpolarisant. L'impact d'un PA est différent d'une synapse à une autre et dépend de nombreux paramètres. Des récepteurs placés sur la membrane post-synaptique récupèrent les neurotransmetteurs libérés. Ces récepteurs induisent alors l'entrée d'ions Ca^{2+} et Na^{+} et la sortie d'ions K^{+} ce qui génère le PPS propagé jusqu'au soma (voir figure 1.3).

L'importance de la temporalité des émissions de PA unitaires dans le fonctionnement cortical et sous-cortical a motivé l'élaboration d'une nouvelle classe de modèles de neurones dits « impulsionnels » (ou neurones à spike, en anglais *spiking neurons*) modélisant les dynamiques sous le seuil et les émissions de PA unitaires.

L'efficacité et la maîtrise du comportement des neurones à seuil et sigmoïdaux, dans des tâches précises de classification ou d'approximation par exemple, leur assure encore aujourd'hui une utilisation intensive. Cependant, les réseaux de neurones impulsionnels (SNN pour *Spiking Neuron Networks*) remplacent progressivement les modèles classiques dans les simulations de réseaux neuronaux biologiquement plausibles car ils présentent plusieurs avantages, en particulier dans le domaine

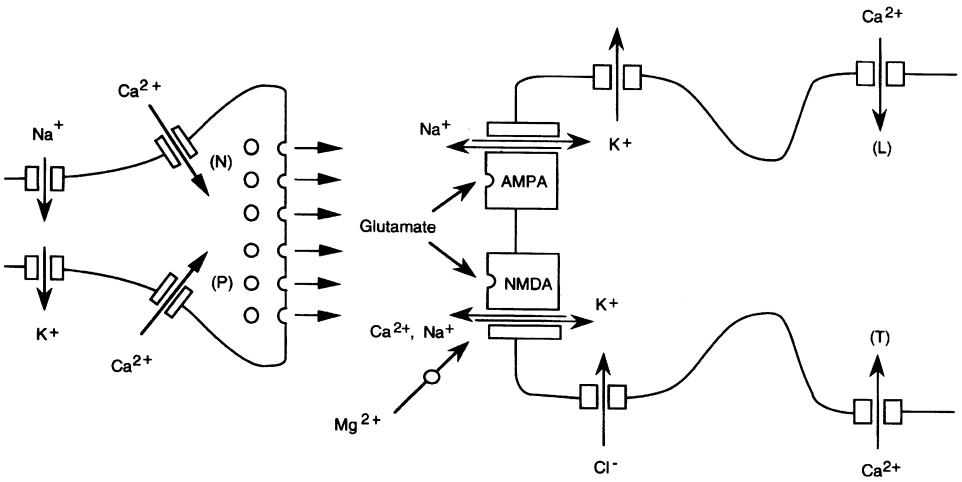

FIG. 1.3 – *La synapse chimique, ici il s'agit d'une synapse libérant du Glutamate (neurotransmetteur excitateur). Les récepteurs associés (AMPA et NMDA) ouvrent des canaux ioniques permettant l'échange d'ions K+, Na+ et Ca2+ entre l'extérieur et l'intérieur du neurone post-synaptique.*

des neurosciences computationnelles, mais également pour les ingénieurs (e.g. traitement de la parole, vision par ordinateur). D'une part, les neurones impulsionnels sont capables de reproduire avec exactitude le fonctionnement des neurones classiques (Maass & Natschläger 1998). D'autre part du fait de leur construction plus proche des neurones biologiques, ils permettent d'étudier aussi bien les fonctionnements unitaires que les interactions entre populations dans les systèmes perceptifs tels que la vision (Thorpe & Gautrais 1997), l'audition (Gerstner et al. 1999) ou l'olfaction (Hugues & Martinez 2005) ou même pour l'étude de concepts fonctionnels tels que le liage (ou *binding* (Von Der Malsburg 1981, Fries et al. 1997, Reynolds & Desimone 1999)), le « contrôle attentionnel » (Grossberg 2001, Siegel et al. 2000, Chevallier & Tarroux 2008), les oscillations (Eckhorn et al. 1989, Engel et al. 1991), la pensée ou la conscience (Grossberg 1999, O'Regan & Noe 2001, Crick & Koch 2003). Ainsi la compréhension et la maîtrise du fonctionnement des neurones impulsionnels représentent aujourd'hui un enjeu majeur de l'évolution des neurosciences computationnelles.

1.2 Le neurone de Hodgkin et Huxley

Bien avant l'arrivée des modèles impulsionnels, Hodgkin & Huxley (1952) avaient proposé un modèle de neurone, issu d'études sur les mécanismes responsables de la génération d'un PA au sein d'un neurone, qui leur ont valu le prix Nobel. Ce modèle (HH) est le modèle de neurone le plus complexe mais également le plus précis existant à ce jour. Il est composé de quatre équations différentielles couplées exprimant la dynamique du potentiel de membrane V_m du neurone. Celui-ci correspond à la différence de potentiel entre l'intérieur du neurone et le milieu extérieur.

V_m est fonction du courant d'entrée I appliqué au neurone lorsque celui-ci est stimulé, des courants I_K et I_{Na} générés par la circulation d'ions K^+ et Na^+ au tra-

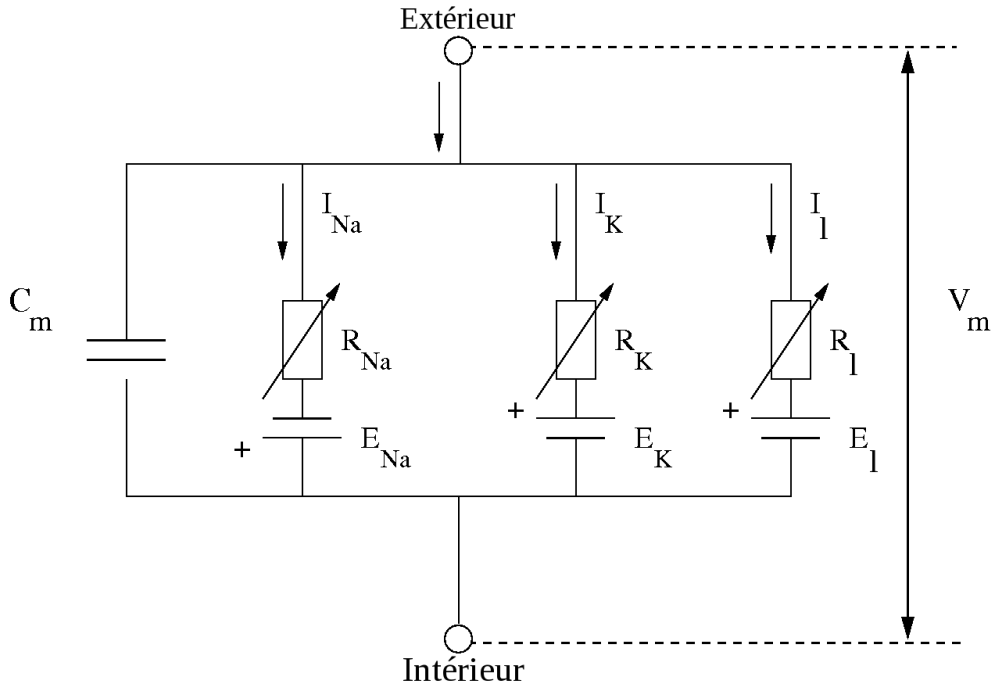

FIG. 1.4 – *Schéma électrique représentant les flux ioniques au travers de la membrane d'un neurone selon le modèle de Hodgkin et Huxley.*

vers de la membrane et d'un courant de fuite I_l représentant notamment les mouvements d'ions Cl^- (figure 1.3). Le schéma électrique correspondant est présenté sur la figure 1.4. Chacun de ces courants est fonction de l'écart entre le potentiel de membrane V_M et les potentiels d'inversion E_{Na}, E_K et E_l. Le potentiel de membrane du neurone est régi par l'équation :

$$C_M \frac{dV_M}{dt} = -g_{Na} m^3 h (V_M - E_{Na}) - g_K n^4 (V_M - E_K) - g_l (V_M - E_l) + I \tag{1.1}$$

où les $g_i, i \in \{K, Na, l\}$ sont des constantes et les paramètres h, m et n décrivent les probabilités d'ouverture/fermeture des canaux ioniques : sodium pour h et m, potassium pour n. Pour chacun de ces paramètres on a :

$$\frac{dx}{dt} = \alpha_x (1 - x) - \beta_x x \tag{1.2}$$

où $x \in \{h, m, n\}$ et où α_x et β_x correspondent respectivement aux probabilités d'entrée de particules à l'intérieur de la membrane et de sortie de particules vers le milieu extérieur. α_x et β_x ne dépendent pas directement du temps, mais dépendent du potentiel de membrane selon les équations :

$$\alpha_n = 0.01(V_M + 10)/(e^{((V_M+10)/10)} - 1) \tag{1.3}$$

$$\beta_n = 0.125 e^{V_M/80} \tag{1.4}$$

$$\alpha_m = 0.01(V_M + 25)/(e^{((V_M+25)/10)} - 1) \tag{1.5}$$

$$\beta_m = 4e^{V_M/18} \tag{1.6}$$

$$\alpha_h = 0.07e^{V_M/20} \tag{1.7}$$

$$\beta_h = 1/(e^{(V_M+30)/10} + 1) \tag{1.8}$$

La température a une influence sur ces paramètres. Les valeurs scalaires indiquées ici correspondent à une température de 6 degrés. De plus, ces équations décrivent le comportement d'un élément de surface du neurone. Dans les travaux de Hodgkin & Huxley, la description ne s'arrête pas au comportement d'un élément de surface. Ils tiennent aussi compte de la propagation des courants ioniques et de la température afin de décrire la transmission des PA d'un élément de surface au suivant. Mais la description du comportement d'un élément de surface est utilisée comme approximation dans la plupart des simulations, principalement pour des raisons de puissance de calcul nécessaire (par exemple à la simulation de la propagation d'un PA le long d'un axone).

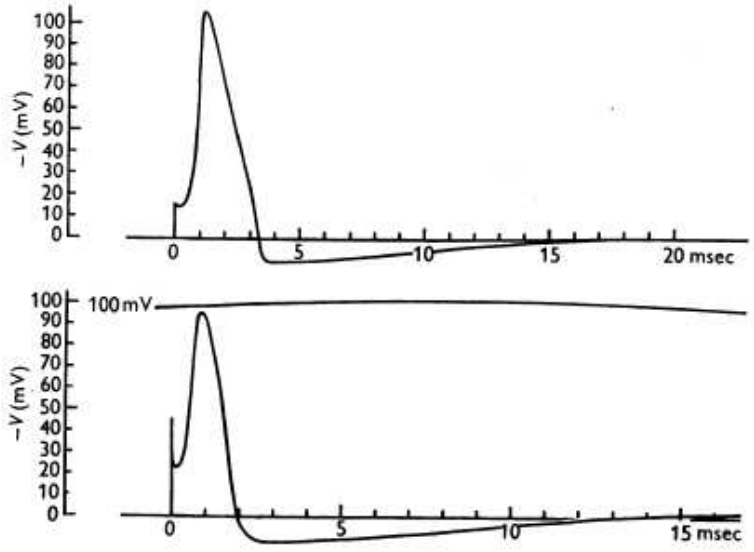

FIG. 1.5 – *Courbe du haut : Solution de l'équation* (1.1) *pour une stimulation à* $t = 0$ *de 15 mV (calculée à une température de 6 degrés). Courbe du bas : PA électrophysiologique (enregistré à une température de 9.1 degrés). D'après (Hodgkin & Huxley 1952).*

A partir de ce modèle, de nombreux comportements neuronaux peuvent être reproduits. La figure 1.5 montre la réponse du modèle à une excitation de 15 mV (figure du haut) et la confronte à la réponse enregistrée au niveau d'un axone géant (figure du bas). Le gros inconvénient de ce modèle est sa complexité. En effet, le couplage des quatre équations différentielles (1.1) et (1.2) avec $x \in \{h, m, n\}$ rend extrêmement lourde la construction de réseaux à partir de tels modèles de neurones.

En conséquence les simulations informatiques utilisant des modèles de type HH (Wang & Buzsaki 1996) sont, la plupart du temps, des simulations unitaires ou de couples de neurones. Il est nécessaire de simplifier ce modèle pour être en mesure de simuler des réseaux d'au moins quelques centaines de neurones et pouvoir ainsi étudier leur dynamique ou la plausibilité d'interactions présupposées entre différentes populations.

1.3 Le modèle Intègre et Tire

Lapicque (1907), très en avance sur les connaissances du fonctionnement neuronal, avait proposé un modèle ne tenant compte que du potentiel de membrane V_M pour représenter l'activité du neurone. Ces travaux furent repris plus récemment (Knight 1972, Abbott 1999) pour lui donner le nom de neurone IF « Intègre et Tire » en anglais : *Integrate and Fire*.

Par rapport au modèle HH décrivant précisément le décours temporel du PA, le neurone IF propose une représentation du PA par une impulsion instantanée (voir figure 1.6). Dans le modèle de Lapicque, la dynamique sous le seuil est donnée par l'équation différentielle linéaire du premier ordre :

$$C_M \frac{dV_M}{dt} = -g_M(V_M - V_r) + I \tag{1.9}$$

Le potentiel de membrane V_M du neurone est constant et égal à son potentiel de repos V_r lorsqu'il n'est pas stimulé. Les PPS reçus par le neurone (et/ou le courant appliqué au neurone) sont intégrés dans le courant I. Ils génèrent une variation de V_M qui revient ensuite à son potentiel de repos selon le coefficient de conductance g_M (voir figure 1.8). C_M est la capacité (de charge) du neurone. Une non-linéarité est introduite dans la dynamique illustrée sur la figure 1.6. Lorsqu'à $t = t_0$, $V_M(t_0)$ franchit une valeur seuil Θ, un PA est alors émis à t_0 et le potentiel de membrane V_M est réinitialisé à un potentiel d'hyperpolarisation V_h qui est parfois le même que le potentiel de repos V_r :

$$V_M(t_0) \geq \Theta, \text{alors} \left\{ \begin{array}{c} \text{émission d'un PA} \\ V_M = V_h \end{array} \right\} \tag{1.10}$$

Ce modèle est plus communément nommé Intègre et Tire à Fuite, (LIF pour *Leaky Integrate and Fire*) du fait que le potentiel de membrane est ramené progressivement à son potentiel de repos comme une fuite le ferait (pour vider un récipient par exemple). Un neurone IF au sens strict est alors un LIF sans fuite, c'est-à-dire qu'il intègre les entrées et émet un PA en dépassant le seuil. Mais il conservera un potentiel de membrane constant, quel que soit le temps de repos du neurone et son état d'excitation, jusqu'à la prochaine stimulation.

Le neurone ainsi défini correspond alors au schéma électrique de la figure 1.7.

Après émission d'un PA, le neurone sera maintenu à la valeur de son potentiel V_h pendant une période dite « réfractaire » τ_r pendant laquelle les stimulations n'auront pas d'impact sur le neurone. Il est également possible de modéliser cette période réfractaire par une augmentation arbitraire de la valeur seuil Θ (ou par une fonction seuil appropriée (e.g. section1.6)). Cette période réfractaire peut se diviser en deux phases (voir figure 1.6) :

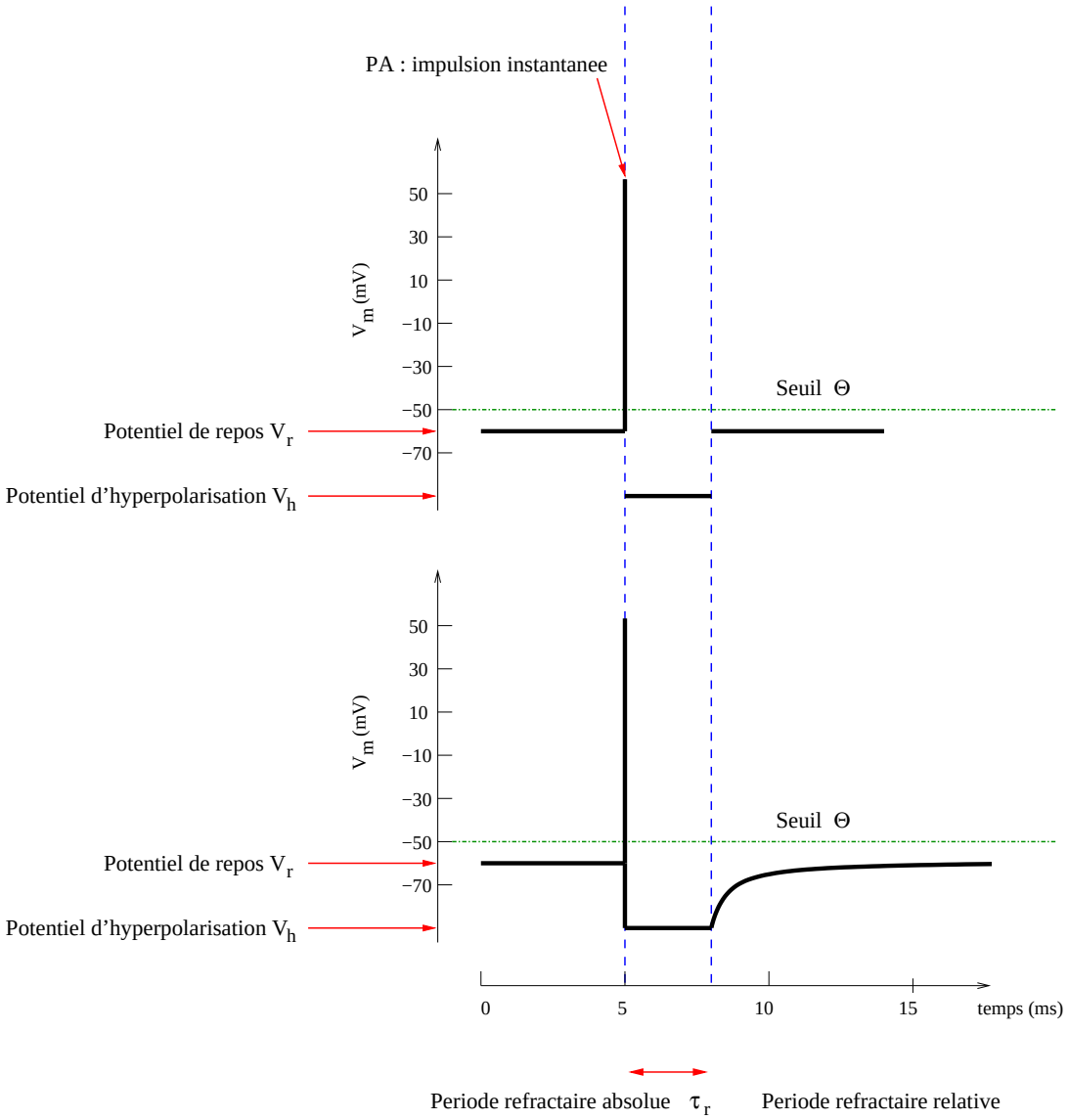

FIG. 1.6 – *Courbe du haut : forme du PA pour un neurone IF avec période réfractaire absolue. Courbe du bas : forme du PA pour un neurone IF avec période réfractaire absolue et relative.* $V_r = -60mV$, $V_h = -80mV$, $\tau_r = 3ms$, *le seuil est fixé à* $\Theta = -50mV$ *et le neurone est soumis à une stimulation d'une amplitude de 10 mV à la date* $t_0 = 5ms$ *et atteint le seuil entraînant l'émission d'un PA. La forme de la période réfractaire relative dépend de* g_M.

1. Une période réfractaire absolue qui maintient V_M à son potentiel d'hyperpolarisation V_h.
2. Une période réfractaire relative, durant laquelle le potentiel de membrane rejoint son potentiel de repos V_r selon l'équation (1.9).

Le modèle LIF réduit de 4 à 1 le nombre d'équations différentielles à résoudre ; il ne modélise plus les variations du potentiel de membrane d'un neurone à un instant donné mais réduit les émissions de PA à une impulsion instantanée et décrit modélise l'évolution du potentiel de membrane en l'absence de PA. Malgré cette simplification,

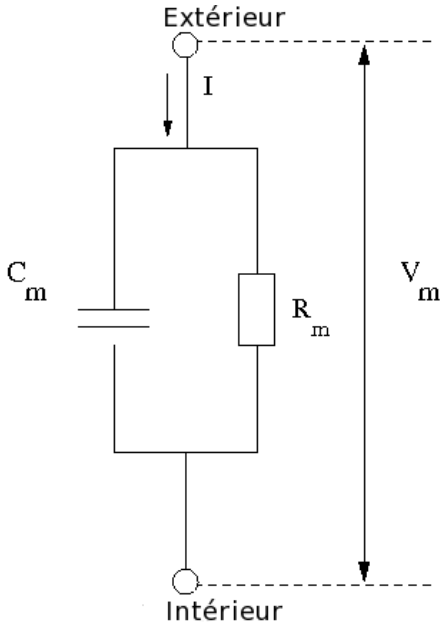

FIG. 1.7 – *Schéma électrique du neurone Intègre et Tire à fuite. Un condensateur de capacité C_M est branché en parallèle avec une résistance $R_M = \frac{1}{g_M}$.*

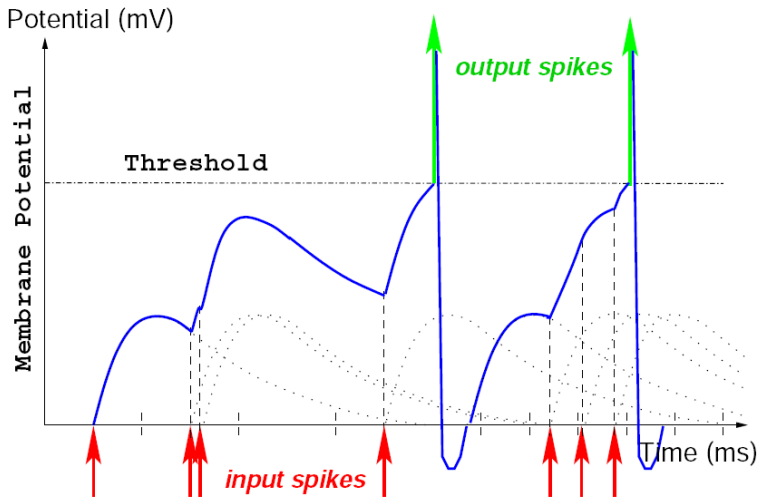

FIG. 1.8 – *Schéma d'intégration de PPS par un neurone de type LIF (D'après (Paugam-Moisy 2006)). Les flèches rouges indiquent la réception d'un PPS. Les flèches vertes représentent les PA émis par le neurone. Lorsque l'arrivée de PPS augmente le potentiel de membrane jusqu'à dépasser le seuil (threshold), alors un PA est émis.*

la puissance computationnelle (Maass 1996) et les capacités de stockage (Izhikevich 2006) d'un réseau utilisant de tels modèles de neurones sont bien supérieures à celles des modèles de neurone à seuil ou sigmoïdaux.

La simplification existant entre les modèles HH et LIF permet d'accéder à des informations sur les comportements neuronaux à des échelles allant du neurone unitaire au réseau de très grande taille. Les temps de simulation et les consommations

mémoire sont significativement réduits par l'utilisation de modèles plus simples que le HH. On trouve entre ces deux modèles différents niveaux de complexité. Chaque simplification entraîne de nouvelles approximations par rapport au comportement de neurones biologiques. Selon les questions abordées par les simulations, différents degrés de simplification existent à travers différents modèles. Les aptitudes de certains modèles à reproduire des comportements unitaires observés en neurobiologie en font des modèles particulièrement intéressants que nous allons présenter.

1.4 Modèles QIF et EIF

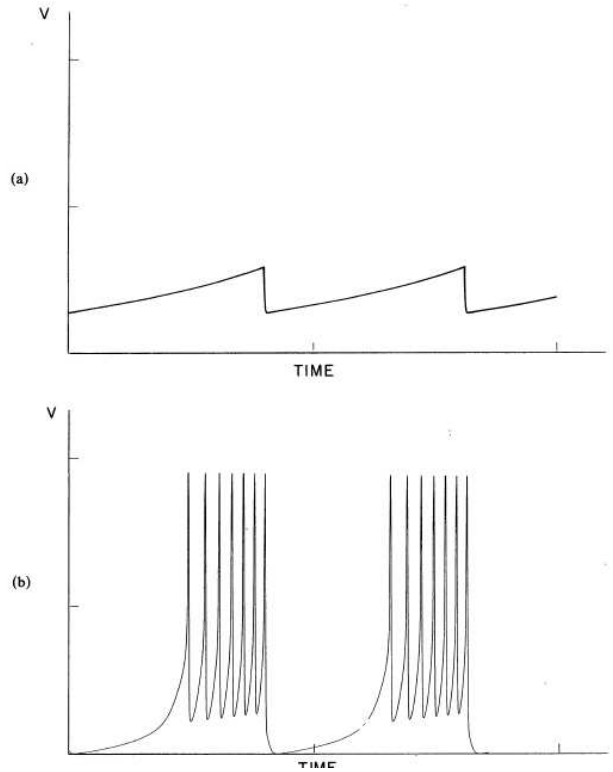

Fig. 1.9 – *Réponse d'un neurone HH simplifié à une vague de stimulation « triangulaire ». Le modèle ne tient compte que du paramètre n dans l'équation* (1.1)*. (D'après (Ermentrout & Kopell 1986))*

Le modèle HH connaît des simplifications diminuant de trois à deux le nombre de paramètres représentant les dynamiques ioniques (Morris & Lecar 1981, Brunel et al. 2003). Kopell & Ermentrout (1986) conservent un seul paramètre en plus du potentiel de membrane qu'ils proposent d'assimiler à l'activation des canaux K^+ (paramètre n dans l'équation (1.1)). Ceci peut se justifier par le fait que les dynamiques des trois paramètres du modèle HH ont des échelles de temps différentes : on peut se permettre de ne conserver que celui des trois qui a le plus d'influence pour approximer les équations (1.1) et (1.2).

Le modèle proposé par Kopell & Ermentrout (1986) reproduit correctement les dynamiques des neurones de Type 1 (Hodgkin & Huxley 1952, Ermentrout & Kopell 1986, pour une caractérisation des Types 1 et 2), ce qui permet de l'utiliser pour

étudier le comportement de ce type de neurones avec des modèles plus simples que le modèle HH (e.g.(Hugues & Martinez 2005)). Suivant le même principe, un modèle nommé « Quadratic Integrate and Fire » est proposé par Fourcaud-Trocmé et al. (2003). La non-linéarité induite par le paramètre ajouté au potentiel de membrane est introduite grâce à un terme de degré deux (en lieu et place de l'équation différentielle décrivant le paramètre n du modèle HH) :

$$C_M \frac{dV_M}{dt} = -g_l(V_M - E_l) + \Psi(V_M) + I(t) \tag{1.11}$$

$$\Psi(V_M) = \frac{g_l}{2\Delta_T}(V_M - V_T)^2 + g_l(V_M - E_l) - I_T \tag{1.12}$$

où $\Psi(V_M)$ est le terme quadratique ajouté à l'équation (1.9) du neurone LIF. V_T correspond au potentiel seuil de déclenchement d'un PA et I_T est le courant seuil correspondant. Δ_T représente la pente du potentiel d'action. Une variante de ce modèle nommée « Exponential Integrate and Fire » consiste à utiliser une exponentielle plutôt qu'un second degré dans la fonction $\Psi(V_M)$, l'équation (1.12) est remplacée par :

$$\Psi(V_M) = g_l \Delta_T e^{((V_M - V_T)/\Delta_T)} \tag{1.13}$$

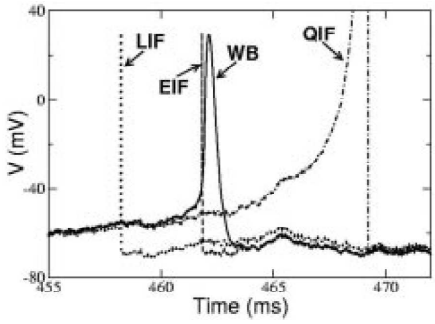

FIG. 1.10 – *D'après Fourcaud-Trocmé et al. (2003). Comparaisons des réponses de différents modèles de neurone impulsionnel pour l'application d'un même courant d'entrée. Les neurones Leaky Integrate and Fire (LIF), Quadratic Integrate and Fire (QIF), Exponential Integrate and Fire (EIF) montrent des réponses sensiblement différentes des modèles de type HH (ici WB proposé par Wang & Buzsaki (1996)) à de petites échelles de temps bien que leur comportement soit similaire à de plus grandes échelles de temps.*

La figure 1.10 montre les PA générés par les modèles de neurones LIF, EIF, QIF et un modèle équivalent du modèle HH proposé par Wang & Buzsaki (1996).

Le modèle QIF ainsi défini permet de réintroduire la forme du PA dans la dynamique du neurone. Cependant, la comparaison avec la dynamique du modèle HH montre que l'introduction du terme quadratique ne permet qu'une approximation de la forme du PA. En moyenne la forme du PA est stéréotypée et le PA a une durée de l'ordre de la milliseconde, ce qui justifie l'approximation faite à ce niveau par le modèle LIF qui n'a qu'un faible impact sur la dynamique, à des échelles de temps supérieures à la milliseconde. Bien que l'utilité des QIF ne soit plus à prouver, notamment en tant qu'oscillateurs (e.g Ermentrout (2005)), de nombreux comportements neuronaux ne peuvent être reproduits avec ce type de modélisations.

1.5 Modèle d'Izhikevich

Un autre type de modèle (Izhikevich 2003), basé sur deux équations différentielles couplées et inspiré du modèle de Fitzhugh-Nagumo (FitzHugh 1961, Nagumo et al. 1962), a ouvert l'accès à la reproduction de nombreux comportements observés sur les neurones biologiques. Ce modèle est décrit par les équations :

$$v' = 0.004v + 5v + 140 - u + I \tag{1.14}$$

$$u' = a(bv - u) \tag{1.15}$$

$$\text{Si } v \geq 30mV, \text{ alors } \left\{ \begin{array}{lcl} v & \leftarrow & c \\ u & \leftarrow & u + d \end{array} \right\} \tag{1.16}$$

où v et u sont des variables sans dimension et qui correspondent au potentiel de membrane et à la récupération de la membrane selon (Izhikevich 2003). En faisant varier les quatre paramètres a, b, c et d on peut obtenir la plupart des réponses neuronales connues (voir figure 1.11), ce qui rend ce modèle particulièrement intéressant. De plus, malgré le couplage de ses deux équations différentielles, ce modèle reste relativement simple en comparaison des modèles de type HH et il est particulièrement adapté aux simulations de réseaux de neurones impulsionnels (Izhikevich 2004).

Malgré ces simplifications, le QIF (et variantes reproduisant le comportement des neurones de Type 1) et le modèle d'Izhikevich conservent néanmoins plusieurs équations différentielles et la plupart des travaux utilisant ces modèles se concentrent sur de petits réseaux. Même si quelques exceptions existent (Izhikevich & Edelman 2008), c'est au prix d'énormes charges computationnelles.

Permettre des simulations, dans des temps raisonnables, de réseaux étendus, tout en conservant des dynamiques neuronales individuelles et une plausibilité biologique, nécessite donc d'autres approches pour la définition de modèles de neurones.

1.6 Le *Spike Response Model*

Deux différences majeures distinguent les modèles précédents du modèle SRM (pour *Spike Response Model*) décrit par Gerstner & Kistler (2002) qui trouve ses origines dans Gerstner & Van Hemmen (1992). Tout d'abord, les paramètres influençant le potentiel de membrane ne sont plus des fonctions de ce potentiel, ils dépendent de la date $\hat{t}_i$ du dernier PA émis par ce neurone. De plus, le potentiel de membrane n'est plus obtenu à l'aide d'équations différentielles ; il s'agit désormais d'une intégrale tenant compte du passé du neurone pour décrire l'évolution de son potentiel de membrane. Le potentiel de membrane u_i d'un neurone n_i est alors donné par l'équation :

$$u_i = \eta(t - \hat{t}_i) + \sum_j w_{ij} \sum_{t_j^{(f)}} \epsilon_{ij}(t - \hat{t}_i, t - t_j^{(f)}) + \int_0^\infty \kappa(t - \hat{t}_i, s) I^{ext}(t - s) ds \tag{1.17}$$

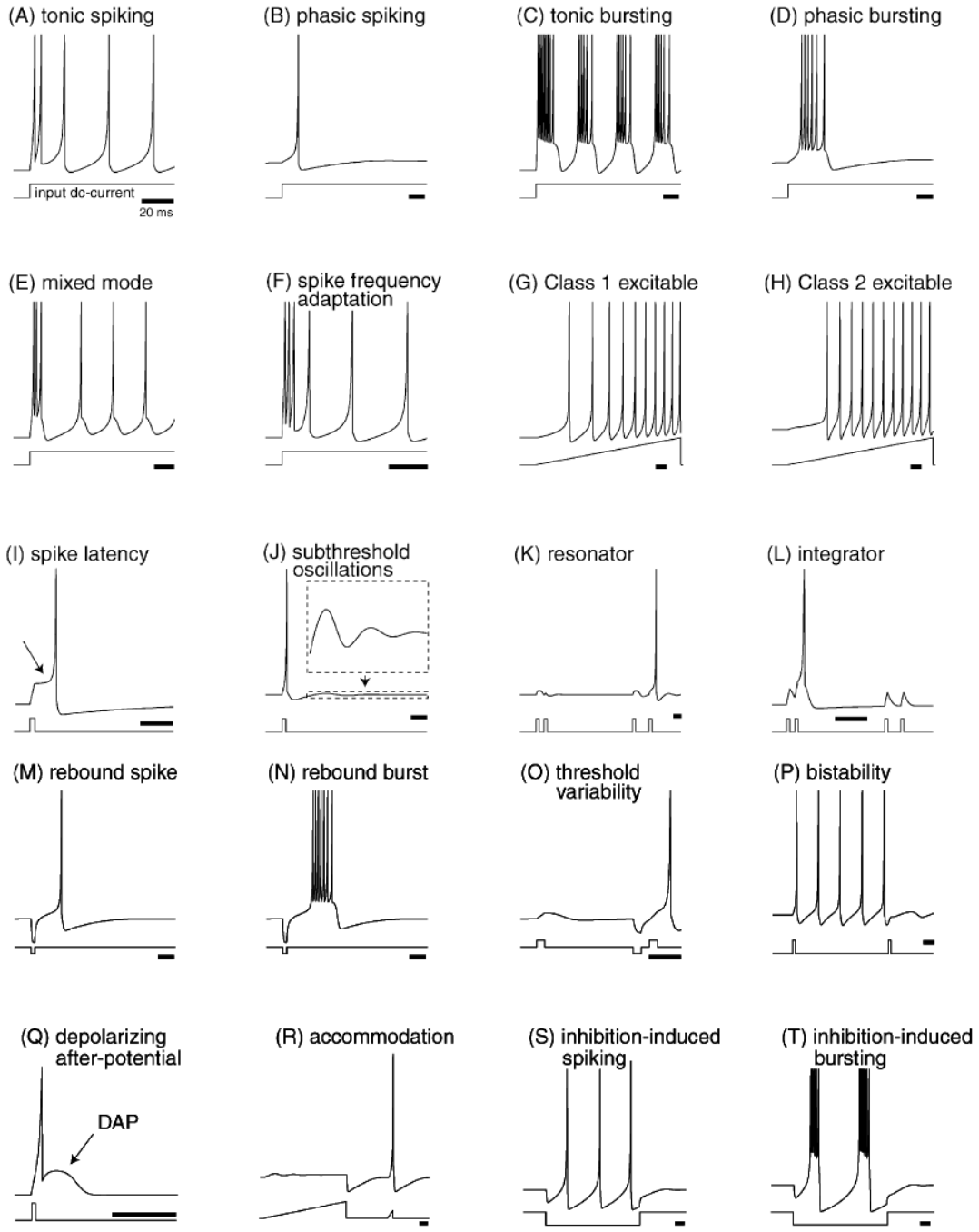

FIG. 1.11 – *Réponses neuronales obtenues pour différentes valeurs des quatre paramètres du modèle. Chacune est étiquetée par le comportement biologique correspondant.*

Un PA est émis par le neurone n_i à la date t lorsque :

$$u_i(t) = \vartheta(t - \hat{t}_i) \text{ et } \frac{du_i(t)}{dt} > 0 \tag{1.18}$$

Le terme $\eta(t - \hat{t}_i)$ décrit le comportement du neurone lors de l'émission d'un

PA et durant la période réfractaire qui suit. $\hat{t}_i$ est la date du dernier PA émis par le neurone n_i. ω_{ij} est le poids de la synapse entre un neurone pré-synaptique n_j et le neurone post-synaptique n_i. L'impact de l'arrivée d'un PA d'un neurone n_j sur le neurone n_i est décrit par le terme $\epsilon(t-\hat{t}_i, t-t_j^{(f)})$ et dépend de la date du dernier PA post-synaptique $\hat{t}_i$ et de celle d'un ou plusieurs PA pré-synaptiques $t_j^{(f)}$. Dans l'équation (1.17) la double somme est effectuée sur tous les neurones pré-synaptiques et sur tous leurs PA passés. Le dernier terme correspond à l'impact d'un éventuel courant extérieur appliqué au neurone. κ décrit la forme d'une impulsion appliquée au neurone. Le seuil ϑ au-dessus duquel un PA est émis peut également être variable et dépendre de $\hat{t}_i$; il s'exprime alors selon le terme $\vartheta(t-\hat{t}_i)$. La dépendance temporelle du seuil peut être évitée en fixant un seuil constant et en effectuant un changement de variable dans le terme η.

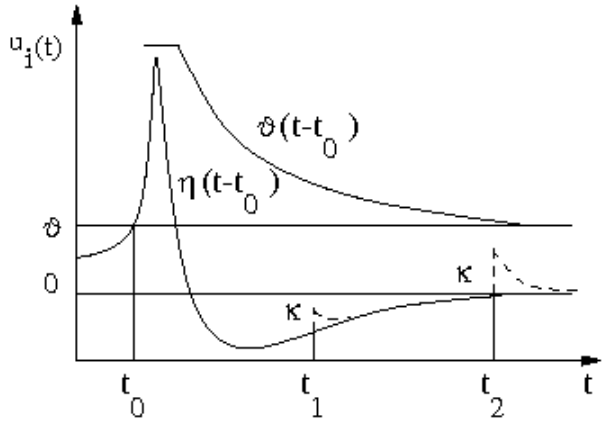

FIG. 1.12 – *Comportement du modèle SRM. Le neurone n_i émet un PA à $t = t_0 = \hat{t}_i$. $\eta(t-\hat{t}_i)$ décrit l'évolution du potentiel de membrane après un PA. κ décrit l'impact d'un PPS sur le potentiel de membrane. Cet impact dépend du temps écoulé depuis le dernier PA (t-$\hat{t}_i$). ϑ décrit l'évolution du seuil après l'émission d'un PA. D'après (Gerstner & Kistler 2002).*

La figure 1.12 fournit un exemple d'évolution du potentiel de membrane d'un neurone n_i que l'on peut obtenir grâce au modèle SRM.

Le modèle ainsi défini permet d'obtenir de nombreux comportements neuronaux. Le modèle LIF est alors un cas particulier du modèle SRM. Le modèle de Hodgkin et Huxley peut être reproduit avec une précision remarquable (Kistler et al. 1997) comme l'illustre la figure 1.13. Cependant, le modèle SRM reste complexe dans sa forme complète du fait qu'il est nécessaire de prendre en compte tous les spikes précédemment reçus par le neurone (équation (1.17)). Les auteurs proposent alors une variante beaucoup plus légère : le SRM_0. Dans ce modèle, la forme des impacts ne dépend plus du neurone pré-synaptique impliqué ($\epsilon_{ij} \rightarrow \epsilon$). Ce modèle ne tient plus compte de la date du dernier PA post-synaptique dans la forme des PPS (terme $\epsilon(t-\hat{t}, t-t_j^{(f)}) \rightarrow \epsilon(t-t_j^{(f)})$) et dans la forme des impulsions de courant extérieur (terme $\kappa(t-\hat{t}, t-t_j^{(f)}) \rightarrow \kappa(t-t_j^{(f)})$).

Le terme η représentant la forme du PA peut être exprimé comme un Dirac suivi d'une réinitialisation à un potentiel inférieur au potentiel de repos puis d'un retour vers le potentiel de repos avec une constante de temps τ_r.

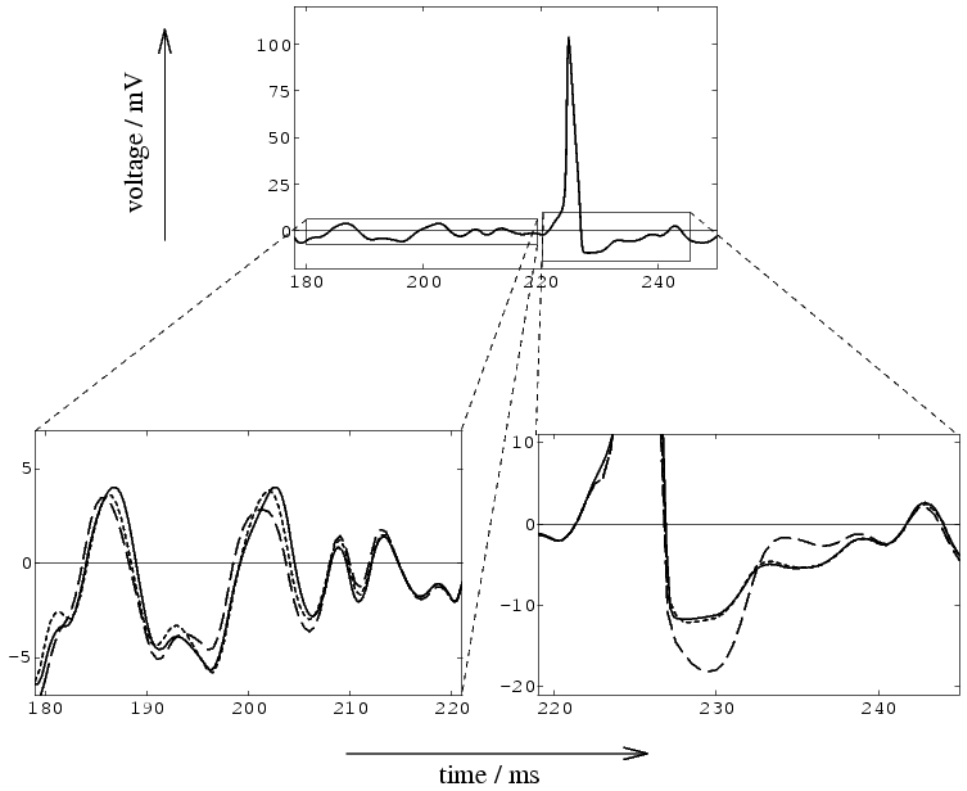

FIG. 1.13 – *D'après (Kistler et al. 1997), comparaison des dynamiques sous le seuil pendant les périodes pré-PA (encart de gauche) et post-PA (encart de droite), du modèle HH (ligne pleine) par rapport à son approximation à l'aide du modèle SRM (ligne pointillés) et à l'aide du modèle SRMo (ligne tirets).*

$$\eta(t-\hat{t}) = \delta(t-\hat{t}) - \eta_0 exp(-\frac{t-\hat{t}}{\tau_r}) \text{ avec } \eta_0 > 0 \tag{1.19}$$

Ceci exprime l'hyperpolarisation post PA. L'équation (1.17) devient alors :

$$u_i = \eta(t-\hat{t}_i) + \sum_j w_{ij} \sum_{t_j^{(f)}} \epsilon(t-t_j^{(f)}) + \int_0^\infty \kappa(s) I^{ext}(t-s) ds \tag{1.20}$$

Du fait de sa simplicité pour le calcul du potentiel de membrane (u_i), le modèle SRM_0 est un très bon candidat pour les simulations de grands réseaux. Il permet, comme le modèle SRM, d'approcher le modèle LIF (Gerstner et al. 1996). Le modèle SRM définit en fait le cadre d'un modèle de neurone. Ce sont surtout les fonctions choisies pour la forme du PA, du courant d'entrée et des réponses synaptiques qui détermineront la complexité réelle du modèle.

1.7 Réponses synaptiques

Dans de nombreuses simulations, la forme du PA est réduite à une impulsion et les courants d'entrée appliqués ont des formes simples. Pour tous les modèles de neurones cités, lorsque ceux-ci sont placés dans un réseau, les impacts des PA émis sur

les membranes post-synaptiques jouent un rôle important. Enrichir les modèles de réponses synaptiques est un autre bon moyen de se rapprocher du fonctionnement biologique. De plus, la mise en place, à une échelle locale, de règles de fonctionnement, même simples, permet l'émergence de comportements complexes à l'échelle d'un réseau.

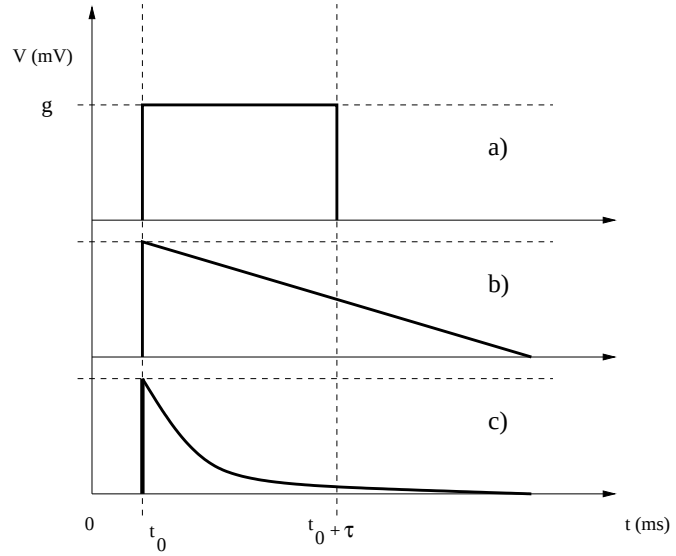

FIG. 1.14 – *Trois formes de réponse synaptique les plus simples. Les courbes correspondent à la réponse synaptique à l'arrivée d'un PPS, où g représente l'amplitude de la dépolarisation et τ la constante de temps.* **a)** *Impulsion en créneau.* **b)** *Impulsion instantanée suivie d'une fuite linéaire.* **c)** *Impulsion instantanée suivie d'une fuite exponentielle.*

D'un point de vue computationnel, dans un réseau de neurones en fonctionnement normal, les fonctions modélisant les réponses synaptiques (ϵ) sont plus utilisées que les équations modélisant la forme des PA et de la période réfractaire (η) (puisque chaque PA génère un grand nombre de PPS) ce qui implique que la complexité des fonctions mises en place pour modéliser les réponses synaptiques a plus d'impact sur la charge computationnelle requise pour la simulation.

La forme de réponse la plus simple (figure 1.14) est représentée par un créneau constitué d'une impulsion à la date d'impact t_0 du PPS, d'amplitude donnée g et maintenue pendant une durée τ. Ce modèle a permis à Maass (1994) de déterminer la puissance computationnelle d'un neurone impulsionnel simple (Maass & Markram 2004). On peut utiliser une impulsion instantanée à t_0, suivie d'une descente linéaire ou exponentielle. Il est aussi possible d'utiliser des fonctions linéaires par morceaux pour approximer les fonctions exponentielles présentées ici. (Mayrhofer et al. 2002)

Des réponses synaptiques, plus proches des réponses biologiques, basées sur une ou plusieurs exponentielles sont également couramment utilisées. À $t_0 = 0$, l'impact génère une montée exponentielle, d'amplitude g suivie d'une descente exponentielle avec une constante de temps τ. Si les constantes de temps de la montée et de la descente de la réponse sont les mêmes (voir figure 1.15, courbe du haut), alors une fonction α peut être utilisée :

$$\alpha(t) = \frac{t}{\tau} exp(1 - \frac{t}{\tau}) \tag{1.21}$$

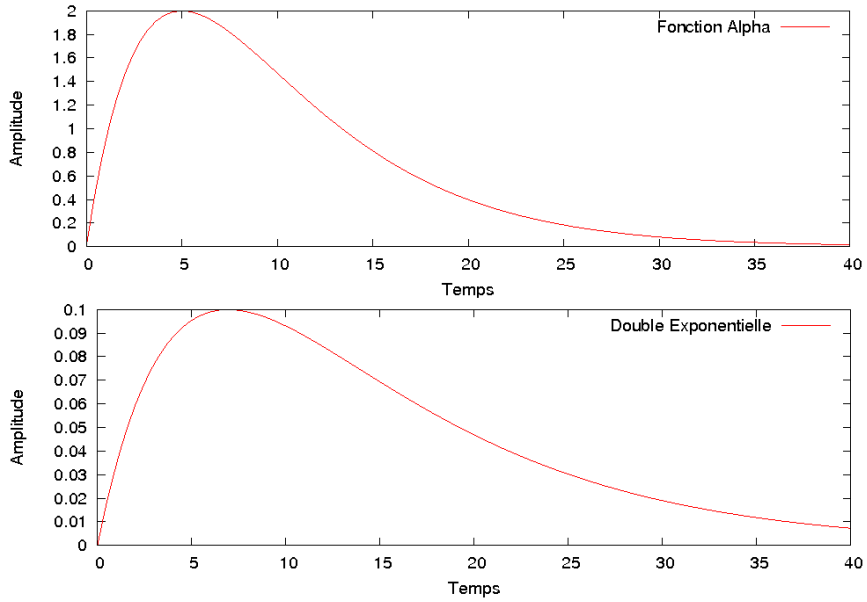

FIG. 1.15 – *Formes de réponses synaptiques en « fonction α » avec une constante de temps $\tau_1 = 5$ (courbe du haut) ou double exponentielle avec les constantes de temps $\tau_1 = 5$ et $\tau_2 = 10$ (courbe du bas).*

Dans le cas où les constantes de temps sont différentes, on utilise fréquemment une forme en double exponentielle selon :

$$\phi(t) = \frac{exp(-\frac{t}{\tau_1}) - exp(-\frac{t}{\tau_2})}{\tau_1 - \tau_2} \tag{1.22}$$

Dans les modèles de neurones présentés, les réponses synaptiques peuvent être introduites via une conductance synaptique ou un courant synaptique exprimés grâce aux équations (1.21) ou (1.22). Certains modèles dits « biophysiques » tenant compte des ouvertures/fermetures des canaux ioniques impliqués dans la réponse synaptique permettent de modéliser les dynamiques synaptiques avec précision (Destexhe et al. 1994). Il est également possible de faire varier les constantes de temps (τ) en fonction du temps ou du potentiel de membrane, de même que les amplitudes : variation de conductance (g) ou de courant (I) dans les réponses afin de modéliser les phénomènes d'adaptation observés dans les synapses biologiques (Brette 2007).

Les conductances synaptiques unitaires peuvent être approchées à l'aide d'une seule conductance, de deux conductances distinctes représentant les influences excitatrices et inhibitrices avec des dynamiques différentes, ou encore à l'aide d'une conductance par synapse ce qui alourdit considérablement le coût computationnel et l'espace mémoire nécessaire. En contrepartie, utiliser une conductance spécifique de chaque synapse donne la possibilité d'observer des dynamiques complexes au niveau de synapses individuelles et augmente ainsi la puissance computationnelle de tels réseaux.

Les modèles de neurones gIF (pour *conductance based IF*) de type 1-2 et 3, définis par Rudolph & Destexhe (2006) sont un exemple de modèle dont les dynamiques neuronales sont des extensions d'un modèle IF à conductances (Destexhe 1997, Brette

2006) et utilisent deux conductances distinctes, excitatrice et inhibitrice. Les trois principaux types de gIF correspondent à des modèles de neurone de plus en plus « adaptatifs » :

1.7.1 Modèle gIF1

Le gIF 1 est une extension du neurone LIF classique. L'équation (1.9) peut être exprimée comme :

$$\tau_M \frac{dV_M}{dt} + V_M = 0 \tag{1.23}$$

où τ_M est la constante membranaire. Cette constante définit la pente de la fuite du potentiel de membrane. On a la relation :

$$\tau_M = \frac{C_M}{g_M} \tag{1.24}$$

Le modèle gIF 1 tient compte de l'influence de l'activité synaptique sur la constante membranaire par l'intermédiaire de conductances exponentielles :

$$g_M(t) = g_M^L + g_M^S(t) \tag{1.25}$$

$$g_M^S(t) = g_M^e(t) + g_M^i(t) \tag{1.26}$$

où $g_M(t)^L$ est la conductance de fuite et $g_M(t)^S$ est la contribution synaptique à la conductance membranaire. L'arrivée d'un PPS excitateur ou inhibiteur apporte une contribution synaptique $\Delta g_M^{\{e,i\}}$ à la conductance membranaire g_M qui retourne ensuite exponentiellement à sa valeur de repos. $\{e,i\}$ distingue les contributions excitatrices et inhibitrices respectivement :

$$g_M^{\{e,i\}}(t) = g_M^{\{e,i\}}(t_0) e^{-\frac{t-t_0}{\tau_{\{e,i\}}}} \tag{1.27}$$

La constante membranaire τ_M du neurone dépendant de deux constantes membranaires excitatrice τ_M^e et inhibitrice τ_M^i et d'une constante membrane *de repos* τ_M^L selon :

$$\frac{1}{\tau_M(t)} = \frac{1}{\tau_M^L} + \frac{1}{\tau_M^e(t)} + \frac{1}{\tau_M^i(t)} \tag{1.28}$$

Les constantes membranaires excitatrices et inhibitrices évoluent en fonction des conductances associées (équation (1.24)).

Sans activité synaptique, les constantes de temps reviennent à leurs valeurs de repos. Plus l'activité synaptique (excitatrice ou inhibitrice) est forte, plus la fuite associée sera rapide (les constantes de temps diminuent) et plus précoce sera le retour aux valeurs de repos.

Le modèle gIF1 offre donc au potentiel de membrane des contributions excitatrices et inhibitrices distinctes. Les constantes de temps de ces contributions varient en fonction de l'activité excitatrice ou inhibitrice arrivant sur le neurone. Par extension, il est possible de sommer les contributions de toutes les dendrites, et donc de définir des constantes de temps spécifiques à chaque synapse.

1.7.2 Modèle gIF2

Le modèle gIF 2 est une extension du gIF de type 1. En plus de la variation de la constante membranaire en fonction de l'activité synaptique, le gIF 2 tient compte de l'influence de l'activité synaptique sur l'amplitude ΔV_M de l'impact d'un PPS :

$$\begin{aligned}\Delta V_M^{\{e,i\}}(\tau_M(t_0)) = \Delta V_M^{L\{e,i\}}(\tau_M^L) \times (\tfrac{1}{\tau_M^L} + \tfrac{1}{\tau_{\{e,i\}}} + \tfrac{1}{\Delta\tau_M^{\{e,i\}}}) \\ \times (\tfrac{1}{\tau_M(t_0)} + \tfrac{1}{\tau_{\{e,i\}}} + \tfrac{1}{\Delta\tau_M^{\{e,i\}}})^{-1}\end{aligned} \tag{1.29}$$

La variation d'amplitude a donc une partie $\Delta V_M^{L\{e,i\}}(\tau_M^L)$ correspondant à l'amplitude d'un PPS arrivant sur un neurone au repos. $\tau_M(t_0)$ est la valeur de la constante membranaire à t_0, date d'impact du PPS dont on calcule l'amplitude.

Sans activité synaptique, les contributions excitatrices et inhibitrices (dans le cas de l'arrivée d'un PPS à t = t_0) reviennent à leur valeurs de repos $\Delta V_M^{Le}(\tau_M^L)$ et $\Delta V_M^{Li}(\tau_M^L)$ respectivement. Plus l'activité synaptique est forte moins l'impact d'un PPS est important.

1.7.3 Modèle gIF3

Finalement, le modèle gIF 3 est un gIF de type 2 qui tient compte des potentiels d'inversion. Lorsque le potentiel de membrane s'éloigne de son potentiel d'inversion V_M^r à cause d'une stimulation (excitatrice ou inhibitrice), la fuite le ramène progressivement à ce potentiel. L'équation (1.9) devient :

$$\begin{aligned}&\tau_M(t)\tfrac{dV_M(t)}{dt} + (V_M(t) - V_M^r) = 0 \\ \text{avec}\quad &V_M^r = (\tfrac{V_M^L}{\tau_M^L} + \tfrac{V_M^e}{\tau_M^e(t_0)} + \tfrac{V_M^i}{\tau_M^i(t_0)}) \times (\tfrac{1}{\tau_M^L} + \tfrac{1}{\tau_M^e(t_0)} + \tfrac{1}{\tau_M^i(t_0)})^{-1}\end{aligned} \tag{1.30}$$

Lors de l'arrivée d'un PPS, le potentiel d'inversion V_M^r est mis à jour grâce à l'équation (1.30). L'amplitude de l'impact est alors donnée par :

$$\begin{aligned}\Delta V_M^{\{e,i\}}(\tau_M(t_0), V_M(t_0)) = \quad &\Delta V_M^{L\{e,i\}}(\tau_M^L, V_M^L)\tfrac{V_M(t_0) - V_{\{e,i\}}^r}{V_M^L - V_{\{e,i\}}^r} \\ &\times (\tfrac{1}{\tau_M^L} + \tfrac{1}{\tau_{\{e,i\}}} + \tfrac{1}{\Delta\tau_M^{\{e,i\}}}) \\ &\times (\tfrac{1}{\tau_M(t_0)} + \tfrac{1}{\tau_{\{e,i\}}}\tfrac{1}{\Delta\tau_M^{\{e,i\}}})^{-1}\end{aligned} \tag{1.31}$$

Le potentiel d'inversion est considéré comme constant entre deux réceptions de PA. Entre deux réceptions de PPS, le potentiel de membrane revient progressivement à son potentiel d'inversion selon l'équation :

$$\begin{aligned}V_M(t) = \quad &V_M^r + (V_M(t_0) - V_M^r) \times exp[-\tfrac{t-t_0}{\tau_M^L} \\ &-\tfrac{\tau_e}{\tau_M^e(t_0)} \times (1 - e^{-\frac{t-t_0}{\tau_e}}) \\ &-\tfrac{\tau_e}{\tau_M^e(t_0)} \times (1 - e^{-\frac{t-t_0}{\tau_e}})]\end{aligned} \tag{1.32}$$

L'utilisation des potentiels d'inversion excitateur V_e^r et inhibiteur V_i^r permet de moduler l'impact des PPS en fonction de l'écart au potentiel d'inversion. L'utilisation

de ce modèle est très utile notamment pour son impact sur les inhibitions qui peuvent devenir excitatrices si la membrane est « trop » hyperpolarisée.

Les modèles gIF permettent de se rapprocher des modèles biophysiques par rapport aux LIF. L'adaptation des constantes de temps, des amplitudes d'impact et l'utilisation de potentiels d'inversion permettent au modèle de réguler sa fréquence de décharge même en cas de stimulation forte. Ces modèles sont également adaptés à des activités synaptiques à forts taux de variations.

1.8 Délais de transmission dans les réseaux de neurones impulsionnels

La modulation des réponses synaptiques est un des outils des modélisateurs qui étudient les réseaux de neurones impulsionnels. Source d'une grande richesse de comportements, elle implique cependant un coût computationnel important, c'est pourquoi un nombre minimum de paramètres est souvent recherché afin d'exploiter la puissance de calcul pour agrandir la taille des réseaux étudiés plus que pour affiner le comportement des neurones.

La richesse des comportements neuronaux observés chez l'animal est également due à la temporalité des émissions de PA au sein des réseaux de neurones impliqués comme nous l'avons vu en section 1.1.3. La temporalité des émissions de PA dépend étroitement des délais de transmission des PA entre les neurones. Dans le cerveau, la transmission de l'influx nerveux d'un neurone vers un autre n'est pas instantanée. Le délai temporel entre l'émission d'un PA par un neurone et la réception de celui-ci par les différents neurones cibles est variable (Ferster & Lindström 1983, Swadlow 1988). Ce délai inclut la propagation du signal dans l'axone, le franchissement de la synapse et la propagation dans la dendrite jusqu'au soma du neurone post-synaptique. En moyenne, le délai de transmission d'un PA entre deux neurones est de l'ordre de la milliseconde même s'il peut atteindre plus de 40 millisecondes (Swadlow 1985).

Les modèles de neurones impulsionnels, même les plus simples (LIF, SRM_0), ont des dynamiques non linéaires (au moins dues à la fonction seuil qui modifie brutalement le potentiel de membrane, voir figure 1.8). L'intégration temporelle réalisée par les neurones biologiques, ainsi que par les modèles de neurones impulsionnels, implique que l'ordre chronologique des impacts des PPS sur la membrane post-synaptique influence l'instant d'émission d'un PA.

On modélise les délais de transmission de la même manière que les poids synaptiques : à l'aide d'un vecteur contenant un délai par connexion. La modélisation des délais de transmission permet d'étudier certaines aptitudes des réseaux biologiques (Gerstner et al. 1999, Mysore & Quartz 2005). Elle permet également d'étendre certaines des propriétés des assemblées de neurones synchrones (tels que les *synfire chains* (Abeles 1982; 1991)par exemple) à des assemblées de neurones dont les émissions de PA sont organisées temporellement selon un schéma d'activation précis et reproductible. Izhikevich (2006) nomme ces assemblées : « groupes polychrones ». L'identification des groupes polychrones activés durant une simulation pourrait permettre d'associer des classes de vecteurs d'entrée à des assemblées de neurones polychrones stables et différentes (Izhikevich et al. 2004, Paugam-Moisy et al. 2008). Un groupe est défini par un ensemble de couples (n_i,t) où n_i est un neurone et t est la

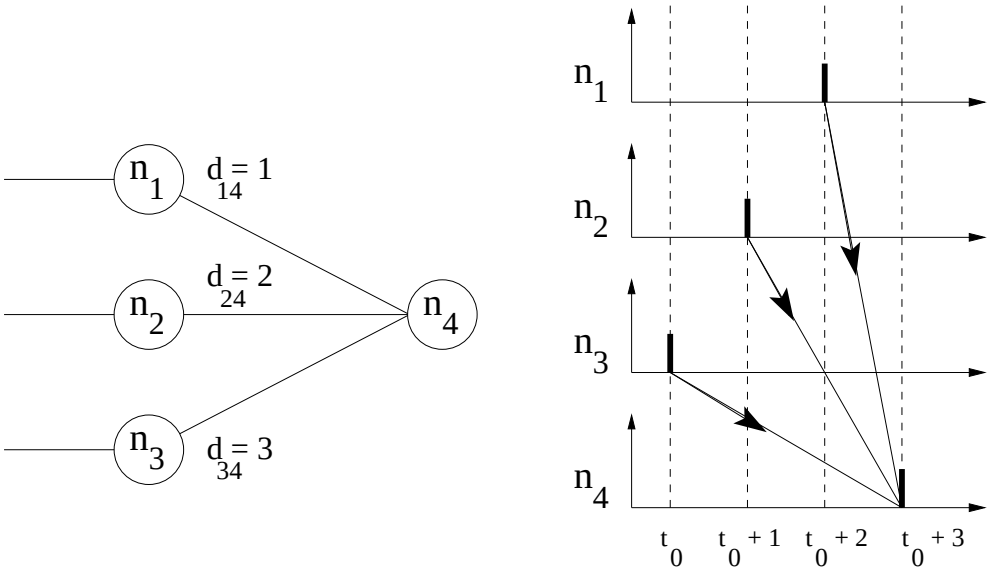

FIG. 1.16 – *Activation d'un groupe de neurones polychrones. La séquence temporelle d'émission des PA des neurones n_1 n_2 et n_3 est reconnue par le neurone n_4 grâce au jeu de délais affecté aux connexions. Le neurone n_4 a un seuil d'émission fixé à trois PPS. L'émission d'un PA par le neurone n_4 indique alors la détection de la séquence d'entrée.*

date d'émission d'un PA. Sur la figure 1.16 par exemple, le neurone n_1 a un délai de 1 ms avec le neurone n_4. Les neurones n_2 et n_3 ont des délais de 2ms et 3ms avec le neurone n_4. Si l'on considère que l'impact simultané de trois PPS suffit pour qu'un neurone émette un PA, alors lorsque n_3 émet un PA à une date t_0, n_2 émet un PA à t_0 + 1 et n_1 émet un PA à t_0 + 2, alors n_4 recevra simultanément les trois impacts à t_0 + 3 et émettra alors un PA.

Comme pour les poids synaptiques, il ne suffit pas de fixer aléatoirement les valeurs des délais pour que l'activité du réseau conduise aux sorties désirées en partant d'un vecteur d'entrées donné. Des méthodes d'apprentissage non-supervisées inspirées de celles utilisées pour les poids synaptiques sont développées pour les délais(e.g. Hüning et al. (1998)).

L'existence et les bases biologiques des modifications des délais en fonction des entrées ne sont pas encore bien claires. Cependant des résultats récents (Bakkum et al. 2008) montrent, lors d'expériences in-vitro, que des modifications s'opèrent sur les délais axonaux en fonction du pattern de stimulation. Certains travaux ont également montré qu'il est possible de faire varier les délais en s'appuyant sur la variation des poids synaptiques (Senn et al. 2002). Bien qu'encore à un stade préliminaire, l'application de méthodes d'apprentissage sur les délais peut augmenter la puissance computationnelle des réseaux modélisés, comme le démontrent des études théoriques (Maass & Schmitt 1999).

1.9 Stratégies d'apprentissage

Les stratégies d'apprentissage à partir d'exemples appartiennent majoritairement aux deux familles suivantes :

– apprentissage supervisé ;
– apprentissage non supervisé.

L'apprentissage supervisé est très utilisé dans les réseaux de neurones classiques basés sur des neurones à seuil ou sigmoïdaux (Perceptron (Rosenblatt 1958), Adaline (Widrow & Hoff 1960), rétropropagation dans les MLP (Rumelhart et al. 1986) ...). Il porte essentiellement sur la modification des poids du réseau à partir d'un ensemble de couples de vecteurs (X,Y), où X est un exemple présenté en entrée et Y est un vecteur de sortie désiré. On souhaite déterminer le meilleur vecteur de poids W qui permette d'obtenir les sorties désirées pour les exemples X fournis. Ce type de méthode d'apprentissage nécessite de disposer d'une base d'exemples, c'est-à-dire d'un ensemble de couples (X,Y). La définition d'une fonction d'erreur permet alors d'évaluer la distance entre un vecteur de sortie calculé et le vecteur de sortie attendu. Les méthodes de moindres carrés ou de descente en gradient sont les méthodes d'apprentissage supervisé les plus classiquement utilisées. Les simulations utilisant un apprentissage supervisé se divisent généralement en deux phases :

1. Une phase d'apprentissage, pendant laquelle le vecteur de poids est modifié, grâce aux méthodes choisies, à partir de la base d'exemples
2. Une phase de généralisation, durant laquelle le vecteur de poids est fixé aux valeurs obtenues suite à la phase d'apprentissage et le réseau est appliqué au traitement de motifs non appris.

La qualité de l'apprentissage réalisé est évaluée par la capacité du réseau à généraliser (Vapnik & Chervonenkis 1971). Par exemple, en classification, les exemples utilisés durant la phase d'apprentissage doivent être déterminés de manière à permettre au réseau de faire une approximation efficace des différentes classes. Le réseau de neurones effectue une séparation de l'espace des entrées possibles en plusieurs régions dont la forme et le nombre dépendent du type de neurones utilisés et de la taille du réseau.

L'apprentissage non supervisé regroupe un large spectre de méthodes d'apprentissage. Les données traitées en apprentissage non supervisé, à l'inverse de l'apprentissage supervisé, ne sont pas associées à des sorties désirées. Sans a priori sur les données, les règles d'apprentissage peuvent se baser par exemple sur la distribution des données en projection sur un ensemble de points tel que la méthode des *K-means* ou k-moyennes (voir Steinhaus 1956). Il s'agit de répartir les données à traiter, les points, en différents *clusters* (ou sous-ensembles) grâce à une heuristique.

Certaines formes d'apprentissage non-supervisé s'appuient sur le principe physiologique énoncé dans la loi de Hebb (1949). Il propose que l'activation conjointe de deux neurones connectés renforce leur connexion, et ainsi facilite leur activation conjointe dans le futur :

> « *When an axon of cell A is near enough to excite B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased* »

La loi de Hebb a été mise en œuvre, sous diverses formes algorithmiques, pour définir les règles d'apprentissage de plusieurs modèles connexionnistes, à commencer par le réseau de Hopfield (1982).

Plus récemment, cette loi a souvent été revisitée, dans un contexte temporel : les modélisations de réseaux de neurones impulsionnels exploitent directement ces principes pour modéliser une forme de *plasticité synaptique* en tant que règle d'apprentissage non-supervisée.

1.10 Plasticité synaptique

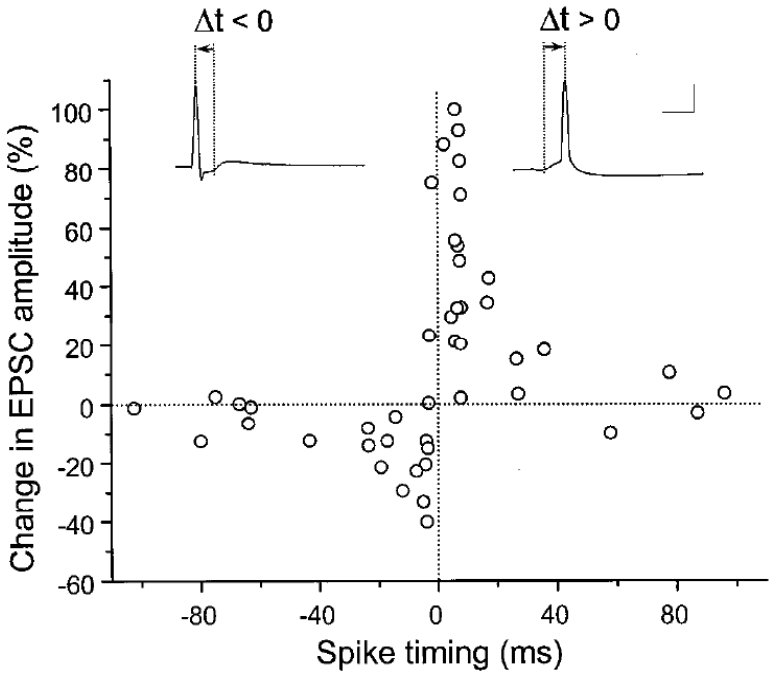

Fig. 1.17 – *Variation d'amplitude du courant post-synaptique (potentialisation et dépression) en fonction des temps d'arrivée des PPS par rapport au PA d'un neurone post-synaptique (EPSC : Excitatory Post-Synaptic Current).* $\Delta t = t_{post} - t_{pre}$*, les instants d'émission des neurones post-synaptique et pré-synaptique. Bi & Poo (1998) obtiennent ces enregistrements 20 à 30 minutes après une répétition de stimulations.*

L'influence, longtemps supposée et désormais avérée en biologie (section 1.1.3), des temps d'émission des PA pré et post synaptiques sur la réponse d'une synapse, a donné naissance à des règles de plasticité synaptique. Un grand nombre de modélisations de plasticité synaptique sont basées sur les instants d'arrivée des PA (ou « Spike Timing Dependent Plasticity » (Abbott & Nelson 2000)).

Une telle plasticité a été constatée en électrophysiologie (Bi & Poo 2001). Les quantités de neurotransmetteurs libérées dans la synapse varient en fonction de l'activité des neurones. Ces quantités dépendent notamment des stocks de neurotransmetteurs disponibles et de la capacité de stockage des vésicules (figure 1.3, page 7). L'ensemble des paramètres interagissant génère des variations dans les formes d'impacts et opère à des échelles de temps diverses.

1. À l'échelle de la seconde, on parle alors de « Short Term Potentiation STP » ou de « Short Term Depression STD » selon qu'il s'agit d'une augmentation

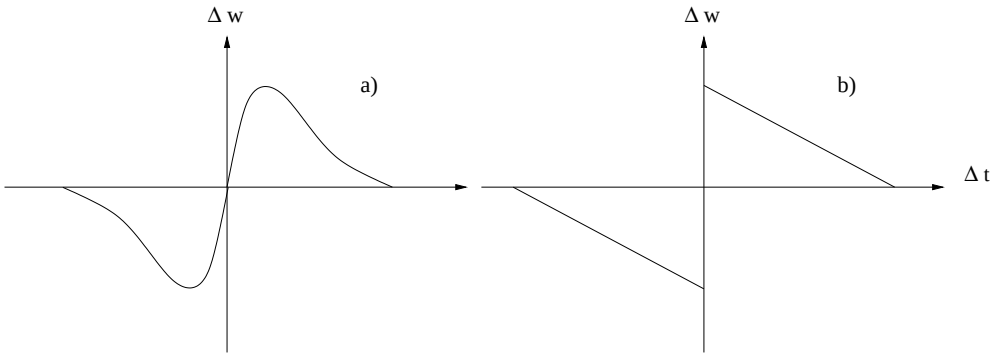

FIG. 1.18 – *Exemples de fenêtres de STDP utilisées dans la littérature. En abscisse, l'intervalle entre l'émission d'un PA post-synaptique et celle d'un PA pré-synaptique, où $\Delta t = t_{post} - t_{pre}$. En ordonnée, Δw, la variation de poids engendrée. a) Fenêtre régulant les intervalles inter PA autour d'un Δt optimal. b) Fenêtre simplifiée, approximant les variations observées en électrophysiologie.*

ou une diminution de l'impact synaptique. À cette échelle, il s'agit principalement de prendre en compte l'adaptation des synapses aux activités pré et post-synaptiques.

2. À l'échelle de plusieurs heures (ou de la journée), on parle alors de « Long Term Potentiation LTP » ou « Long Term Depression LTD ». Dans ce cas les modifications induites dans les synapses subsistent plus longtemps et on considère alors qu'une forme de mémorisation est effectuée.

Des études (Markram et al. 1997, Bi & Poo 1998, Song & Abbott 2001) ont permis de déterminer les variations induites par les impacts des PA pré-synaptiques sur une synapse en fonction de leur date d'impact, dans une fenêtre temporelle autour de la date d'émission d'un PA post-synaptique (figure 1.17). Différents types de STDP (voir figure 1.18) sont dérivés de ces travaux (Kempter et al. 1999, Song et al. 2000, Rao & Sejnowski 2001, Senn et al. 2001, Nowotny et al. 2003, ...) et permettent l'émergence de dynamiques plus proches de celles observées en biologie à l'échelle d'un réseau (Morrison et al. 2008, pour une revue récente).

Par exemple Nowotny et al. (2003) proposent une règle d'apprentissage basée sur les enregistrements de Bi & Poo (1998). La variation du poids d'une synapse $\Delta w(\Delta t)$, avec $\Delta t = t_{post} - t_{pre}$, les instants d'émission du PA post-synaptique et d'impact du PA pré-synaptique, sont définis par la relation :

$$\begin{aligned} \Delta w(\Delta t) &= A_+ \tfrac{\Delta t}{\tau_+} \times e^{-\frac{\Delta t}{\tau_+}} \quad lorsque \quad \Delta t \geq 0 \\ \Delta w(\Delta t) &= A_- \tfrac{\Delta t}{\tau_-} \times e^{-\frac{\Delta t}{\tau_-}} \quad lorsque \quad \Delta t \geq 0 \end{aligned} \tag{1.33}$$

Bien évidemment la seule prise en compte de la temporalité des PA ne peut être suffisante pour expliquer entièrement les phénomènes de mémoire auxquels on associe la plasticité synaptique (Bi & Rubin 2005). La complexité des phénomènes étudiés a de lourdes conséquences sur nos capacités à les modéliser, de par le manque d'informations disponibles d'une part, mais aussi et surtout par le manque de puissance de calcul exploitable par les simulations.

1.11 Modèles de réseaux

L'étude du fonctionnement des neurones biologiques permet de définir des modèles de neurones. Mais lorsque les neurones se trouvent interconnectés en réseaux, des phénomènes (non définis explicitement par les modèles de neurones) émergent tels que des synchronisations transitoires, des oscillations qui sont autant d'interactions entre populations. Depuis le début des années 90, de nombreux modèles de réseaux, basés sur des modèles de neurones non-impulsionnels, ont été développés et étudiés. Ces modèles de réseaux peuvent s'appliquer aux neurones impulsionnels et leur efficacité n'est plus à prouver. D'une manière générale on distingue les réseaux « feed-forward » des réseaux récurrents et la plupart des modélisations d'inspiration biologique sont basées sur des réseaux récurrents.

1.11.1 Réseaux feed-forward

Dans le cas de réseaux dits « feed-forward », l'information circule des entrées vers les sorties et la topologie du réseau ne contient aucun cycle. Les réseaux développés sur ce modèle (e.g. MLP, RBF) sont parfois appelés réseaux à couches du fait que leur traitement est basé sur une succession de couches de neurones.

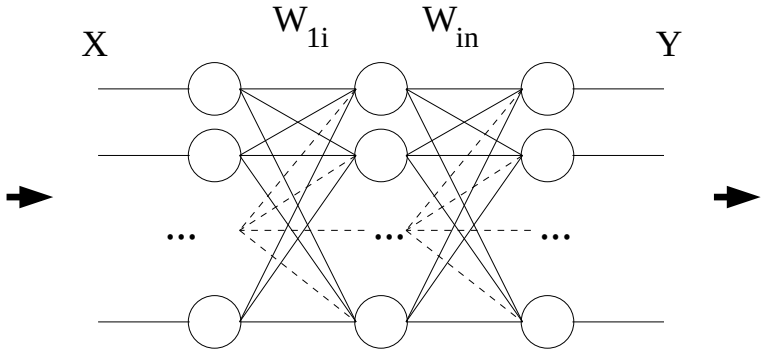

Fig. 1.19 – *Exemple de réseau multicouche. Il peut être constitué de plusieurs couches dites « cachées » entre les couches d'entrée X et de sortie Y. Un vecteur de poids W_{ij} effectue la transformation souhaitée (ou obtenue suite à l'apprentissage) d'une couche à la couche suivante.*

1.11.2 Réseaux récurrents

Un réseau est dit récurrent s'il existe au moins un cycle dans la topologie du réseau. Par exemple, en partant d'un réseau feed-forward, on peut ajouter des connexions « feedback » modulant ainsi les entrées à l'aide des sorties précédentes (figure 1.20 à droite). Cela peut aussi se traduire par des connexions au sein d'une même couche de traitement via des connexions latérales (figure 1.20 à gauche). Les réseaux de (Hopfield 1982), dans lesquels chaque neurone est connecté à tous les autres, sont un exemple de réseaux récurrents.

Dans le cadre plus restreint des modèles de neurones impulsionnels, de nombreux travaux récents cherchent à mieux comprendre comment manipuler ces nouveaux modèles de neurones pour tirer parti de leur puissance computationnelle potentielle.

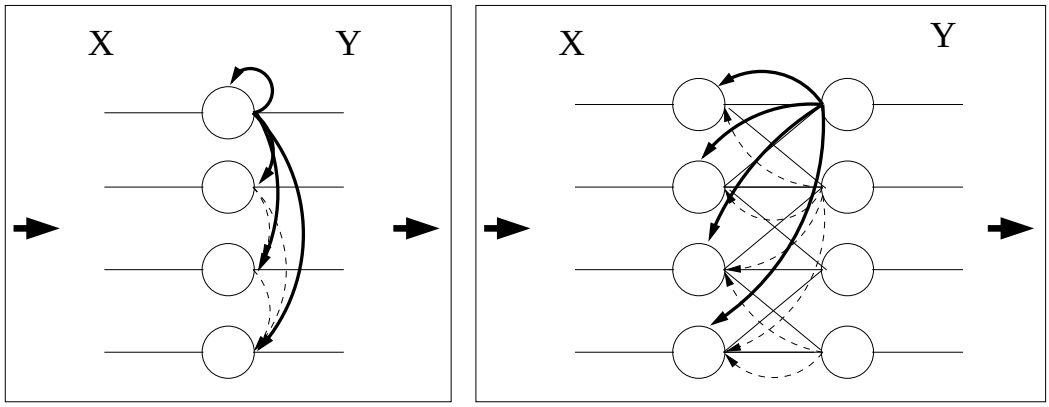

FIG. 1.20 – *Exemple de réseaux. À gauche, réseau à connexions latérales. des connexions latérales (arcs de cercles fléchés) sont ajoutées à un réseau feedforward. À droite, réseau récurrent. À partir d'un réseau feedforward, des connexions top-down sont ajoutées entre les couches (arcs de cercles fléchés).*

Certaines approches tentent d'exprimer, à l'aide d'équations mathématiques, le comportement de populations de neurones (e.g. Cessac et al. (1994), Gerstner & Kistler (2002)). D'autres études utilisant des modèles détaillés au niveau neuronal, utilisent des réseaux de type « sacs de neurones » (ou réservoirs). Ce type de réseaux a peu à peu émergé dans la littérature sous différentes formes telles que les « ESN ; Echo State Networks » (Jaeger 2001) ou bien , « LSM ; Liquid state machine » (Maass et al. 2002) et l'on parle maintenant de « reservoir computing » pour désigner ces approches. Ces réseaux, et plus généralement les réseaux de type réservoir sont des réseaux récurrents, dont les topologies sont basées sur des probabilités de connexions entre les neurones du réseau.

1.11.3 Réseaux d'inspiration biologique

Du fait de leur proximité avec les neurones biologiques, le traitement effectué par les réseaux de neurones impulsionnels peut être comparé à un traitement cortical. La topologie des connexions corticales animales peut alors inspirer les modèles de réseaux étudiés. Par exemple, la définition de connexions d'une couche *plus avancée* dans le traitement de l'information vers une couche *moins avancée* (e.g. Siegel et al. (2000), Mouraud & Paugam-Moisy (2006)) via des connexions dites « Top-down ». Souvent nommées ainsi dans la littérature (Larkum et al. 1999, Hopfinger et al. 2000, Larkum et al. 2001), ces connexions pourraient être porteuses de flux d'informations corticaux relatifs à des traitements dits de haut niveau. Ces connexions sont pour cela souvent associées au contrôle attentionnel (Kastner & Ungerleider 2000). L'étude de la neuroanatomie corticale montre également la répétition de topologies de petites *unités élémentaires* au sein du cortex. Bien que leur fonction reste encore inconnue à ce jour (Swadlow et al. 2002, Horton & Adams 2005), l'ensemble du néocortex serait constitué d'une *juxtaposition* de ces unités appelées *colonnes corticales* (Mountcastle 1978; 1997; 2003). Ces colonnes contiennent des neurones répartis sur les six couches du néocortex. De nombreux types différents de neurones sont présents, mais on retrouve majoritairement certains types de neurones dans certaines couches. Les colonnes sont plus ou moins distinctes et peuvent même être considérées comme un

groupe de « mini-colonnes » (Buxhoeveden & Casanova 2002). Dans les six couches dénombrées, les types de neurones et les différentes règles de projections au sein de ces couches sont étudiées afin d'en comprendre l'utilité fonctionnelle et computationnelle (Bannister 2005, Alexandre 2009).

Lorsqu'il est inspiré du biologique, un réseau de neurones peut donc avoir de nombreuses topologies différentes, selon que l'on modélise des circuits corticaux, sous-corticaux et/ou moteurs. L'étude de modèles de réseaux de neurones impulsionnels inspirés de ces topologies pourrait mener à une meilleure connaissance de leur fonction biologique et permettre une exploitation informatique de celle-ci dans des modèles artificiels. Cependant, la barrière de la taille des réseaux corticaux biologiques vis-à-vis des ressources matérielles nécessaires à leur modélisation pose, encore aujourd'hui, un épineux problème.

1.12 Conclusion

Dans le domaine des neurosciences computationnelles, les simulations de réseaux de neurones impulsionnels permettent, depuis de nombreuses années, de repousser les frontières du mimétisme neuronal. La proximité des principes de fonctionnement de ces modèles avec le fonctionnement biologique permet de l'étudier à un niveau de description inaccessible auparavant.

La complexité des modèles exploitables à l'aide des ressources matérielles actuellement disponibles reste limitée, mais des modèles intermédiaires tels que le modèle LIF proposent un compromis efficace entre la complexité du modèle et la qualité du mimétisme biologique.

Pour atteindre des tailles de réseaux qui soient à l'échelle biologique, i.e. de l'ordre de plusieurs millions de neurones, la puissance de calculs requise nécessite l'utilisation de supports matériels parallèles. La mise à disposition de simulateurs permettant ce type d'exécutions est donc l'une des attentes des neurosciences computationnelles aujourd'hui.

Stratégie de simulation événementielle

2

Les études des comportements des réseaux de neurones impulsionnels s'intéressent par exemple à la comparaison du comportement de réseaux artificiels vis-à-vis de réseaux biologiques (Lytton et al. 1996, Kirkland & Gerstein 1999, Grossberg et al. 2004, Brette & Gerstner 2005, par exemple). On tente alors de valider ou invalider certaines hypothèses faites en neurosciences en réalisant des modèles de réseaux mimant des réseaux corticaux. L'étude des dynamiques ou de la répartition des poids après apprentissage par exemple, offre un point de vue intéressant pour orienter de nouvelles pistes de recherche à partir des modèles.

D'autres travaux s'intéressent à des aspects computationnels tels que les méthodes d'apprentissage, les modélisations de délais de transmission ou de réponses synaptiques à l'aide de simulations de réseaux de neurones impulsionnels et tentent d'en comprendre les influences respectives sur les dynamiques neuronales simulées (par exemple Legenstein et al. (2003), Brette (2006; 2007)).

Après avoir discuté de l'utilité, pour la communauté scientifique, de disposer de simulateurs performants (section 2.1), nous présenterons le concept de « temps virtuel » (section 2.2) au sein des simulations de réseaux de neurones impulsionnels. La stratégie de simulation *clock-driven* sera rapidement présentée (section 2.3) et nous étudierons dans quelle mesure les caractéristiques des simulations de réseaux de neurones impulsionnels permettent d'optimiser la plupart des simulations en utilisant une stratégie événementielle (section 2.4).

2.1 Utilité de simulateurs

Pour mener leurs travaux, deux solutions s'offrent aux chercheurs effectuant des simulations de réseaux de neurones impulsionnels : soit il leur faut utiliser des outils adaptés permettant la manipulation de réseaux de neurones artificiels dans le cadre souhaité ; soit il leur faut développer des outils spécifiques de l'expérience menée. La première solution est clairement préférable à la seconde. Cette dernière implique souvent beaucoup de temps pour le développement informatique des outils logiciels, nécessaires à la modélisation et à la simulation de l'expérience mise en place, d'une part, mais implique surtout une grande spécificité des outils. Cette spécificité mène alors à devoir recommencer un travail similaire pour chaque nouvelle expérience.

C'est pourquoi, de nombreux outils de type « simulateurs » ont vu le jour ces dernières années, ou des outils de type librairies (par exemple Boniface et al. 1999).

Question	NEURON	GENESIS	NEST	NCS	CSIM	XPP	SPLIT	Mvaspike
HH	B.I.	B.I.	YES	B.I.	B.I.	YES	B.I.	POSS
leaky IF	B.I.	POSS	YES	B.I.	B.I.	YES	POSS**	B.I.
Izhikevich IF	YES	B.I.	YES	NO	B.I.	YES	POSS**	POSS**
Cable eqs	B.I.	B.I.	NO	NO	NO	YES	B.I.	NO
ST plasticity	YES	B.I.	YES	B.I.	B.I.	YES	B.I.	YES
LT Plasticity	YES	YES	YES	B.I.	B.I.	YES	NO**	YES
Event-based	B.I.	NO	YES	NO	NO	YES	NO	YES
exact	B.I.	-	YES	-	-	NO	-	YES
Clock-based	B.I.	B.I.	YES	B.I.	YES	YES	YES	POSS**
interpolated	B.I.	NO	YES	NO	NO	YES	B.I.	POSS
G synapses	B.I.	B.I.	YES	B.I.	B.I.	YES	B.I.	POSS**
parallel	B.I.	YES	B.I.	B.I.	NO**	NO	B.I.	NO**
graphics	B.I.	B.I.	NO(*)	NO(*)	NO(*)	YES	NO	NO
simple analysis	B.I.	B.I.	YES	NO(*)	NO(*)	YES	NO	NO
complx analysis	B.I.	YES	NO(*)	NO(*)	NO(*)	YES	NO	NO
development	YES	YES	YES	YES	YES	YES	YES	YES
how many p.	3	2-3	4	2-3	2	1	2	1
support	YES	YES	YES	YES	YES	YES	YES	YES
type	e,p,c	e	e	e	e	e	e	e
user forum	YES	YES	YES	NO	NO	YES	YES	NO
publ list	YES	YES	YES	YES	YES	NO	NO	NO
codes	YES	YES	YES	YES	YES	YES	NO	NO
online manual	YES	YES	YES	YES	YES	YES	YES	YES
book	YES	YES	NO	NO	NO	YES	NO	NO
XML import	NO**	POSS	NO**	NO**	NO	YES	NO	NO**
XML export	B.I.	NO**	NO**	NO**	NO	NO	NO	NO**
web site	YES	YES	YES	YES	YES	YES	YES	YES
LINUX	YES	YES	YES	YES	YES	YES	YES	YES
Windows	YES	YES	YES	YES	YES	YES	NO	NO
Mac-Os	YES	YES	YES	NO	NO	YES	NO	NO
Interface	B.I.	B.I.	POSS	B.I	YES	POSS	POSS	POSS
Save option	B.I.	YES	NO**	B.I.	NO	NO	NO	NO

FIG. 2.1 – *Comparaison des fonctionnalités de différents simulateurs de la littérature. La définition des termes utilisés dans la figure se trouve en annexe A. (D'après (Brette et al. 2007))*

D'une manière générale, tandis qu'une librairie fournit des fonctionnalités plus ou moins complexes à intégrer dans un programme, un simulateur comprend un (ou plusieurs) programme(s) exécutable(s) permettant d'effectuer des simulations sans programmation supplémentaire.

Les simulateurs les plus connus sont :

- GENESIS (Bower & Beeman 1998),
- SPLIT (Hammarlund & Ekeberg 1998),
- SpikeNet (Delorme et al. 1999),
- SpikeNNS (Marian & Reilly 2001),
- NEURON (Hines & Carnevale 2001),
- NEST (Diesmann & Gewaltig 2002),
- MvaSpike (Rochel & Martinez 2003),
- CSIM (Natschläger et al. 2003),
- XPPAUT (Ermentrout 2002),
- Neocortex (Muresan & Ignat 2004),

– NCS (Goodman 2005),
– BRIAN (Goodman & Brette 2008).

Chacun de ces simulateurs possède ses propres forces et faiblesses. Bien évidemment, certains simulateurs sont maintenus et développés par des équipes entières de chercheurs ce qui permet le développement de fonctionnalités variées (modèles prédéfinis, gestion graphique, analyses des données ...). Mais nous verrons que certains atouts des simulations événementielles ne sont que rarement exploités dans les simulations. Brette et al. (2007) rapportent les fonctionnalités des simulateurs les plus connus à ce jour sur la figure 2.1.

Quelle que soit l'expérience menée, un simulateur est utile pour ce qu'il apporte de « déjà implémenté » mais il doit également laisser le plus de latitude possible aux utilisateurs dans les trois domaines suivants :

1. La diversité des réseaux de neurones simulables
2. La diversité des protocoles d'expérience simulables
3. Les données de sortie récupérables

Un simulateur doit donc avant tout offrir un maximum de souplesse pour le choix des expériences réalisables. Par ailleurs, mettre à disposition des outils pré-construits (tels que des modèles de neurones, des modèles de réseaux, des règles d'apprentissage ...) et permettre la conservation des données de simulations déjà réalisées doivent également être des priorités pour un simulateur. Enfin, il est également important de pouvoir manipuler des facteurs aléatoires, utiles dans les simulations, et de pouvoir en contrôler précisément les effets.

2.1.1 Diversité des réseaux

Pendant le déroulement de la simulation les dynamiques des neurones et des synapses requièrent la conservation de nombreuses variables telles que les potentiels de membranes et les poids synaptiques.

Cependant à la création du réseau, ces variables ne sont pas spécifiées une à une. Des règles de connexions sont utilisées. Ces règles sont la plupart du temps peu nombreuses et sont appliquées à tout le réseau, ou à des sous-réseaux. Les modèles de neurones obéissent également à des règles de fonctionnement génériques : un même modèle est utilisé pour un grand nombre de neurones dans un réseau, et peu de modèles coexistent dans une simulation. Ceci implique que la définition d'un réseau doit pouvoir se faire par un ensemble de règles génériques décrivant la marche à suivre pour la création du réseau.

En partant du concept simple de « couche » de neurones, on peut définir un modèle de réseau aussi spécifique que nécessaire.

Définissons une couche comme un tableau de neurones n_i, $0 \leq i < N$, où N est le nombre de neurones dans la couche. Fournissons à cette couche un ensemble de règles de projections. Une règle de projection définit les paramètres des connexions synaptiques entre les neurones d'une couche *pre* et ceux d'une couche *post*. Ces projections peuvent être de nombreux types tel que des projections contrôlées par des Gaussiennes (e.g. champs récepteurs), des projections totales où chaque neurone *pre*

Algorithme 1 : Définition d'un réseau

1 - Nombre de couches : 3
2 - Taille des couches : 10 50 10
3 - Projections :
4 - couche 1 → couche 1 : aucune
5 - couche 1 → couche 2 : gauss
6 - couche 1 → couche 3 : aucune
7 - couche 2 → couche 1 : aucune
8 - couche 2 → couche 2 : totale
9 - couche 2 → couche 3 : aléatoire
10- couche 3 → couche 1 : aléatoire
11- couche 3 → couche 2 : totale
12- couche 3 → couche 3 : aucune

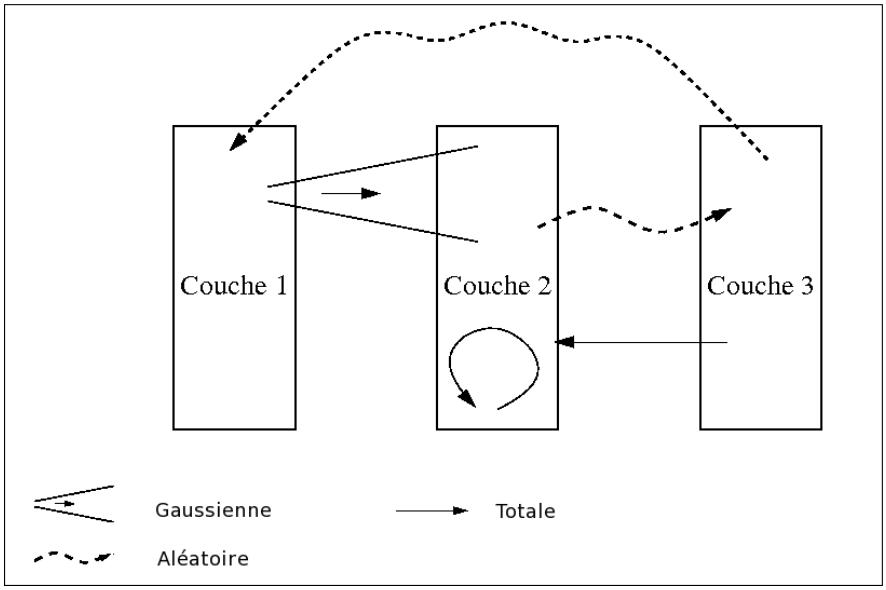

FIG. 2.2 – *Exemple de réseau défini par l'algorithme 1. Le réseau contient des projections aléatoires, des projections sous forme de Gaussienne, ainsi que des projections totales.*

fait une connexion avec chaque neurone *post*, ou des projections aléatoires ... Une projection sera dite intra-couche si les couches *pre* et *post* sont la même couche.

Le choix du terme couche ne doit pas induire en erreur ici ; on peut les considérer comme des ensembles de neurones non ordonnés, ou bien comme des vecteurs uni-dimensionnels, bi-dimensionnels ou tri-dimensionnels, seules les règles de projections appliquées et l'affichage graphique (s'il existe) du réseau définiront visuellement le schéma des projections.

L'important pour un simulateur est de permettre ce type de définition en laissant un maximum de liberté dans la création des topologies de réseaux.

À partir de cette définition d'une couche, on peut générer facilement un réseau d'architecture quelconque. Les modèles de réseaux classiques (MLP, Hopfield, SOM ...) sont modélisables de cette manière. Chaque couche possédera des règles de pro-

jections intra-couche ainsi que des règles de projection inter-couches. On peut par exemple définir la topologie du réseau de la figure 2.2 avec les quelques lignes de l'algorithme 1.

Dès lors que le réseau souhaité devient plus complexe, on peut ajouter des précisions sur le modèle de neurone utilisé pour chaque couche et avec des règles de projection spécifiques on peut définir ainsi très simplement une topologie de réseau très élaborée. La plupart des simulateurs utilisent ce type de définition à l'aide de balises inscrites dans un fichier de configuration. De telles méthodes inspirées du langage XML (Van der Vlist 2002) sont fournies dans certains simulateurs. Elles sont utiles pour leur généricité et leur portabilité même si l'importation/exportation de configurations en XML (voir figure 2.1) est rarement implémentée (Hines et al. 2004, Qi & Crook 2004).

2.1.2 Protocoles de stimulations d'un réseau

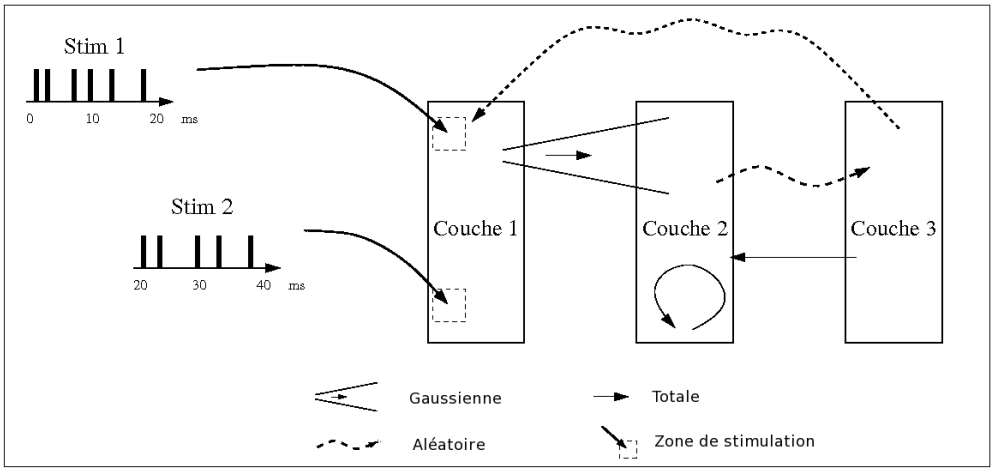

FIG. 2.3 – *Exemple de deux trains de PA : Stim 1 et Stim 2, envoyés sur deux zones distinctes de la couche 1 du réseau.*

Le réseau défini précédemment pourrait être utilisé pour une simulation dans le cadre d'un protocole d'expérience prédéfini. Par exemple, considérons qu'il représente un schéma cortical dans une modalité sensorielle. La simulation pourrait consister à étudier les réponses de ce réseau à une série de stimuli. Ces stimuli seraient appliqués sous forme de courants électriques directement sur les neurones ciblés (à la manière d'une électrode) ou bien sous forme de trains de PA reçus par les neurones ciblés au niveau de leurs dendrites.

Poursuivons notre exemple en appliquant au réseau ainsi défini une succession de deux stimulations chacune constituée d'un train de PA. Le pattern temporel des PA est spécifique de chaque stimulation, ainsi que la zone du réseau stimulée (voir figure 2.3). Il est nécessaire que le simulateur facilite la définition de tels protocoles, afin que l'expérimentateur puisse effectuer rapidement une série de simulations sans redéfinir le protocole à chaque fois. Dans le même temps et à l'instar des paramètres de réseau (section 2.1.1) il faut également que le simulateur laisse le maximum de liberté dans les types de stimuli modélisables.

2.1.3 Données de simulation récoltées

Enfin, si l'on souhaite effectuer une simulation d'un réseau de neurones, c'est avant tout afin d'étudier certaines données tels que les émissions de PA, par exemple pour créer ce que l'on nomme un *raster*. Un *raster* est une figure sur laquelle sont tracés tous les PA émis par des neurones (un point par PA, voir figure 2.4). L'un des avantages des simulations sur les enregistrements biologiques, est que l'on peut également étudier l'évolution des potentiels de membrane, l'évolution des variables synaptiques, ou tout autre donnée définie dans les modèles.

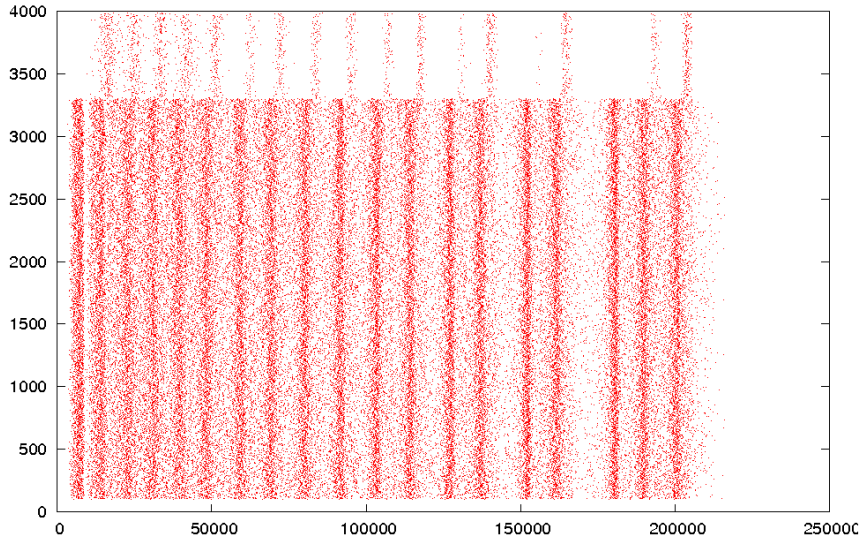

FIG. 2.4 – *Exemple de raster obtenu en exploitant les données récupérées après une simulation. Dans ce cas, les données utiles sont les instants d'émission de PA de chaque neurone du réseau. Il s'agit d'un réseau de 4000 neurones. Le temps est indiqué en abscisse (ms) et le numéro du neurone émettant un PA est indiqué en ordonné. Chaque PA est matérialisé par un point sur la figure.*

Un simulateur doit permettre aux utilisateurs d'avoir accès aux données qu'il souhaite récupérer. Certaines expériences nécessiteront même d'avoir accès à ces informations durant la simulation.

Les spécificités de chacun des simulateurs existant dans la littérature restreignent le cadre des simulations possibles pour chacun et ainsi, selon le problème considéré, l'un ou l'autre de ces simulateurs sera le mieux adapté. En particulier, la modélisation du temps dans la simulation est un élément principal pour distinguer les simulateurs.

2.2 Notion de « temps virtuel »

Des phénomènes temporels sont étudiés dans les simulations de réseaux de neurones impulsionnels. Les simulations informatiques discrétisent le temps selon l'échelle temporelle choisie. Plus généralement, la gestion du temps « virtuel » dans les simulations a une importance particulière.

Dans l'exemple précédent on enregistre l'évolution d'un réseau de neurones stimulé durant 40 ms (deux stimulations successives de 20 ms, voir figure 2.3). Il faut alors être en mesure de décrire l'évolution de tous les paramètres du réseau pendant

cet intervalle de temps. En moyenne, on considère qu'une précision de l'ordre du dixième de milliseconde permet une bonne approximation. Dans la littérature des précisions de 10^{-3} s à 10^{-6} s sont le plus couramment utilisées.

Pour les modèles de neurones non-impulsionnels qui ne tiennent pas compte explicitement du temps, obtenir la sortie d'un réseau en réponse à une entrée se fait « instantanément » . Chaque neurone traite l'entrée qui lui est fournie pour obtenir sa sortie. L'ensemble des sorties des neurones fournit la sortie du réseau.

Les modèles de neurones impulsionnels permettent d'identifier ce temps virtuel à un temps réaliste, parfois nommé « temps biologique », dans la littérature. L'entrée et la sortie de ces neurones ne sont plus des réels (ou entiers ou booléens...) représentant des fréquences de décharge mais des instants d'émission de PA. La fonction seuil devient une équation qui décrit l'évolution du potentiel de membrane en fonction du temps.

2.3 Simulation en temps discret ou « Clock-driven »

Il existe deux stratégies pour la gestion du temps dans les simulations de réseaux de neurones impulsionnels. Dans les deux cas, il s'agit de simulations discrètes (par opposition à continues). Une première stratégie, synchrone, est basée sur un découpage du temps en petites fenêtres temporelles de taille Δt (par exemple $1ms$). La figure 2.5 illustre ce découpage avec un potentiel de membrane quelconque (obtenu par exemple grâce aux équations définissant l'un des modèles du chapitre 1). La simulation est alors dite « clock-driven » ou dirigée par une horloge (également appelé « time-driven »), et consiste à évaluer l'état de tous les neurones du réseau à chaque *pas de temps* d'une horloge. Une évaluation de l'état de toutes les synapses est également effectuée si les réseaux simulés utilisent des variables synaptiques le nécessitant, comme des conductances unitaires par exemple (voir section 1.7).

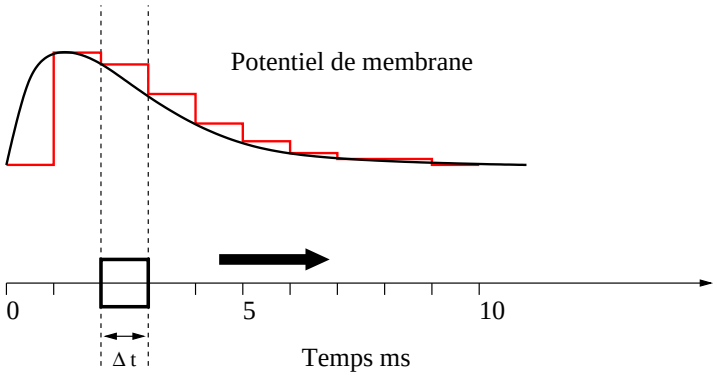

Fig. 2.5 – *Découpage du temps dans les simulations « clock-driven ». Une fenêtre temporelle élémentaire de taille Δt glisse et parcours tous les pas de temps δt_i de taille Δt. A chaque instant, l'état de tous les neurones du réseau est évalué. L'évaluation commence par exemple en une approximation de la valeur du potentiel de membrane au pas de temps considéré.*

Pour des réseaux dont les dynamiques neuronales sont décrites par des équations différentielles, dans une simulation « clock driven » on effectue une approximation (Euler ou Runge-Kutta (Press et al. 1992) en général) de la valeur du potentiel de

membrane au pas de temps δt_i considéré. Cette approximation peut générer des erreurs, comme des décalages de PA si le seuil est franchi plus tard du fait de l'approximation. Il peut également s'agir de PAs manquants, du fait de précédents décalages, ou d'un franchissement de seuil omis (si les courbures des influences synaptiques sont fortes en regard du pas d'itération Δt choisi). Les répercussions globales de telles erreurs sont difficiles à déterminer (Shelley & Tao 2001, Lee & Nabil H. Farhat 2001). Ces répercussions peuvent être, à l'échelle du réseau, des perturbations dans les synchronisations ou dans les phases relatives d'émission de PA (Hansel et al. 1998). Il peut également y avoir des répercussions sur les apprentissages dépendant de ces PA (par exemple STDP, voir section 1.9).

D'une manière générale, une simulation clock-driven consiste à réaliser l'algorithme 2 avec des pas de temps δt_i d'une durée Δt :

Algorithme 2 : Algorithme de base d'une simulation *clock-driven*

```
pour i de 0 à T par pas de Δt faire
    pour j de 0 à N par pas de 1 faire
        Mise à jour du neurone n_j au pas de temps δt_i
        si émission de PA alors
            Propagation vers C_j neurones cibles
        fin
    fin
fin
```

Avec T la durée de simulation, et N le nombre de neurones. La mise à jour consiste, pour chaque neurone n_j, à :

- mettre à jour les variables d'état dendritiques (s'il en est) et membranaires entre δt_{i-1} et δt_i
- prendre en compte les PA pré-synaptiques reçus à δt_{i-1} pour mettre à jour les variables d'état du neurone n_j et ajouter ce neurone à la liste des neurones émettant un PA à δt_i si ce neurone passe le seuil.

L'étape de propagation utilise la liste des neurones émettant à δt_i et propage chaque PA vers ses C_j neurones cibles en appliquant l'impact synaptique sur le potentiel de membrane des neurones cibles.

2.3.1 Complexité

A chaque pas de temps, la complexité de la mise à jour des variables d'état des neurones du réseau est de l'ordre du nombre de neurones N si le nombre de variables d'états dans les modèles de réponses synaptiques définis ne dépend pas du nombre de synapses (par exemple une seule conductance synaptique par neurone). Si les conductances sont spécifiques à chaque synapse, la complexité est de l'ordre du nombre de synapses S. Si on considère un taux de décharge moyen F des neurones du réseau, la complexité de l'étape de propagation est de l'ordre de $F \times N \times C$, chaque neurone émettant à une fréquence moyenne F et chacun de ces PA impactant C neurones cibles (en moyenne).

Donc pour simuler une durée d'une seconde biologique, la complexité est alors

de l'ordre de :

$$\text{Étape 1 : a) } c_{maj} \times \frac{N}{\Delta t}, \text{ ou b) } c_{maj} \times \frac{S}{\Delta t} \tag{2.1}$$

$$\text{Étape 2 : } c_{prop} \times F \times N \times C \tag{2.2}$$

c_{maj} et c_{prop} représentent respectivement le coût d'une mise à jour et d'une propagation de PA. Donc dans le cas a), le moins coûteux, on obtient une complexité :

$$c_{maj} \times \frac{N}{\Delta t} + c_{prop} \times F \times N \times C \tag{2.3}$$

Notons ici que le cas a), « le moins coûteux » n'apparaît pas comme le plus pertinent. En effet, il est souvent avancé que les variables d'états des réponses synaptiques (conductances, courants ...) sont mathématiquement équivalentes (Brette et al. 2007, par exemple) à une unique variable synaptique pour le neurone puisque le neurone fait la somme des conductances de ses dendrites. Mais cela revient à considérer que les dendrites peuvent être assimilées à une unique dendrite, et donc que la répartition des PA entrant sur les dendrites n'a aucune influence sur le neurone. Ainsi, bien que cette approximation soit faite la plupart du temps, elle doit être considérée comme très limitative de l'information traitée par le neurone. La figure 2.6 illustre un exemple de conséquence possible.

Ce type de situation peut également se produire lorsque les conductances synaptiques individuelles sont constantes et identiques si celles-ci ont des dynamiques non linéaires. Cependant, d'un point de vue biologique, il semble pertinent de considérer que les constantes de temps synaptiques (et/ou les conductances) varient indépendamment, éventuellement de façon linéaire, en fonction de leurs propres entrées. L'approximation largement admise utilisant une unique variable synaptique interdit pourtant ce type de modèles.

Le terme de l'étape 2, qui est souvent considéré comme prépondérant, peut alors devenir négligeable par rapport à celui de l'étape 1, si l'on considère le cas b) dans l'équation 2.1 :

$$c_{maj} \times \frac{S}{\Delta t} + c_{prop} \times F \times N \times C \tag{2.4}$$

Alors le second terme est négligeable si :

$$\frac{c_{prop}}{c_{maj}} \times F \times \Delta t \ll 1 \tag{2.5}$$

Ainsi, avec par exemple le coût élémentaire d'une mise à jour plus important que celui d'une propagation $c_{prop} < c_{maj}$, une précision temporelle de l'ordre de la milliseconde $\Delta t = 1 \times 10^{-3} s$, et un taux de décharge moyen de $1Hz$, on obtient :

$$\Delta t \times F \times \frac{c_{prop}}{c_{maj}} < 1 \times 10^{-3} \tag{2.6}$$

La précision temporelle choisie peut donc rendre le premier terme prépondérant. La complexité de la simulation pourra dépendre majoritairement du terme de mise à jour.

La plupart des simulateurs de la littérature utilisent une gestion du temps clock-driven (voir figure 2.1). Par exemple SpikeNet (Delorme et al. 1999) qui a la particularité de s'intéresser au classement temporel des premiers PA émis par une couche

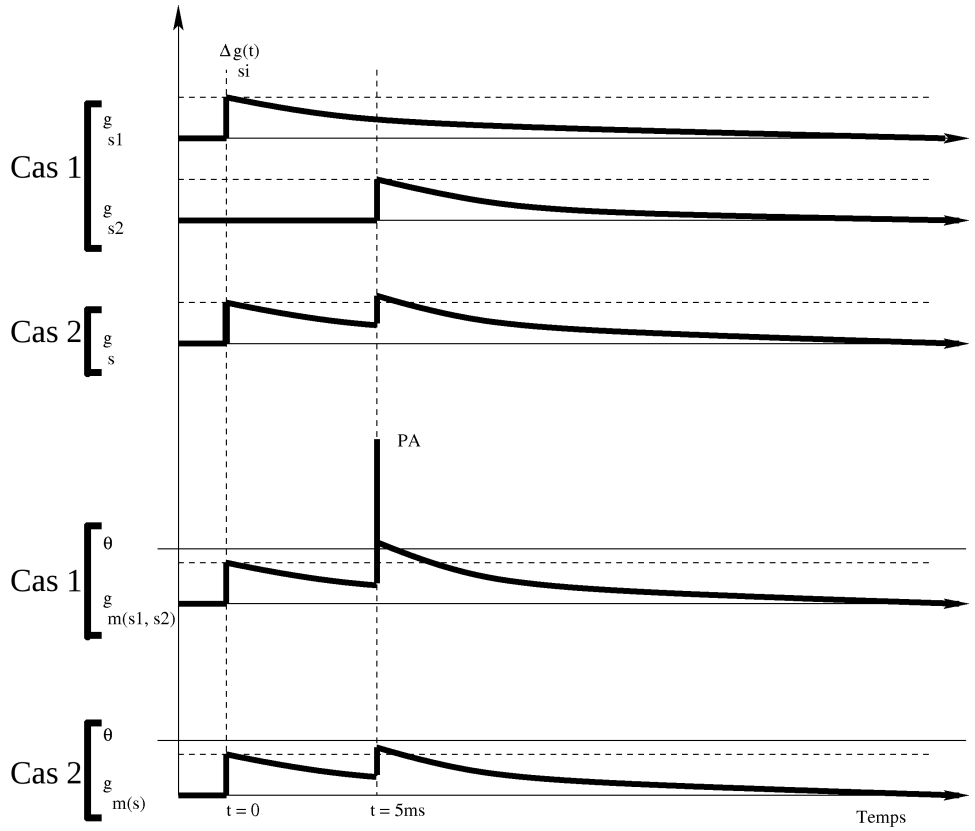

FIG. 2.6 – *Comparaison des dynamiques de conductance membranaire g_m d'un neurone n dans deux situations. Dans le premier cas, deux synapses distinctes possèdent chacune leur propre conductance (courbes g_{s1} et g_{s2}). A réception d'un PA, à $t = 0$ pour la synapse s_1 et à $t = 5ms$ pour s_2. Une variation de conductance $\Delta g_{si}(t)$ est ajoutée. Elle dépend de l'activité pré-synaptique et possède une fuite vers une valeur « de repos » (voir section 1.7). Les synapses s1 et s2 ont des conductances de repos g_{s1} et g_{s2} (deux courbes du haut). La variation de conductance de ces synapses est définie par $\Delta g_{s1}(t)$ et $\Delta g_{s2}(t)$. Dans le second cas, une unique conductance synaptique g_s est considérée, son comportement est décrit par la somme des conductances g_{s1} et g_{s2}, de même pour la variation d'amplitude $\Delta g_s(t)$ qui dépend de $\Delta g_{s1}(t)$ et $\Delta g_{s2}(t)$. Dans la situation présentée, deux PA pré-synaptiques arrivent à $t = 0$ et $t = 5ms$. Dans le premier cas, le premier PA arrive sur la synapse s1 et le second sur la synapse s2. Dans le second cas, l'unique synapse reçoit les deux PA. La comparaison des dynamiques de membranes associées illustre bien les différences qui peuvent être la conséquence de l'utilisation ou non de variables synaptiques individuelles. Ici, un PA est émis à $t = 5ms$ par le neurone dans le cas où les deux synapses possèdent une conductance synaptique spécifique.*

de neurones en réponse à une entrée donnée (une image par exemple). Les simulateurs NEURON (Hines & Carnevale 2001) et GENESIS (Bower & Beeman 1998) utilisent également une gestion clock-driven et sont très performants pour effectuer des simulations de neurone unique ou de couples de neurones car ils permettent une modélisation précise du neurone (géométrie, canaux ioniques, compartiments multiples). Cependant, le coût computationnel de simulations de ce type de modèles en limite l'utilisation.

2.3.2 Paramètres biologiquement plausibles dans les réseaux de neurones impulsionnels

Par ailleurs, le taux de connectivité du réseau simulé (C) de même que l'activité moyenne du réseau (F) et la précision du pas de temps (Δt) durant la simulation sont des paramètres qui influencent fortement les temps nécessaires à la simulation (équations 2.1 2.2 et 2.3).

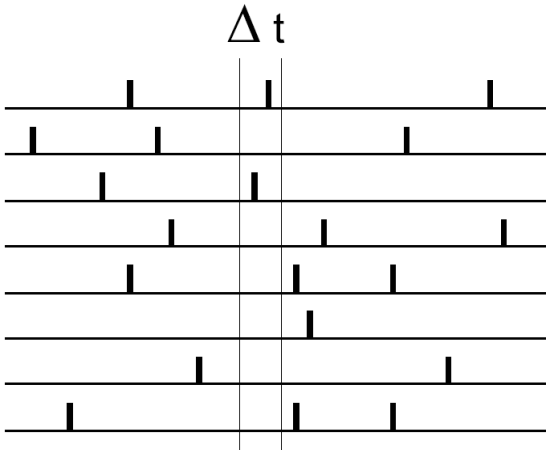

FIG. 2.7 – *Les PA émis par un neurone du réseau sont représentés sur un raster. Deux neurones seulement émettent un PA dans l'intervalle de durée Δt représenté.*

Dans des simulations d'inspiration biologique, les taux de connectivité et l'activité neuronale sont généralement faibles. Un total d'environ 10^{10} neurones dans le cerveau dotés de 10^4 connexions chacun implique une connectivité d'environ $0,000001\%$. L'activité moyenne des réseaux est de l'ordre de 1 à 10Hz en moyenne sur l'ensemble du réseau.

Dans de telles conditions, un PA émis par un neurone impulsionnel au sein d'un réseau génère alors des modifications d'état sur une faible proportion du réseau. De plus une activité moyenne faible implique que peu de neurones seulement seront actifs à chaque pas de temps comme l'illustre la figure 2.7. Dès lors, l'évaluation exhaustive (selon les équations (2.3) ou (2.4)) de l'état de tous les neurones, et éventuellement de toutes les synapses, à chaque pas de temps, représente une perte de temps considérable.

2.4 Simulation événementielle

Une seconde stratégie dite « événementielle » considère les neurones impulsionnels comme des entités échangeant des événements. Ce type de simulations est aussi appelé plus largement *event-driven*, à événements discrets (DES ; *Discrete Event Simulation*). Le principe de simulation événementielle est utilisé depuis de nombreuses années dans d'autres domaines tels que dans les systèmes d'exploitation où le fonctionnement est basé sur des traitements d'événements, mais l'utilisation de ce principe pour effectuer des simulations de réseaux de neurones artificiels est plus récente Watts (1994).

Depuis lors, l'intérêt est croissant pour les simulations basées sur un fonctionnement événementiel. En premier lieu, examinons en quoi consiste une simulation événementielle afin de comprendre son intérêt dans le cadre des neurones impulsionnels.

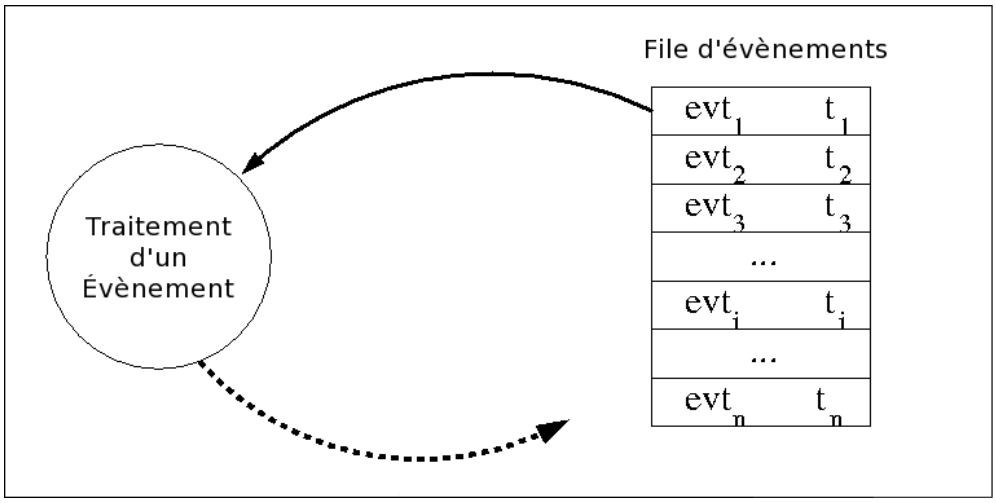

FIG. 2.8 – *Une simulation événementielle consiste à traiter des événements. Les événements sont stockés dans une file de priorité. Le prochain traitement sera fait sur l'événement le plus prioritaire de la file. L'ordre de priorité peut être, par exemple, les dates d'émission t_i des événements. Chaque traitement, peut entraîner l'insertion d'un (ou plusieurs) nouvel événement dans la file.*

2.4.1 Principe événementiel

Le principe de base d'une simulation événementielle consiste à traiter, l'un après l'autre, tous les événements d'une file. La simulation s'arrête lorsque la file est vide. Le type de traitement effectué peut être très variable, il dépend de la simulation. Le traitement d'un événement peut entraîner la création d'autres événements insérés à leur tour dans la file (voir figure 2.8).

Les événements sont triés à l'intérieur de la file. Le type de tri dépend de la simulation effectuée. Le premier événement de celle-ci est le plus prioritaire des événements. L'insertion d'un événement dans la file nécessite alors de prendre en compte la priorité de cet événement afin de l'insérer à une position adéquate. On appelle ainsi communément ces files des « files de priorité ». Les files de priorités utilisées ont généralement des implémentations en tas (*heap*) ou en arbres afin d'optimiser les méthodes d'accès au « prochain » événement ainsi que l'insertion d'événements.

Cas des simulations de neurones impulsionnels

Dans le cas des réseaux des neurones impulsionnels, les événements stockés dans la file sont des PA. L'ordre de ces PA dans la file est déterminé par la date d'émission prédite pour ces PA. Le traitement de l'événement en tête de file correspond alors au traitement du plus proche PA émis dans le futur.

L'algorithme 3 décrit le déroulement d'une simulation. La condition fin peut dé-

pendre de paramètres autres que la présence ou non d'événements dans la file tels qu'une durée maximale pour la simulation ou un nombre d'événements traités ...

Algorithme 3 : Algorithme d'une simulation *event-driven*

tant que $!fin$ et $!file.estVide$ **faire**
  Récupérer le prochain événement dans la file
  Mettre à jour les variables d'état des neurones ciblés
  et celles du neurone émetteur.
  Insérer dans la file les éventuels PA générés.
fin

2.4.2 Complexité

En réutilisant les termes introduits à la section 2.3.1, le coût computationnel d'une simulation événementielle repose sur trois opérations de base : c_{maj}, le coût d'une mise à jour des variables d'un neurone, c_{pre}, le coût de la prédiction du prochain PA émis par un neurone et c_{file}, le coût de l'insertion d'un événement dans la file. L'extraction du prochain événement dans la file a un coût négligeable. S'il y a N×F PA émis durant la simulation, alors on peut approcher le coût computationnel total de la simulation par l'expression :

$$(c_{maj} + c_{pre} + c_{file}) \times N \times F \times C \tag{2.7}$$

Cette équation vaut dans le cas où une prédiction d'émission est nécessaire. C'est le cas lors de l'utilisation de modèles de neurones à plusieurs équations différentielles (section 2.4.4), mais également lors de l'utilisation de réponses synaptiques non-instantanées (section 2.4.5)). Dans les autres cas, le terme c_{pre} disparaît.

Complexité en événementiel par rapport au clock-driven

Si l'on compare les expressions (2.7) et (2.4), représentant les complexités dans les cas événementiel et *clock-driven*, on remarque l'absence du premier terme dépendant du nombre de neurones ou du nombre de synapses suivant les types de réponses synaptiques modélisées (voir section 2.3.1). Ainsi la complexité de la simulation événementielle devient « indépendante » (pas tout à fait comme nous le verrons dans la section 2.4.3) de la précision temporelle (Δt dans l'équation (2.4)).

Dans les simulations *clock-driven*, le premier terme peut devenir prépondérant, comme nous l'avons vu dans la section 2.3.1, les valeurs relatives de c_{prop} d'une part et de la somme de $c_{maj}+ c_{pre}+ c_{file}$ sont déterminantes de l'intérêt d'utiliser ou non une méthode *event-driven* ou événementielle.

Certains travaux utilisent des tables statiques contenant des plages de valeurs possibles des fonctions tabulées décrivant les variables neuronales (Reutimann et al. 2003, Morrison et al. 2005, Carrillo et al. 2007) (aussi appelées *Look up tables*, (Grassmann & Anlauf 1998, Reutimann et al. 2003, Morrison et al. 2005, Ros et al. 2006, Carrillo et al. 2007)) ce qui permet, lors de la simulation, de remplacer le calcul de certaines valeurs (telles que les exponentielles) par un simple accès mémoire et ainsi de réduire le coût

computationnel global, en agissant sur les valeurs c_{maj} et c_{pre}. Cependant ces techniques impliquent une approximation dépendant de la précision de ces tables, ainsi qu'une charge mémoire supplémentaire pour stocker toutes les valeurs souhaitées aux dépends du reste de l'implémentation.

Limitation

Dans le second terme de l'équation (2.4), la constante c_{prop} est remplacée par la constante $(c_{maj} + c_{pre} + c_{file})$ dans l'équation (2.7). Si l'on se place dans le cas où aucune prédiction n'est nécessaire, il reste le terme c_{maj} dont le coût en événementiel est équivalent au coût *clock-driven*, et le terme c_{file}. Les structures de données utilisées pour implémenter les files de priorité classiques sont des tas (tels les *dequeue* de la STL du langage C/C++).

Le coût de l'insertion d'un élément dans ce type de file est en $\mathcal{O}(ln(n))$ où n est le nombre d'éléments dans la file. Ainsi, le nombre d'événements et la gestion en files de priorités qui en est faite, sont des paramètres déterminants dans une simulation événementielle. L'utilisation de *calendar queues* (Brown 1988) peut rendre le coût d'insertion constant (par exemple Claverol et al. (2002)). Nous présenterons rapidement les méthodes de la littérature dans la section 2.4.6 et nous détaillerons une méthode complète de ce type au chapitre 4.

D'autre part, lorsque les paramètres des modèles de neurones et de synapses choisis rendent la prédiction des PA nécessaire, alors la valeur de c_{pre} peut devenir prépondérante dans le coût global de la simulation.

2.4.3 Intérêts de la méthode événementielle

L'intérêt principal de la méthode événementielle est d'éviter les mises à jour de tous les neurones à chaque pas de temps d'une horloge. Des calculs sont effectués uniquement pour traiter des Potentiels d'Actions et leurs impacts. À l'inverse d'une simulation basée sur une horloge, une simulation événementielle est tout particulièrement adaptée à des réseaux d'inspiration biologique (voir section 2.3.2). Les bases de la gestion événementielle ont été bien développées et plus particulièrement dans le cadre de simulations parallèles (Ferscha 1996, Marin 2000a).

La stratégie événementielle a également l'avantage de la simplicité, au moins apparente, mais son implémentation soulève quelques difficultés, par exemple au niveau de la gestion des réponses synaptiques ou des délais, comme on le verra dans les sections qui suivent.

Un argument supplémentaire en faveur d'une stratégie événementielle est que la simulation n'est plus limitée à la précision du pas de temps minimal, comme dans une simulation *clock-driven*. Il est alors théoriquement possible de réaliser des calculs exacts pour les instants d'émission de PA. Bien évidemment, d'un point de vue informatique, cette exactitude est toute relative puisque le calcul sur ordinateur est inévitablement limité par la précision des nombres utilisés. Plus on utilise une précision fine, plus on se rapproche d'une simulation exacte, mais plus les ressources computationnelles nécessaires à la simulation deviennent importantes.

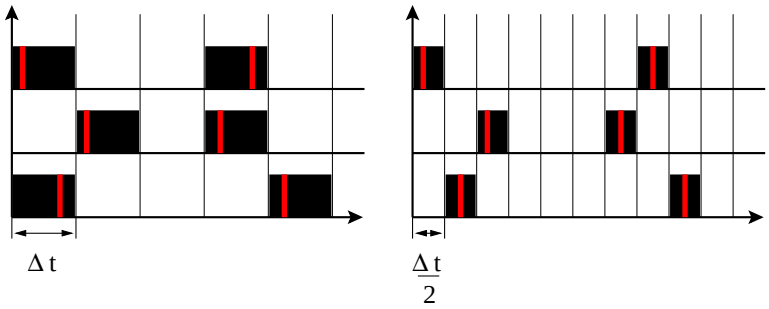

FIG. 2.9 – *À gauche, simulation événementielle avec une précision de Δt. Trois neurones émettent chacun deux PA (impulsions en Rouge). Dans ce cas, une simulation event-driven fera des mises à jour dans le réseau de neurones dans 4 des 5 pas de temps de la simulation (en Noir). À droite se trouve la même simulation que précédemment avec une précision de $\frac{\Delta t}{2}$. Recalculer les instants d'émission avec cette nouvelle précision peut mener au raster présenté à droite (en Noir). Les PA occupent désormais 6 pas de temps au lieu de 4.*

Influence de la précision

Augmenter la précision temporelle dans une simulation événementielle a moins d'impact que dans une simulation basée sur une horloge, car tous les pas de temps ne sont pas « explorés » dans le cas événementiel. L'activité du réseau définira quels « pas de temps » (ceux de la précision utilisée pour les calculs) seront utiles dans la simulation. Sur la figure 2.9 par exemple, augmenter la précision augmente le nombre de pas de temps explorés (de 4 à 6) mais augmente également le nombre de pas de temps inexplorés (de 1 à 4).

Il faut néanmoins relativiser le gain potentiel, entre simulation événementielle et basée sur une horloge, lorsque l'on augmente la précision. En effet dans la plupart des simulations, des facteurs aléatoires, sur les poids, les délais ou sur les stimulations par exemple, peuvent « répartir » les *spikes* émis pour homogénéiser l'activité du réseau. Ainsi, plus la précision augmente, plus la dispersion des PA peut s'opérer, et ainsi occuper l'espace des pas de temps possibles. De plus, des arrondis sont effectués sur le calcul des variables. Dans l'exemple de la figure 2.9, en utilisant une précision de Δt, 4 des 5 pas de temps sont occupés par des PA. Une simulation événementielle fera donc des mises à jour de neurones dans 80% des pas de temps qui seraient explorés par une simulation clock-driven (voir aussi la figure 2.7). De plus une simulation clock-driven fera une mise à jour de tous les neurones à chaque pas de temps alors que l'événementielle fera une mise à jour d'un sous-ensemble de neurones (connectés) pour un sous-ensemble des pas de temps possibles.

Mais lorsque la précision est augmentée, la répartition des PA est donc généralement plus diffuse sur la grille temporelle. Dans l'exemple précédent, au lieu de 4 pas de temps occupés, l'augmentation de précision les répartit sur 6 pas de temps. Ainsi, dans cet exemple, en augmentant la précision on peut s'attendre à ce que la simulation événementielle explore 40% des pas de temps explorés en clock-driven mais il y en aura en fait 60% d'explorés. Ainsi, malgré ses atouts, la méthode événementielle n'est pas réellement indépendante de la précision temporelle choisie.

Des algorithmes événementiels séquentiels, adaptés aux réseaux de neurones impulsionnels, ont été proposés (Mattia & Del Giudice 2000, Claverol et al. 1999; 2002,

Ros et al. 2006, Carrillo et al. 2007). Mais peu de simulateurs, disponibles pour la communauté scientifique, exploitent actuellement ce type de simulation (Diesmann & Gewaltig 2002, Rochel & Martinez 2003, Muresan & Ignat 2004) et tous sont récents. Pourtant, la stratégie événementielle est de plus en plus utilisée pour les simulations d'inspiration biologique basées sur des modèles impulsionnels, car elle est bien adaptée aux caractéristiques des réseaux modélisés.

2.4.4 Types de neurones exploitables en événementiel

L'atout majeur de la méthode événementielle repose sur la fait que des mises à jour sont faites pour des neurones uniquement lors de la réception d'un événement. Cet avantage est aussi l'inconvénient majeur de la méthode événementielle.

En effet, c'est à réception d'un événement, et uniquement à ce moment, qu'un neurone doit déterminer quelle sera la date d'émission de son prochain PA. Effectuer cette prédiction est un problème qui peut s'avérer complexe. La difficulté dépend avant tout du modèle de neurone et des réponses synaptiques modélisées.

Implémentation simple

L'implémentation d'un modèle de neurone dans une simulation événementielle est simple lorsque les modèles de neurones utilisés sont définis par une seule équation différentielle linéaire entre deux réceptions de PA comme les neurones LIF et gIF (voir sections 1.3, 1.6 et 1.7). Le modèle SRM, qui est exprimé sous forme d'intégration, peut également être implémenté de manière événementielle si la fonction utilisée pour décrire la dynamique de membrane ($\eta(t-\hat{t})$ voir section 1.6) est une équation différentielle linéaire du premier ordre (ou une fonction plus simple).

Les événements de la simulation sont des PA et à chaque réception d'un PA, un neurone met à jour son potentiel de membrane. Dans ces cas, la seule la fuite du potentiel de membrane influence le neurone pour le ramener progressivement à son potentiel de repos, en l'absence de stimulations. Alors le calcul de la date du prochain PA est simple :

- Les solutions exactes des équations différentielles peuvent être calculées directement.
- Seul l'impact synaptique dû au PA reçu déterminera le franchissement ou non du seuil et donc l'émission d'un nouveau PA (voir section 2.4.5).

Ces modèles de neurones sont donc particulièrement adaptés à la simulation événementielle. Ils sont de plus en plus utilisés et développés dans les simulations de réseaux de neurones impulsionnels (Claverol et al. 2002, Marian et al. 2002, Rudolph & Destexhe 2006, Brette 2006; 2007).

Implémentation possible

Pour des modèles de neurones dont les dynamiques sont décrites par plus d'une équation différentielle, comme les modèle d'Izhikevich, les modèles QIF-EIF ou le modèle de Hodkin-Huxley, l'implémentation des modèles dans une simulation événementielle devient plus difficile. Les équations différentielles utilisées ne permettent

pas le calcul de solutions exactes, et ne permettent donc pas de déterminer avec précision la date de la prochaine émission de PA. De plus le PA n'est pas représenté explicitement dans ces modèles, ils calculent explicitement les variations du potentiel de membrane, qui dépend de la dynamique couplée de plusieurs variables. On peut s'aider de la pente de la courbe du potentiel de membrane pour déterminer si un PA va être généré ou s'il s'agit d'une variation sous le seuil. Par exemple, des modèles tels que le modèle de Hodking et Huxley (Lobb et al. 2005) ou le modèle QIF (Tonnelier et al. 2007) ont été développés en événementiel et des travaux sont en cours pour le passage en événementiel du modèle d'Izhikevich (Izhikevich 2003).

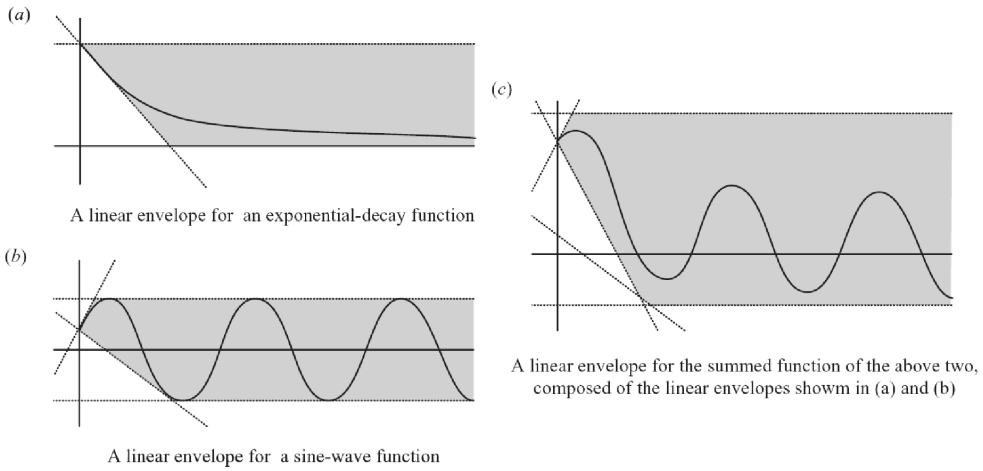

FIG. 2.10 – *Exemple d'approximation linéaire d'une fonction exponentielle décroissante (a), d'une fonction sinusoïdale (b) et d'une fonction représentant la somme de ces dernières. D'après (Makino 2003).*

Des événements « intermédiaires », ne correspondant pas à des PA, sont utilisés dans ce cas afin d'effectuer des mises à jour des variables des neurones plus fréquemment qu'à la seule réception d'un PA (par exemple pour mettre à jour l'état d'un neurone après une durée correspondant à la durée de génération d'un PA plus la durée de la période réfractaire (Lobb et al. 2005)).

En effet, dans une simulation événementielle, la date du prochain PA émis par un neurone peut être recalculée à chaque réception d'événement. Compte tenu de la nature du modèle de neurone, le potentiel de membrane n'est pas nécessairement à sa valeur maximum à la date de réception d'un événement (par exemple si la dynamique est décrite par une fonction sinusoïdale telle que celle de la figure 2.10). On peut appeler ceci une émission différée, ou *delayed firing*. Il faut donc prévoir si le seuil sera franchi avant que le maximum soit atteint. Pour résoudre ce *Delayed Firing Problem*, Makino (2003) propose une méthode de partitionnement incrémental du potentiel de membrane, chaque partition contenant, au plus, un franchissement du seuil. Ces partitions sont calculées en se basant sur des approximations linéaires des fonctions étudiées. Par exemple, la figure 2.10 illustre le type d'approximations faites dans la méthode développée par Makino. Cette méthode permet alors de manipuler des enveloppes linéaires de fonctions non-linéaires complexes.

La nécessité de prédire les émissions de PA à venir n'est pas uniquement induite par le modèle de neurone utilisé mais également par les types de réponses synaptiques envisagées.

2.4.5 Types de réponses synaptiques exploitables en événementiel

On peut distinguer deux catégories parmi les types de réponses synaptiques présentés dans la section 1.7.

Réponses instantanées

Tout d'abord les réponses synaptiques dites instantanées sont celles dont le maximum d'amplitude de la réponse est atteint à la date d'impact du PA sur la synapse (générant un PPS de type impulsion instantanée (figure 1.14)). Malgré la simplicité apparente de ces types de réponses, le cas c de la figure 1.14 est une bonne approximation des réponses de modèles biophysiques beaucoup plus précis (Rudolph & Destexhe 2006). Ces types de réponses sont très utilisés, particulièrement dans les simulations événementielles car elles permettent d'éviter le problème de la prédiction d'émission différée.

L'utilisation de modèles de neurones tels que les LIF ou gIF, associés à des types de réponses synaptiques instantanées, permet d'effectuer des simulations événementielles sans la nécessité de prédiction d'émission. Cela permet d'alléger le coût computationnel de la simulation et donc, à puissance égale, d'augmenter la taille des réseaux simulés, ou le nombre de synapses. En contrepartie, les approximations ont un impact sur les comportements neuronaux et synaptiques modélisés. Cet impact est aujourd'hui considéré comme minime à l'échelle d'un réseau et sur la durée d'une simulation complète (plusieurs centaines de millisecondes), en partie car le temps que met la réponse synaptique pour atteindre son maximum est en moyenne constant, et de l'ordre de la milliseconde au niveau biologique.

Réponses non instantanées

L'utilisation de réponses synaptiques non instantanées permet de reproduire plus fidèlement les observations biologiques. Ces réponses synaptiques peuvent être de type fonctions α ou double exponentielle (voir section 1.7 et figure 1.15). Le *Delayed Firing Problem* (Makino 2003) se pose alors à nouveau puisque l'amplitude maximum de la réponse synaptique n'est pas atteint au moment de l'impact. Des méthodes de prédiction peuvent alors être utilisées dans ce cas afin que des réponses synaptiques « non-instantanées » puissent être utilisées dans les simulations événementielles.

Le fait de procéder par prédiction des émissions de PA implique que certaines émissions de PA prévues initialement peuvent être décalées ou annulées. Les simulateurs événementiels doivent donc prévoir ce type de situations. Une solution consiste à considérer les événements contenus dans la file comme provisoires jusqu'à ce qu'ils soient réellement traités. Dans l'absolu, seul l'événement en tête de file peut être considéré comme sûr. Cet événement peut en effet avoir une incidence sur les événements suivants dans la file. En pratique on peut étendre cette propriété à un plus grand nombre d'événements, comme par exemple lorsque l'on tient compte des délais de transmission comme nous allons le voir.

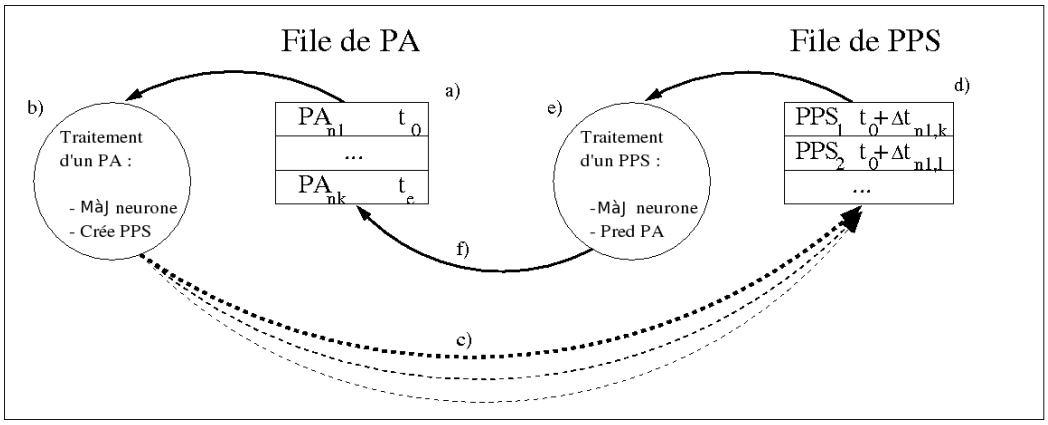

FIG. 2.11 – *La file d'événements est séparée en deux files (dans un souci de clarté). La première, la File de PA, stocke les événements PA en attente d'émission et la seconde, la File de PPS, stocke les événements PPS en attente de réception. A t_0 le neurone n_1 a émis un PA (a)). Cet événement se trouve en tête de la File de PA et est le prochain événement à traiter. Le traitement du PA_{n1} (b)) engendre la création de plusieurs événements PPS insérés dans la File de PPS (c)), ainsi que la mise à jour des variables d'état du neurone n_1. Lorsqu'il sera en tête de file (d)), le PPS_1 généré sera traité par le neurone n_k à la date $t = t_0 + \Delta t_{n1,k}$. Le délai $\Delta t_{n1,k}$ est spécifique de la connexion entre les neurones n_1 et n_k. Le traitement du PPS_1 consiste alors à mettre à jour les variables d'état du neurone n_k à réception du PPS (e)). Cette mise à jour peut éventuellement mener à la prédiction d'émission d'un PA à $t = t_e$, qui sera inséré dans la File de PA (f)).*

2.4.6 Prise en compte des délais de transmission

Nous avons jusqu'ici considéré la transmission des PA entre les neurones connectés comme instantanée. Dans la réalité biologique, la transmission du PA d'un neurone pré-synaptique à un neurone post-synaptique prend un certain temps. Selon les modélisations, le délai considéré peut être le délai pré-synaptique (dendritique), synaptique ou post-synaptique (axonal). En général, un seul délai est utilisé pour modéliser l'ensemble des temps de transmission d'un neurone à l'autre (voir section 1.8).

La gestion des délais implique de diviser en deux étapes l'émission d'un PA (voir figure 2.11). Dans un premier temps, lorsqu'un PA prédit est effectivement émis, on peut créer des événements PPS destinés aux neurones atteints par ce PA et mettre à jour les variables du neurone émetteur (période réfractaire, retour au potentiel de repos, prédiction du prochain PA...). Dans un second temps, lorsqu'un événement PPS se trouve en tête de file, on met à jour les variables de celui-ci (potentiel de membrane, PA prédit...). Les événements PPS peuvent également être gérés dans une seconde file spécifique. Dans ce cas, récupérer le prochain événement à traiter consistera à comparer les dates des événements en tête de file, dans les deux files, avant de choisir lequel des deux doit être traité en premier.

Au premier abord, il peut sembler que la gestion des délais complique la simulation, mais en définitive, cela permet d'optimiser les traitements. En effet, de prime abord, générer des événements PPS augmente sensiblement le nombre d'événements à insérer (et donc à trier) dans la (les) file(s). Ceci a un impact sur le coût computationnel. D'un autre côté, il est possible d'adapter les structures de données gérant la file de priorité afin de tenir compte des délais et d'optimiser le traitement.

Dans le cadre de la prédiction d'émission notamment, en utilisant judicieusement

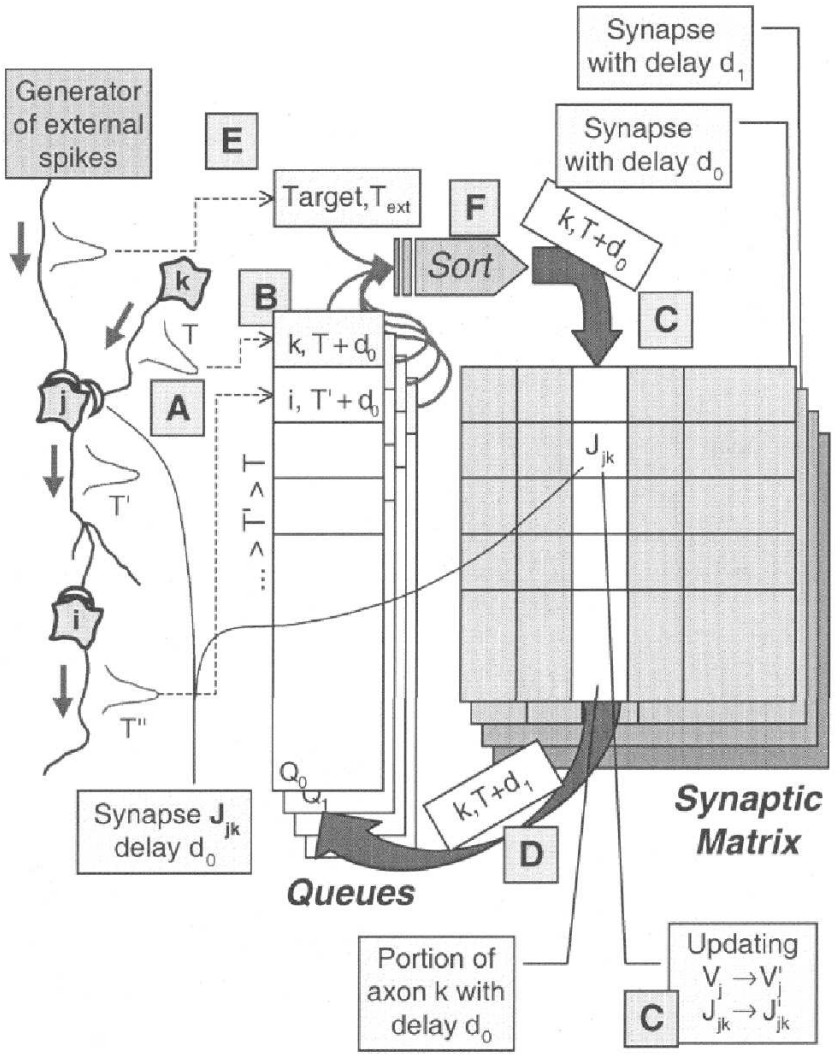

FIG. 2.12 – *Schéma présentant le fonctionnement de la gestion des événements dans (Mattia & Del Giudice 2000). Ils sont placés dans des tableaux (B) spécifiques du délai. Dans leurs travaux, les synapses sont également classées dans des matrices spécifiques du délai (C).*

les délais, on peut « certifier » certains événements prédits. En effet, si δ_{min} est le délai minimum implémenté dans la topologie du réseau, un événement e_1 émis à une date t_1 n'aura pas d'impact avant la date $t_1 + \delta_{min}$. Donc un événement e_2 dont la date t_2 est telle que $t_1 \leq t_2 < t_1 + \delta_{min}$ ne sera en aucun cas influencé (annulé, avancé ou retardé) par le traitement de e_1. Ceci implique que tous les événements dont les dates sont comprises dans cet intervalle peuvent être certifiés. Nous parlerons plus en détail de cet avantage dans le cadre de simulations distribuées au chapitre 3.

Dans tous les cas, quel que soit le mode de simulation (dirigée par une horloge ou événementielle), on peut optimiser les files de priorité grâce aux délais. Connaissant le délai minimum δ_{min}, le délai maximum δ_{max} et la précision temporelle de la simulation, on peut alors définir la file comme un ensemble de tableaux en fonction du délai. Chaque tableau stocke les événements à traiter pour un délai donné δ. Mat-

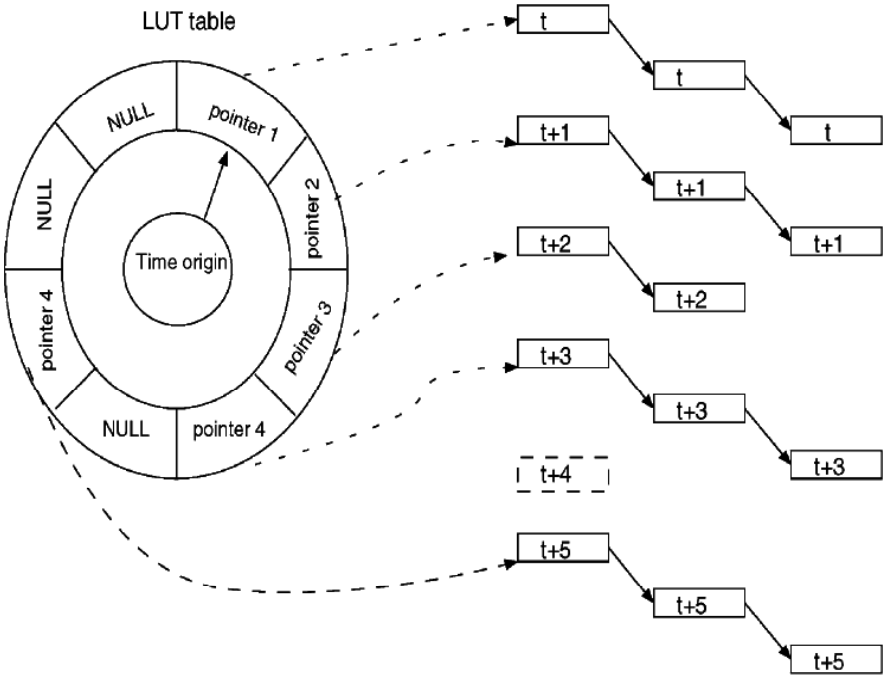

FIG. 2.13 – *Schéma présentant le fonctionnement de la gestion des événements dans (Claverol et al. 2002). Les événements sont placés dans une file circulaire. Chaque élément de la file pointe vers un tableau d'événements spécifique d'un délai donné. La taille de la file correspond au délai maximum. Le terme LUT (Look Up Table) est utilisé ici pour décrire la structure en file circulaire.*

tia & Del Giudice (2000) utilisent une méthode de ce type (voir figure 2.12). Dans le même état d'esprit, une structure de donnée de type file circulaire (*calendar queues*) est utilisée dans (Claverol et al. 2002) (voir figure 2.13). Ce type de méthode permet de réduire les coûts d'insertion dans la file à une valeur constante. Cependant, cela nécessite une utilisation supplémentaire de mémoire afin de stocker la file circulaire qui contiendra $T = \frac{\delta_{max} - \delta_{min}}{dt}$ éléments. Malgré tout, la surcharge est limitée puisque seuls les pointeurs de la file adressant des tableaux contenant des événements occuperont de la place en mémoire (les autres pointeurs seront nuls). Une méthode similaire, efficace et sans coût computationnel supplémentaire est proposée au chapitre 4.

2.5 Conclusion

Toutes les simulations de réseaux de neurones impulsionnels nécessitent la mise en place d'un cadre stable et paramétrable pour fournir une implémentation des modèles, stimuler et étudier précisément les caractéristiques d'intérêt. Ce fait motive depuis longtemps la réalisation de simulateurs. La réalisation d'un simulateur à la fois suffisamment efficace et suffisamment générique pour être en mesure d'effectuer facilement les simulations, dans des temps raisonnables, quels que soient les exigences des expériences menées, semble impossible, notamment car l'outil réalisé serait alors inutilement lourd dans tous les cas d'utilisation (en terme de charge computationnelle). La solution la meilleure est alors de choisir pour un problème donné le simulateur le mieux adapté. En effet, les échelles de modélisation choisies par exemple

(à un niveau populationnel, à un niveau cellulaire, à l'aide de compartiments, à un niveau ionique ...) peuvent nécessiter une gestion très différente des simulations.

Malgré tout, il est indispensable de travailler à la généricité des simulateurs réalisés afin que leurs évolutions soient le moins restrictives possible, afin que les communautés scientifiques ne soient plus contraintes d'utiliser de multiples simulateurs différents en fonction du problème étudié, ainsi que pour fournir un maximum de degrés de liberté aux utilisateurs. En effet, pour un modélisateur, la recherche d'un simulateur sera motivée par ces critères. Et malgré le nombre non négligeable de simulateurs disponibles, il est encore difficile de trouver un outil à la fois suffisamment facile d'utilisation tout en étant suffisamment paramétrable.

L'utilisation de méthodes événementielles, *event-driven*, pour la simulation de réseaux de neurones impulsionnels est récente. Cependant, les approches événementielles sont de plus en plus utilisées du fait de leur adéquation avec les principes de fonctionnement des neurones impulsionnels et pour leur gain computationnel par rapport aux simulations dirigées par une horloge, *clock-driven*, dans des conditions de fonctionnement inspirées de la neurobiologie.

Malgré les atouts de la méthode événementielle, la précision des modèles de neurones et de synapses, la gestion des délais, les méthodes d'apprentissage, la taille des réseaux et leurs dynamiques sont autant de facteurs « gourmands » en termes de charge computationnelle. Les machines de bureau actuelles évoluent mais en contrepartie, étant donné que la puissance des machines nécessaires à ces simulations ne peut augmenter au même rythme que la recherche, on peut faire appel à des simulations basées sur des supports parallèles.

Simulations distribuées 3

Seuls de très rares projets ont les moyens d'investir dans des machines suffisamment puissantes pour réaliser des simulations précises et à grandes échelles (par exemple IBM - « Blue Gene » (Markram 2006)). Mais les machines de bureau sont de plus en plus souvent équipées de plusieurs processeurs. Elles sont très généralement disponibles en réseaux dans les entreprises ou universités. Pour exploiter l'intérêt de la multiplication des processeurs, des méthodes de simulation de réseaux de neurones impulsionnels exploitant plusieurs processeurs doivent être implémentées.

L'utilisation d'un simulateur pour exploiter un support matériel parallèle permet au modélisateur de réseaux de neurones de ne pas tenir compte du fait que la simulation est réalisée sur plusieurs machines. Ce doit tout au moins être l'un des objectifs principaux dans la réalisation de tels simulateurs. À l'inverse, un modélisateur souhaitant utiliser plusieurs processeurs ou plusieurs machines pour accélérer les simulations est quasi-obligé de faire appel à des simulateurs.

Dans ce chapitre, les enjeux de l'utilisation de ressources matérielles distribuées pour les simulations de réseaux de neurones impulsionnels sont abordés (section 3.1), avant de définir les besoins des utilisateurs et les bases conceptuelles des simulations parallèles et distribuées (section 3.2). Différentes méthodes de gestion du temps virtuel dans le cadre de simulations distribuées seront étudiées afin de déterminer les optimisations possibles pour les réseaux de neurones impulsionnels (section 3.3). Enfin, nous aborderons l'utilisation de multiples *threads* dans les simulateurs (section 3.4) et proposerons un méthode pour optimiser le recouvrement des temps de calcul et de communication dans ces simulations distribuées.

3.1 Des simulateurs utilisant le support parallèle

La mise en place d'une « couche logicielle » permettant à tout modélisateur de réseaux de neurones impulsionnels d'exploiter un support matériel parallèle, sans nécessité de compétences dans ce domaine, représente un enjeu considérable pour la communauté. La perspective de l'utilisation d'un nombre de processeurs et d'une quantité de mémoire suffisamment élevés pour cumuler

- précision des modèles,
- dynamiques d'apprentissage,
- simulation de réseaux de grande taille,

donne un intérêt supplémentaire aux simulations exploitant un support parallèle.

À l'heure actuelle, parmi les simulateurs destinés aux réseaux de neurones impulsionnels, il en existe très peu qui utilisent le parallélisme (Morrison et al. 2005). Quelques simulateurs déjà existant sous forme séquentielle ont récemment été développés dans des versions parallèles tels que NEURON (Migliore et al. 2006) ou GENESIS (Lobb et al. 2005).

Si l'on s'intéresse aux simulateurs utilisant une méthode événementielle, seul le simulateur NEST exploite le principe événementiel couplé à un support parallèle (Eppler et al. 2006, Plesser et al. 2007) qui utilise un système hybride *clock-driven/event-driven*, (voir figure 2.1).

Les premiers Grassmann & Anlauf (1998) avaient proposé un cadre de simulation distribué et entièrement événementiel mais aucun simulateur exploitant cette méthode n'a, à ce jour, été mis à la disposition de la communauté scientifique par les auteurs.

3.2 Environnement de simulation

Un simulateur exploitant un support parallèle repose avant tout sur les caractéristiques et les rôles des différentes ressources matérielles utilisées : processeurs, mémoire, réseau de communication. Mais il repose également sur un environnement logique de communication entre les différents processus de la simulation.

3.2.1 Architectures matérielles

Exploiter un support parallèle pour exécuter des simulations de réseaux de neurones impulsionnels implique que lesdits supports soient accessibles à la communauté effectuant ces simulations, de manière aisée, peu coûteuse et efficace. L'architecture MIMD *(Multiple Instructions, Multiple Data flows)* (voir Flynn & Rudd 1996) est la plus répandue actuellement. Ce type d'architecture matérielle est également parmi les plus abordables du fait d'une demande croissante du public et des industriels. La miniaturisation et l'augmentation de la cadence de traitement ayant atteint leurs limites :

- Le passage aux ordinateurs quantiques n'est pas pour demain, augmenter la puissance computationnelle passe donc par une multiplication des coeurs.
- Les processeurs cadencés à des vitesses au-delà de 2 à 3 GHz sont trop gourmands en énergie nécessaire au simple refroidissement du processeur. Il devient plus « rentable » en termes computationnels au moins, de diminuer la cadence et d'utiliser l'énergie du refroidissement pour alimenter un second cœur.

Des architectures matérielles dédiées telles que les *FPGA Field Programmable Gate Array* sont parfois utilisées (Bumble & Coraor 1998, Hellmich & Klar 2004) mais restent très contraignantes notamment en termes de modèles et de taille des réseaux simulables.

Ainsi, l'intérêt des applications exploitant des architectures MIMD est grandissant depuis quelques années. C'est donc ce type d'architectures qu'il est intéressant d'exploiter dans un simulateur destiné à être utilisé par le plus grand nombre de modélisateurs et non uniquement par quelques spécialistes du parallélisme.

Plus précisément, la gestion de la mémoire dans ce type d'architecture distingue deux classes :

1. MIMD-DM : à mémoire distribuée (*Distributed Memory*)
2. MIMD-SM : à mémoire partagée (*Shared Memory*)

La forme la plus répandue sous laquelle on utilise des architectures MIMD-DM est le réseau de stations de travail puisque chaque machine possède sa propre mémoire physique. C'est donc plus particulièrement à ce type d'architecture matérielle que nous nous intéressons dans la suite de ce travail de thèse, afin que le travail réalisé soit utilisable par la majorité de la communauté désireuse d'exploiter un support parallèle pour des simulations de réseaux de neurones.

Par ailleurs, il est possible de faire usage, sur chaque machine, de plusieurs processus (ou *thread*) distincts. Il est également possible que les machines utilisées possèdent chacune plusieurs processeurs (ou cœurs), ce qui est de plus en plus le cas (doubles, quadruples ou même octuples cœurs) compte tenu des évolutions de la technologie. Si chaque processeur de la machine est exploité en tant que tel dans le système distribué modélisé, cela revient à réaliser un traitement couplant mémoire partagée et mémoire distribuée. Certains messages seront envoyés sur le réseau vers d'autres processeurs, et d'autres seront envoyés via la mémoire locale.

Dans les faits, dans les programmes implémentant des systèmes distribués, on trouve généralement un unique programme exécutable qui est lancé sur chaque processeur. Les processeurs exécutent chacun une partie du code en fonction de leur identifiant (un numéro la plupart du temps). Les différentes instances du programme communiquent entre elles par échanges de messages. Ce mode utilisant un même programme qui exécute des données différentes sur chaque processeur est nommé *SPMD : Single Progam Multiple Data*. Le mode d'utilisation de la mémoire, partagée ou distribuée, sera désigné par un mode *SPMD-SM* ou *SPMD-DM*. Pour faciliter la programmation de simulations distribuées, une norme définissant des protocoles d'échanges de messages existe depuis quelques années : la norme MPI *Message Passing Interface*. Il existe plusieurs implémentations de cette norme dont MPICH, LAM/MPI et MVICH qui ont des performances similaires (Ong & Farell 2000). Ces implémentations permettent d'accéder aux fonctionalités définies par la norme MPI dans les langages de programmation C, C++ ou Fortran par exemple.

3.2.2 Influence des topologies réseau

Dans l'environnement relatif à des simulations distribuées, le réseau physique de communication entre les différents processeurs (ou groupes processeurs/mémoire) a également sa place. Les topologies de réseaux utilisées pour le calcul parallèle visent à optimiser les temps de communication entre processeurs. On peut citer quelques unes des plus communes (voir figure 3.1) :

- topologie en « bus » : les machines sont connectées à un seul et même cable central ou via un *hub*.
- topologie en « anneau » : les machines sont connectées de façon circulaire. Chaque machine est connectée à deux machines.
- topologie en « étoile » : les machines sont toutes reliées à un switch central. C'est ce qui est le plus courant dans les réseaux locaux.

L'utilisation de ces topologies dépend des architectures logiques des systèmes distribués exécutés sur ces architectures physiques. Par exemple un système distribué de type client/serveur tirera avantage d'une topologie physique en étoile.

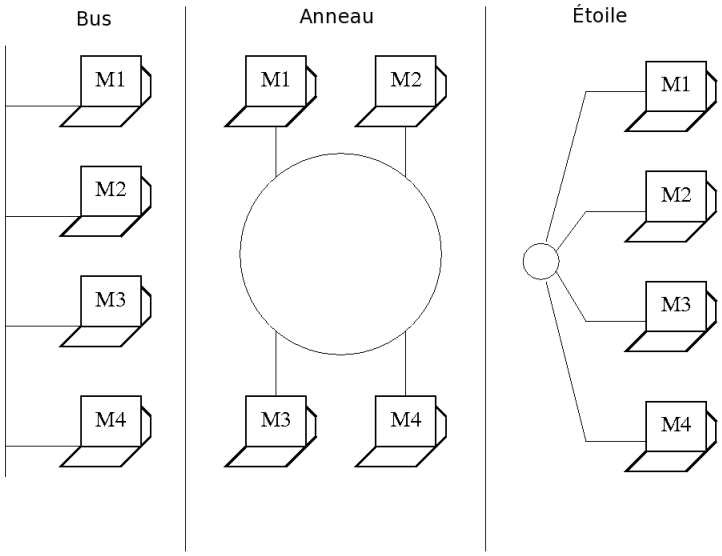

FIG. 3.1 – *Quelques-unes des topologies de réseau utilisées couramment.*

Selon le système exécuté, des topologies plus complexes (telles que des topologies en arbre, en grille, en tore, grille torique ou hypercube par exemple) ainsi que des topologies spécifiques du système (un *pipeline* par exemple) peuvent être utilisées. Les topologies les plus courantes sont les topologies en étoile (avec un *switch* central), qui sont présentes dans beaucoup de réseaux locaux (LAN) (entreprises, universités, laboratoires ...). On peut considère souvent ces topologies de résau comme des topologies entièrement connectées (tous à tous) où le débit réseau est limité au débit du switch. Une topologie *r*éellement en étoile est constituée d'une machine centrale connectée aux autres machines. Cette machine fait alors partie intégrante du traitement. Pour les machines dédiées, les topologies en grille torique ($2D$ ou $3D$) sont parmi les plus utilisées.

La norme MPI ajoute une couche logicielle au réseau de communications. Un réseau logique est défini, contenant des nœuds (associés à un ou plusieurs processeurs physiques). Entre ces nœuds, des canaux de communication logiques sont créés. Pour optimiser les traitements, MPI permet d'utiliser des topologies logiques de communications basées sur les topologies physiques classiques.

3.2.3 Processus logiques

Les processeurs physiques peuvent être considérés comme des processus de la simulation. Cependant, plusieurs processeurs physiques peuvent également ne représenter qu'un seul processus de la simulation et, inversement, un même processeur physique peut héberger plusieurs processus de la simulation.

Pour bien distinguer les processeurs disponibles sur l'architecture matérielle des processus communicants par échanges de messages dans la simulation, nous définirons un Processus Logique, comme un processus au sens du système distribué modélisé (Chandy & Misra 1979).

Le système consiste donc en un ensemble de LP (*Logical Process*). La figure 3.2 illustre différents niveaux d'observation de l'architecture utilisée dans une simula-

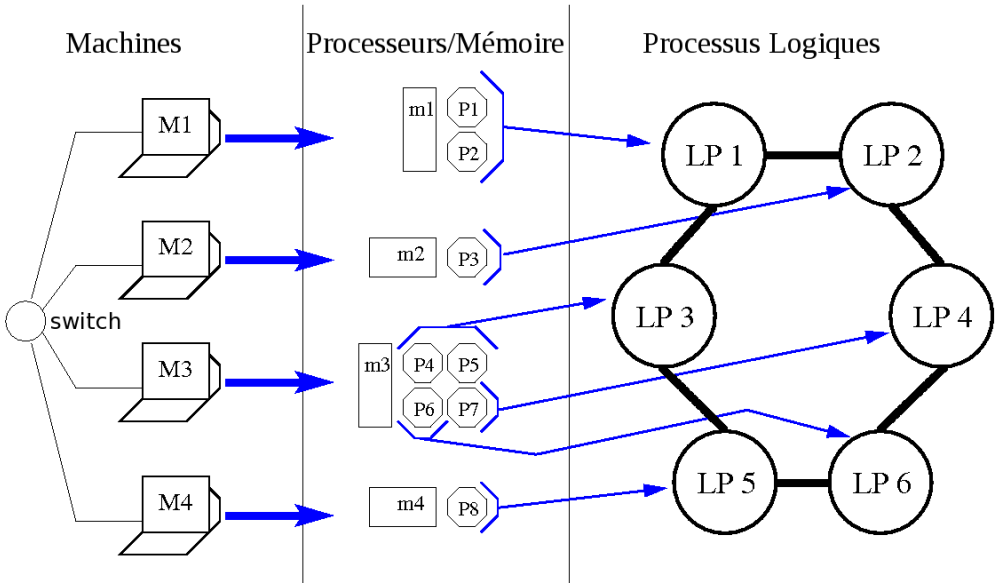

FIG. 3.2 – *Différents niveaux d'observation dans une simulation distribuée. La simulation d'un système distribué utilise plusieurs machines (M1 à M4). Chacune de ces machines possède sa propre mémoire (m1 à m4) et un certain nombre de processeurs (P1 à P8). Les processeurs d'une même machine communiquent entre eux via leur mémoire locale. Les processeurs situés sur différentes machines communiquent entre eux par le biais du réseau (le switch faisant le lien entre les différentes machines). Enfin le programme exécuté pour la simulation est exécuté en plusieurs instances, ou Processus Logiques (LP1 à LP6) qui peuvent échanger des messages sur un réseau de communications logique. Ce dernier utilise le réseau physique pour délivrer les messages aux processeurs.*

tion distribuée. On utilise dans cet exemple 4 machines, chacune possède de 1 à 4 processeurs. Les Processus Logiques définis dans cet exemple sont au nombre de 6. Les *LP*1 et *LP*3 utilisent deux processeurs chacun, les autres LP utilisent un seul processeur chacun. Dans cette simulation, tous les LP peuvent communiquer directement les uns avec les autres.

3.2.4 Simulations parallèles et distribuées à événements discrets

Les supports parallèles peuvent être exploités pour exécuter des simulations à événements discrets. Ces simulations parallèles et événementielles sont regroupées sous la dénomination PDES pour *Parallel Discrete Event Simulation* (Simulations Parallèles à événements Discrets).

Ferscha (1996) propose une excellente revue des concepts de la littérature pour la gestion de simulations parallèles (PDES) et distribuées (DDES) à événements discrets.

Dans le cadre des réseaux de neurones impulsionnels, nous avons présenté au chapitre 2 une stratégie de simulation événementielle. En couplant cette stratégie avec les concepts de PDES et DDES, il est possible de développer des méthodes de simulation de réseaux de neurones impulsionnels par stratégie événementielle en exploitant un support matériel distribué.

La plupart des méthodes de simulation parallèle de réseaux de neurones existantes (Jahnke et al. 1997, Mohraz et al. 1997, Schaefer et al. 2002) répartissant les calculs des phases parallélisables d'un programme séquentiel sur plusieurs proces-

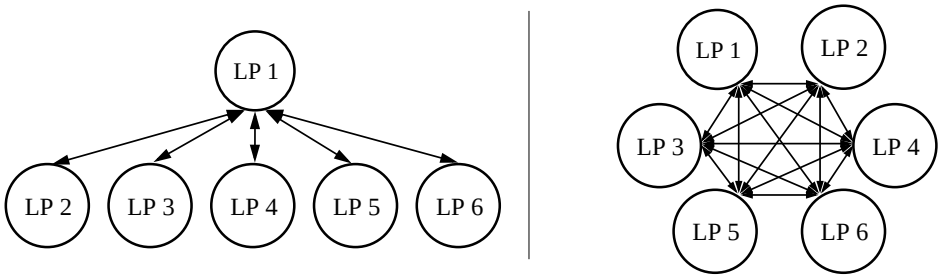

FIG. 3.3 – *Représentation des communications entre processus logiques dans une simulation PDES à gauche ou DDES à droite. À gauche, une approche dite parallèle : les communications entre les différents LP reposent sur le LP_1 contrôleur central. À droite, une approche dite distribuée : les communications reposent sur une collaboration entre tous les LP, sans contrôleur.*

seurs qui se synchronisent ensuite pour la phase suivante (voir Seiffert 2004, pour une revue). Ces méthodes, désignées par le terme PDES, sont désormais majoritairement exécutées sur des architectures physiques de type MIMD : Un processeur central fournit le travail à effectuer et récupère les données par échanges de messages (la même instruction est exécutée par tous de , .

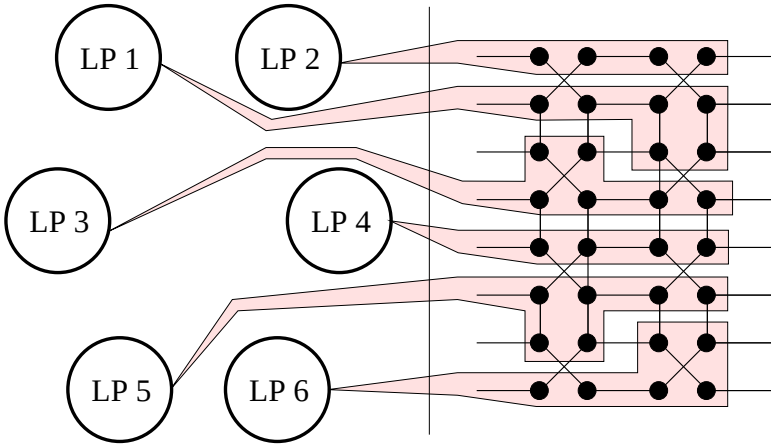

FIG. 3.4 – *La dynamique du réseau de neurones présenté à droite (les connexions aux extrémités représentent les entrées/sorties) va être simulée (selon les modèles de neurones, de synapses, de stimulations, etc) de manière distribuée. Chaque LP prend en charge le traitement des événements relatifs à une sous-partie (ombrée) du réseau de neurones.*

Bien que cette approche parallèle soit envisageable pour des simulations événementielles, une approche « simulation distribuée » est également possible (Misra 1986, Marin 1998). Elle consiste à répartir les traitements relatifs à chaque sous-partie du réseau de neurones sur chaque LP (Grassmann & Anlauf 1998). Les LP échangent entre eux les données de manière distribuée durant la simulation (voir figure 3.3). Les architectures MIMD conviennent particulièrement bien à ce type d'approches. La figure 3.4 fournit un exemple de réseau de neurones réparti sur les six LP de l'exemple (de la figure 3.2).

Chaque LP effectue une simulation événementielle séquentielle, basée sur les événements qu'il reçoit des autres LP, et sur les événements qu'il génère lui-même du fait de l'activité du sous-réseau dont il a la charge. L'idéal est que chaque LP ait connais-

sance des connexions que les neurones de son sous-réseau possèdent avec ceux des sous-réseaux des autres LP.

De plus, si les LP n'ont besoin de définir en mémoire que ces connexions, les connexions internes et la portion de neurones dont ils ont la charge, alors augmenter le nombre de machines revient à augmenter la taille des réseaux que l'on peut définir en mémoire. Dans une simulation événementielle où les événements sont des potentiels d'action (PA), nous avons vu que les événements sont datés (section 2.4.1). Afin que chaque LP effectue séquentiellement les traitements des événements en respectant l'ordre temporel (au sens du temps virtuel) de ces événements, il est nécessaire d'adopter une méthode de gestion globale du temps virtuel.

3.3 Gestion distribuée et temps virtuel

Chaque LP n'a connaissance que d'un sous-ensemble des événements traités dans la simulation. Cependant, du point de vue global de la simulation, l'ordre temporel des événements doit rester cohérent avec une simulation séquentielle équivalente. Les LP devront donc contrôler que la date du prochain événement de leur file est bien cohérente avec le temps global. Deux types de gestion du temps virtuel global sont possibles : synchrone ou asynchrone.

3.3.1 Simulation synchrone

La première consiste à choisir un LP_c en tant que contrôleur, chacun des autres attendra un message de LP_c contenant l'information sur le temps virtuel global lui permettant de traiter ses prochains événements.

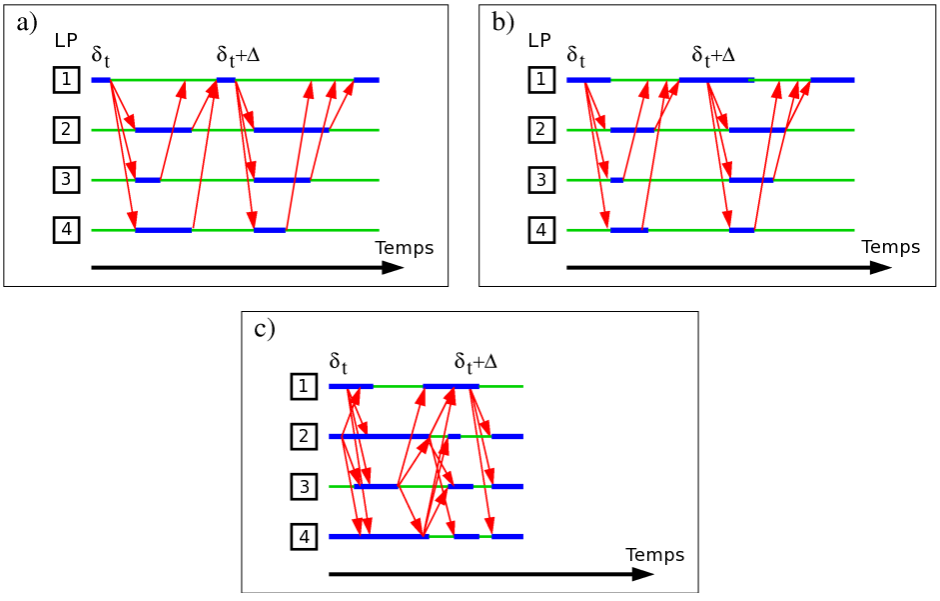

Fig. 3.5 – *Exemple de déroulement d'une simulation synchrone (a,b) et asynchrone (c). Trait épais (en bleu) : période de travail, trait fin (en vert) : période d'attente, flèches (en rouge) : communications. Le détail des figures est fourni dans le texte.*

Les simulations synchrones impliquent de fait une synchronisation explicite des nœuds (ou LP) entre chacune des « phases » de traitement. Cette synchronisation peut être matérialisée par l'utilisation de « barrières » de synchronisation. Une barrière de synchronisation consiste, pour chaque LP, en une instruction à une position précise du programme (par exemple après l'envoi d'un message), à attendre que tous les autres LP de la simulation aient également atteint ce point du programme.

Un nœud LP_c central peut également être utilisé comme contrôleur, imposant la synchronisation des autres nœuds, par exemple à intervalles temporels précis. Dans l'exemple présenté sur la figure 3.5, on suppose la simulation d'un pas de temps δ_t dans un réseau de neurones. On pose $LP_1 = LP_c$ qui joue le rôle de contrôleur et indique le temps global. Localement la date est donc toujours identique pour chaque LP, à celle du LP_c. La figure 3.5 illustre le décours temporel des activités de quatre LP dans une simulation, selon que le LP choisi comme contrôleur est dédié à cette seule tâche, cas a, ou bien effectue également une partie du traitement, cas b. Dans les deux cas, chaque LP (et donc le/les processeurs sous-jacents) passe beaucoup de temps à ne rien faire dans l'attente de la prochaine synchronisation. Le LP_1 doit récupérer les dates des prochains événements de chaque LP, pour fournir ensuite à tous les LP le nouveau temps virtuel global. Le cas c est discuté en section 3.3.3.

3.3.2 Inter-dépendance des événements

Les méthodes de simulation synchrones considèrent chaque événement indépendamment des autres. De ce fait, la progression de la simulation est soumise à des règles de synchronisation explicites. Ce type de méthodes ne peut donc pas prendre en compte le fait que les événements sont rarement indépendants les uns des autres.

Pourtant, les événements ne peuvent pas être émis tant que des événements antérieurs n'ont pas entraîné leur génération. Ainsi le traitement d'un événement dans le futur est prévisible à partir des événements qui le précèdent.

Dans le cas de simulations basées sur une progression temporelle, les événements ne sont pas reçus dans un ordre quelconque par les LP, ce qui accentue la dépendance entre les événements au-delà de la simple « précédence ». En effet, sur le réseau logique, chaque LP établit des canaux de communication entrants et sortants avec tous les autres LP (ou selon la topologie logique utilisée). Sur chaque canal, les messages émis dans un ordre chronologique arrivent dans ce même ordre (FIFO) au LP destinataire.

En particulier dans le cas des simulations de réseaux de neurones impulsionnels, l'évolution du temps virtuel local à chaque machine dans une DDES dépend de la « causalité » (Raynal & Singhal 1996, Verìssimo 1996) ainsi établie entre les événements. Cette caractéristique des simulations est exploitée pour élaborer des méthodes de synchronisation décentralisées, asynchrones.

3.3.3 Simulation asynchrone

Dans une simulation asynchrone, la gestion du temps est décentralisée, les LP communiquent entre eux par échanges de messages pour se synchroniser afin de conserver un temps virtuel global. La figure 3.5 c illustre ce type de simulation asynchrone. La progression du travail de chaque LP est communiquée aux autres, les temps d'attente sont ainsi réduits. Si les LP, au nombre de P, fonctionnent sur des processeurs

distincts, il est montré que l'accélération d'une simulation asynchrone par rapport à une simulation synchrone peut être, au mieux, de $\mathcal{O}(log(P))$ (Felderman & Kleinrock 1990). De plus les méthodes décentralisée génèrent plus d'émissions de messages pour effectuer les synchronisations nécessaires.

Malgré tout, cette borne supérieure étant obtenue sans tenir compte de la dépendance causale entre les événements (Marin 1998), le gain potentiel d'une méthode décentralisée (asynchrone) sur une méthode centralisée (synchrone) dépend donc directement des dynamiques des réseaux simulés générant les événements traités et donc les messages de synchronisation. Il est donc possible de tirer avantage de cette dépendance pour limiter le nombre de messages supplémentaires nécessaires à une gestion décentralisée.

Algorithme 4 : Algorithme de base pour un LP_i dans une DDES

```
T_i = 0
tant que ! fin de simulation faire
    pour j = o ; j < P ; ++j faire
        Calculer les messages à envoyer à T_i
        Envoyer les messages vers LP_j
    fin
    T'_i = T_i
    tant que T'_i == T_i faire
        Attendre de nouveaux messages
        À réception d'un message : mettre à jour l'état de LP_i
        Recalculer T_i = min(T_j), j, 0 < j < P
    fin
fin
```

Au début des années 80, Chandy & Misra (1979) et Bryant (1984) ont proposé des algorithmes similaires pour l'exécution de DES parallèles et distribuées. On désigne généralement ces travaux sous le nom d'algorithme CMB (Chandy-Misra-Bryant). L'algorithme de base décrivant le fonctionnement d'un LP_i est fourni par l'algorithme 4 (D'après Misra 1986). T_i représente la date de l'horloge du LP_i considéré. Les événements transmis dans les messages sont tous datés. La mise à jour de l'état du LP_i dépend du problème considéré.

Un LP ne recalcule son temps virtuel local T_i qu'à réception d'un message depuis l'un des autres LP. Donc si aucun message ne transite dans un canal de communication entre un LP_i et un LP_j, alors le temps virtuel T_i du LP_i ne pourra pas progresser au-delà du dernier temps virtuel T_j reçu. Ce cas peut arriver fréquemment, surtout dans des simulations de systèmes dont les canaux de communication ne sont pas utilisés durant toute la simulation.

Typiquement, dans un réseau de neurones, dès lors que l'on sépare le réseau en sous-réseaux répartis sur les LP, les communications entre les LP peuvent devenir hautement hétérogènes.

Un ou plusieurs LP peuvent alors être bloqués en attente de messages depuis un LP qui est lui-même en attente de l'un d'entre eux par exemple, on aura alors un « cycle ». Dans de tels cas, on parle d' « interblocage » ou *deadlock*. Dans certains cas, un interblocage peut entraîner une surcharge mémoire (ou *overflow*), des événements

s'accumulant localement sans pouvoir être envoyés vers les LP distants. Pour obtenir des algorithmes dépourvus d'interblocages, deux catégories de méthodes sont implémentables soit pour éviter tous types d'interblocage, soit pour repérer les interblocages et réparer les éventuelles erreurs générées.

Méthode de synchronisation conservative

Éviter les interblocages à tout prix est l'objectif d'une méthode conservative. Ces méthodes consistent généralement à envoyer des messages sans contenu (pas d'événements), excepté des informations sur le temps virtuel local du LP émetteur. Également nommé *nullmessage*, un exemple de protocole consiste à envoyer un *nullmessage* lorsqu'un changement du temps virtuel local a besoin d'être communiqué à un LP pour lequel aucun message n'est en attente. Dans ce cas, même sans avoir d'événements à traiter, le LP recevant le *nullmessage* peut faire progresser son temps local en fonction de celui reçu et ainsi éviter l'interblocage.

Cette méthode peut être améliorée (Su & Seitz 1988, Marin 2000b), par l'envoi de messages vides seulement sur demande, ou seulement en cas de blocage par exemple, mais ces améliorations sont souvent spécifiques au type de simulation effectuée. En contrepartie, ce type de méthode augmente le nombre de communications entre les LP.

Dans tous les cas, dans une DDES, il est nécessaire d'être en mesure de définir quand un LP_i peut traiter un événement E à une date t_E, lorsque $T_i < t_E$. Autrement dit, il faut qu'un LP puisse connaître le délai (virtuel) minimum durant lequel il est sûr de ne pas recevoir d'événements afin de pouvoir incrémenter son temps virtuel local. Pour ce faire, chaque LP peut joindre à ses messages une date dans le futur représentant la date virtuelle supposée du prochain événement à émettre. Cette fenêtre de temps (ou *lookahead*) permet aux autres LP de progresser dans leurs traitements en fonction du plus petit *lookahead* reçu. Suivant les types de simulations réalisées, le *lookahead* peut être fixe ou variable mais il est déterminant des performances dans tous les cas.

Méthode de synchronisation optimiste

Une approche différente existe pour assurer une synchronisation globale dans une simulation événementielle distribuée (Korniss et al. 2003). Cette approche, dite « optimiste » consiste non plus à éviter, mais à détecter les interblocages. Dès leur détections, il est alors nécessaire d'effectuer une récupération du système à une date précédente à laquelle aucune erreur ne s'était produite. Pour ce faire, les LP conserverons une mémoire des événements traités dans le passé. Ces méthodes optimistes sont particulièrement efficaces pour des simulations peu sujettes à interblocages, notamment celles nécessitant peu de communications. Dans ces cas, seule la détection des interblocages (dont le coût est faible en comparaison de la récupération) influe sur les temps de traitement.

Dans le cas contraire cependant, lorsque les interblocages sont fréquents, les coûts computationnels de récupération des erreurs peuvent rapidement handicaper la progression des simulations.

3.3.4 Optimisations possibles dans les réseaux de neurones impulsionnels

Dans le cas des simulations de réseaux de neurones impulsionnels, et plus particulièrement dans le cas de simulations événementielles, des événements de type potentiels d'action sont échangés entre les neurones. Les PA sont émis d'un neurone émetteur vers le neurone récepteur avec un délai δ. Ce délai représente la modélisation des délais biologiques de transmission depuis le cône d'émergence du neurone émetteur jusqu'au soma du neurone récepteur (figure 1.2). Certaines modélisations décomposent ce délai en délai axonal, délai synaptique et délai dendritique comme illustré sur la figure 3.6.

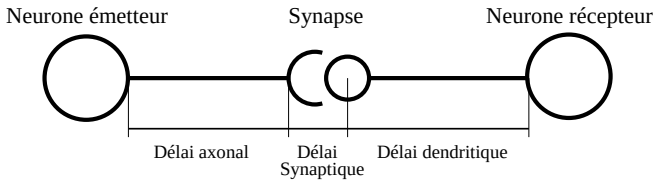

FIG. 3.6 – *Le délai de transmission d'un PA entre le neurone émetteur et le neurone récepteur peut se décomposer en délai axonal, délai synaptique et délai dendritique.*

Dans tout réseau de ce type, il existe alors un délai de transmission minimum δ_{min}, qui correspond à la transmission la plus rapide implémentée dans le réseau. Ce délai minimum représente donc le *lookahead* minimum de cette simulation.

Exploiter la topologie du réseau de neurones

On peut tirer avantage de certaines topologies de réseaux pour effectuer la répartition (ou *mapping*) du réseau de sorte que chaque LP puisse identifier des LP « précédents » et « suivants » dans le traitement. Dès lors, les *nullmessages* pourront être envoyés à un sous-ensemble des LP constituant la simulation. Un LP enverra un *nullmessage* (en cas de besoin) à ses « suivants » et la progression de son temps virtuel sera déterminée par les réceptions en provenance de ses LP « précédents ». Ce type d'optimisation est, dès lors, spécifique à la topologie du réseau de neurones modélisé mais peut être automatisé en affectant les processeurs en fonction de la topologie du réseau.

Pour simuler un réseau ayant une topologie entièrement *feedforward* (de type MLP par exemple) à trois couches comme celui de la figure 3.7, un *mapping* possible consisterait à placer sur chaque LP des neurones appartenant à une même couche, limitant ainsi le nombre de LP « précédents » et « suivants ». Dans un même temps, ce type de *mapping* limite la charge computationnelle attribuée aux LP puisque dans un MLP, il n'existe pas de connexions internes à une couche. Dans ce cas, aucun PA, et donc aucun événement ne sera émis localement. Ceci veut dire que tous les PA émis seront envoyés sur le réseau de communication. Ce type de *mapping* optimise donc la quantité de *nullmessage* circulant, mais dans le même temps cela surcharge le réseau de communications en message contenant des événements. On comprend dès lors que l'équilibre entre charge en calculs (événements calculés localement) et charge en communications (événements transmis à d'autres LP) a un rôle prépondérant dans la performance de la simulation. Il est donc possible, théoriquement, d'adopter une politique de *mapping* exploitant les propriétés topologiques des réseaux de neurones simulés afin d'équilibrer ces deux charges.

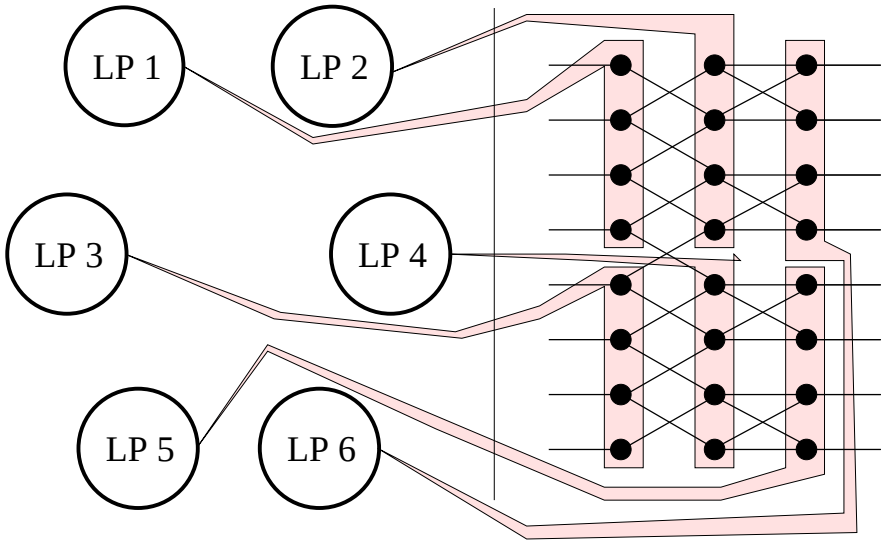

FIG. 3.7 – *Répartition d'un réseau de neurones feedforward sur six processus logiques distincts. La répartition permet à chaque LP d'avoir un seul, ou deux interlocuteurs dans les communications : LP 1 et LP 3 émettent vers deux LP, LP 2 et LP 4 reçoivent de deux LP et émettent vers un LP et les LP 5 et LP 6 reçoivent d'un LP et émettent vers un LP.*

Dans les faits, il faut ajouter dans la balance la prise en compte de la dynamique du réseau en plus de celle de sa topologie. En effet, en fonctionnement « normal », l'activité moyenne (émissions de PA) et sa répartition sur les LP est déterminante de la charge computationnelle et du nombre de communications engendrées par la simulation. En pratique, pour des réseaux de neurones impulsionnels de topologies et de dynamiques quelconques, il est extrêmement complexe de trouver une répartition idéale du réseau de neurones sur les LP. Envisager une répartition dynamique serait lourd à implémenter et mènerait à diminuer encore le gain potentiel en charge computationnelle.

Notons enfin que la topologie logique du réseau de communications entre les LP, ainsi que la topologie du réseau de communications physique sous-jacent ont également une influence importante. Korniss et al. (2003) proposent une topologie logique des LP basée sur un graphe de type « small world » et des communications aléatoires entre LP pour mettre à jour les temps virtuels locaux dans les simulations conservatives.

Exploiter les délais des connexions neuronales

De nouveau dans le cadre des réseaux de neurones impulsionnels, les délais peuvent être spécifiques d'une connexion. Sur la totalité du réseau on aura alors une distribution des délais entre deux bornes δ_{min} et δ_{max}. Ainsi, selon la distribution des délais, il est possible d'envisager une méthode de *mapping* du réseau de neurones cherchant à maximiser le délai minimum δ^i_{min} entre un neurone hébergé par un LP_i et ses cibles hébergées par chacun des autres LP. Pour évaluer la qualité d'un *mapping*, on peut par exemple prendre en compte la moyenne δ^{moy}_{min} des délais minimum sur tous les LP selon l'équation (3.1).

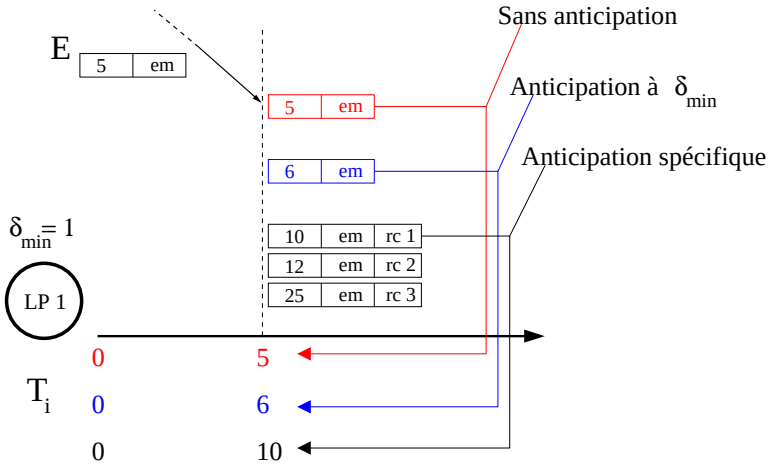

FIG. 3.8 – *Lors de la réception d'un message contenant un événement E émis à $t_e = 5$ par un émetteur em, le LP_1 peut mettre à jour son temps virtuel local $T_i = 5$ (en rouge). En tenant compte du délai minimum dans le réseau, il peut anticiper jusqu'à la date $T_i = 5 + \delta_{min} = 6$ (en bleu), mais en exploitant directement les délais à réception, il peut prédire plus loin dans le futur en fonction de la répartition spécifique des délais. Dans cet exemple, le délai spécifique minimum est $\delta_{em,rc1}$ et donc $T_i = 5 + \delta_{em,rc1} = 10$ (en noir).*

$$\delta_{min}^{moy} = \frac{1}{P} \times \sum_{i=1}^{P} \delta_{min}^{i} \tag{3.1}$$

Outre les questions relatives aux formes de réponses synaptiques dans la gestion des délais (voir section 2.4.5), en ayant accès aux délais implémentés un LP peut « utiliser » le délai de transmission pour anticiper ses futures évolutions de temps. Il peut notamment mettre à jour son temps virtuel dès la réception d'un message (voir figure 3.8).

Si δ_{min} est le délai minimum, un événement E émis à une date t_e n'aura pas d'impact avant la date $t_e + \delta_{min}$. Donc un événement E' dont la date est $t_{e'}$ telle que $t_e \leq t_{e'} < t_e + \delta_{min}$ peut être traité au même instant. Si les délais sont traités par les LP à réception d'un message, alors δ_{min} peut être mis à jour selon le délai minimum parmi toutes les connexions « activées » par l'événement reçu.

Optimiser le *mapping* en se basant sur les délais peut cependant poser un problème si le réseau est soumis à un apprentissage appliqué aux délais. Un tel apprentissage impliquerait une variation des délais dans le réseau et pourrait très rapidement influer sur la qualité du *mapping*.

Une solution pourrait être de rééquilibrer la répartition du réseau de neurones au cours de la simulation, en fonction des modifications des délais. Cependant, le surcoût engendré par les transmissions et mise à jour des structures de données reste à évaluer. Ce type de rééquilibrage dynamique de charge est également possible en fonction de l'activité du réseau de neurones. Les neurones les plus actifs durant la simulation pourraient être redistribués pour optimiser le *mapping* dynamiquement. Encore une fois, le surcoût engendré par ce type de méthodes reste à évaluer. Dans tous les cas, les apports de ces optimisations de *mapping* dynamiques restent très

dépendants de paramètres spécifiques des simulations effectuées que sont les paramètres suivants :

1. types de neurones
2. connexions
3. délais
4. poids
5. stimulations

et peuvent parfois même s'avérer plus coûteux qu'optimisants.

3.4 Multithreading

La technique du *multithreading* consiste à utiliser plusieurs processus légers (ou *threads*) pour effectuer une tâche sur un même processeur. C'est un moyen d'exploiter le parallélisme d'une application exécutée sur un seul processeur. La plupart des applications modernes exploitent ce type de parallélisme, d'autant plus les applications basées sur des événements (éditeurs, compilateurs, systèmes d'exploitation ...). Grâce au *multithreading*, un système événementiel modélisé et exécuté de manière distribuée à l'aide de LP interagissant par échanges de messages, peut être exécuté sur une machine monoprocesseur. Chaque LP est alors associé à un processus léger (*thread*) et les échanges de messages sont effectués par la mémoire associée au processeur. La plupart des implémentations de la norme MPI utilisent cette technologie pour permettre l'exécution de programmes distribués sur des stations de travail ne disposant pas forcément de plusieurs processeurs.

À l'heure actuelle, aucun simulateur de réseau de neurones ne tire avantage du *multithreading* pour exécuter des simulations autrement que via MPI (les tâches MPI sont des threads).

Cependant, une autre approche est possible. Dans notre cas, un LP joue deux rôles distincts :

– Gestion des communications avec les autres LP
– Traitement interne des événements

Le premier aspect concerne les envois/réceptions de messages et le second concerne le traitement des événements reçus des autres LP ainsi que ceux générés localement. Dans le cas des réseaux de neurones, les événements sont des PA ou des PPS (voir section 4.4). Ces deux aspects peuvent être, classiquement, effectués de manière séquentielle par les LP. Cependant, certaines parties pourraient être effectuées en parallèle. Par exemple si, à un instant donné de la simulation, pour un LP, (i) des messages sont en attente d'envoi/réception, (ii) des événements sont en attente de traitement, alors il est possible d'effectuer ces opérations en parallèle.

Aucun simulateur de la littérature, en dehors du travail effectué au cours de cette thèse et présenté au chapitre suivant, n'exploite cette caractéristique des simulations effectuées.

3.5 Conclusion

Dans ce chapitre nous avons fait état des apports potentiels d'une architecture distribuée pour les simulations de réseaux de neurones impulsionnels. La disponibilité des architectures MIMD-DM de type clusters de machines et leur adéquation au mode de fonctionnement des DDES basées sur des interactions entre LP par échanges de messages, les désignent comme candidates idéales pour la simulation distribuée de grands réseaux de neurones impulsionnels. Le gain potentiel offert par une gestion asynchrone du temps virtuel dans une DDES par rapport à une gestion synchrone nécessite néanmoins la mise en place de méthodes spécifiques pour assurer la conservation du temps virtuel au niveau global de l'ensemble des LP. Par ailleurs, il est théoriquement possible d'exploiter la topologie et la répartition des délais du réseau de neurones pour effectuer une répartition optimale du réseau sur les LP de la simulation. Le *multithreading*, inexploité jusqu'à aujourd'hui, pourrait également apporter un gain de performance supplémentaire.

Pour un modélisateur, l'utilisation de l'environnement fourni par un simulateur permet de s'affranchir d'une partie importante du développement nécessaire aux simulations des modèles qu'il définit. Compte tenu de l'évolution des coûts des supports matériels disponibles et de la complexité croissante des besoins des modélisateurs, il apparaît clairement qu'intégrer le support parallèle dans les fonctionnalités offertes par un simulateur est appelé à devenir indispensable.

D. A. M. N. E. D. Un simulateur événementiel, multithreadé et distribué.

4

Une grande partie de ce travail de thèse a consisté à élaborer un simulateur multithreadé, distribué et événementiel de réseaux de neurones impulsionnels. L'acronyme DAMNED pour *Distributed And Multithreaded Neural Event-Driven simulation framework* a été adopté, en partie pour le clin d'oeil à l'interjection anglaise du même nom. On pourrait entendre dire, par exemple : « *Damned ! ! What a surprising name !* », qui pourrait se traduire par : « Bon sang ! ! Quel nom étonnant ! ».

Les objectifs de ce travail sont tout d'abord présentés en section 4.1. Puis, la présentation du fonctionnement global de DAMNED (section 4.2) permettra d'introduire les concepts et structures développés dans le simulateur. Ensuite, pour expliciter le fonctionnement du simulateur, nous présenterons les différents éléments constitutifs de la simulation (section 4.3). Nous suivrons alors pas-à-pas le *cycle de vie* d'un événement au sein du simulateur afin de comprendre les interactions entre les éléments constitutifs (section 4.4). Nous détaillerons les méthodes de gestion distribuée du temps virtuel implémentées dans le simulateur (section 4.5). Les méthodes et outils de gestion des files d'événements seront présentés (section 4.6) ainsi que les caractéristiques *multithread* intégrées au simulateur DAMNED (section 4.7).

4.1 Objectifs

Le simulateur proposé et développé dans ce travail a plusieurs objectifs. Le premier est de permettre la simulation de réseaux de neurones de grandes tailles. Pour ce faire, le simulateur DAMNED fait le choix de l'utilisation ressources matérielles distribuées et du *multithreading*. Un deuxième objectif repose sur le fait que ce simulateur ne doit pas se contenter de fournir la possibilité de faire de grands réseaux, mais également que ces réseaux puissent faire usage de méthodes d'apprentissage, qu'ils puissent être constitués de plusieurs modèles de neurones différents mais surtout, le simulateur DAMNED est destiné à fournir un support logiciel au modélisateur lui permettant de s'affranchir des problèmes de parallélisation, de communications et de synchronisations inhérents aux simulations distribuées.

Enfin, il est indispensable, pour le simulateur, que son utilisation soit aisée pour la définition, la création et la simulation de modèles de réseaux directement inspirés de la neurophysiologie.

4.2 Présentation

Une simulation de DAMNED est composée de P processus logiques (LP) tels qu'ils ont été présentés au chapitre 3. Le simulateur est entièrement programmé en C++ dans une architecture objet.

Chaque LP est en charge d'un sous ensemble des neurones de la simulation (la figure 3.4 présente un exemple de répartition).

La simulation fonctionne sur le principe d'échanges d'événements entre les neurones. Les LP sont donc amenés à échanger entre eux des informations sur le déroulement de la simulation.

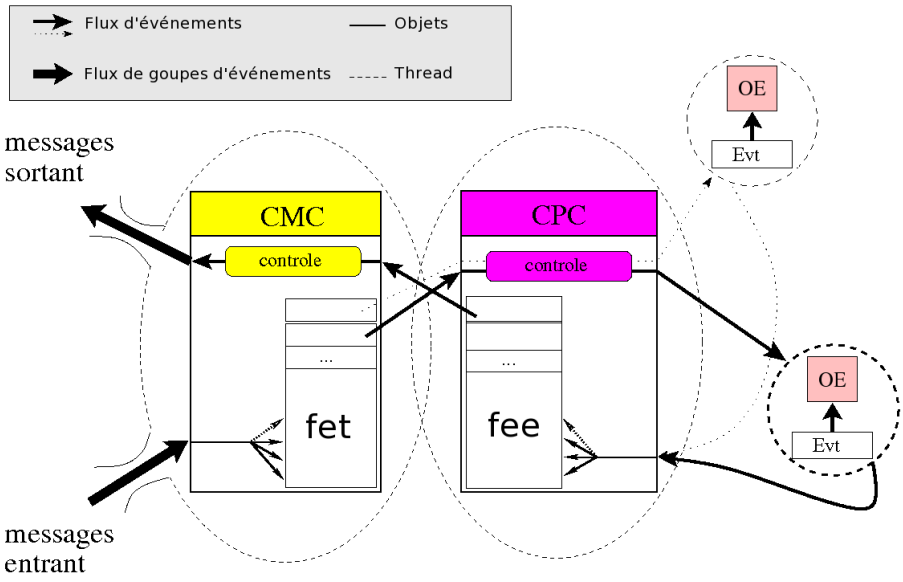

Fig. 4.1 – *Schéma général présentant le fonctionnement de base d'un LP. Un LP contient deux threads permanents : CPC et CMC. CMC est en charge des échanges de messages (entrant et sortant) avec les autres LP. CPC est en charge des traitements d'événements locaux. CPC et CMC interagissent par l'intermédiaire de files d'événements. Ces files contiennent les événements en attente de traitement (file remplie par CMC et vidée par CPC), ou des événements en attente d'émission (file remplie par CPC et vidée par CMC). Un contrôle spécifique est appliqué pour assurer la cohérence des flux d'événements.*

Ces informations seront échangées à l'aide de messages circulant sur le réseau entre les LP. Chaque neurone simulé est considéré par le simulateur comme un Objet Événementiel (OE) dont l'aptitude principale est d'être en mesure de recevoir un événement et d'en émettre éventuellement un en retour.

Sur chaque LP, deux processus légers (*threads*) sont actifs en permanence et gèrent le déroulement de la simulation : le CPC pour ComPutation Controller et le CMC pour ComMunication Controller. Le schéma de la figure 4.1 illustre le fonctionnement général de chaque LP dans les simulations exécutées avec DAMNED. Les sections suivantes permettront de comprendre plus précisément le fonctionnement.

4.3 Éléments constitutifs

4.3.1 Les événements

L'élément de base de la simulation est l'événement. Celui-ci est émis par un neurone en direction d'un ou plusieurs neurones cibles. Une simulation complète peut aisément représenter plusieurs milliards d'événements traités. Tout nouvel événement est daté par l'émetteur. De plus l'émetteur fournit son identité à l'événement (figure 4.2) de sorte que chacune des cibles de cet événement puisse identifier la connexion activée.

Ces évéments sont échangés par les LP de la simulation sous forme de messages regroupant plusieurs événements. Un LP émet vers un autre LP un ensemble d'événements dont les cibles sont hébergées par ce dernier.

4.3.2 Les Objets Événementiels

On assimile les éléments du réseau simulé, dans notre cas des neurones, à des objets dont la principale caractéritique est de pouvoir recevoir un événement et le traiter. Suite à ce traitement un événement peut être émis par l'« Objet Événementiel » (OE), voir figure 4.2.

Chaque événement généré par un OE est émis vers toutes les cibles connues de cet OE. Ces cibles sont déterminées a priori lors de la création du réseau.

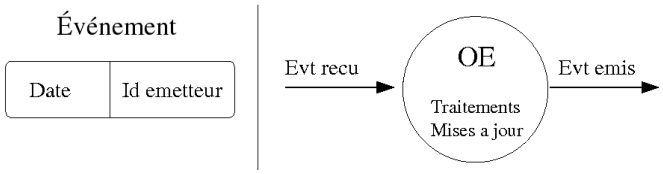

Fig. 4.2 – *Représentation d'un événement, à gauche, il possède un champ identité et un champ date d'émission. Représentation d'un Objet Événementiel, à droite, il effectue un traitement sur un événement et peu en émettre un en retour.*

Le mode de simulation choisi pour le simulateur est un mode événementiel (voir chapitre 2). Il est important de noter que les OE ne peuvent effectuer les mises à jour de leurs variables internes (figure 4.2) que lors du traitement d'un événement. C'est à ce moment que leurs calculs leur indiquent s'ils vont générer un nouvel événement.

Toutes les simulations représentant des réseaux d'objets répondant à ces critères peuvent donc être exécutées avec DAMNED. Si l'on se restreint aux réseaux de neurones impulsionnels, ceci donne par exemple la possibilité d'étudier des réseaux interconnectés de nombreux types de neurones différents. Les réseaux simulés peuvent être hétérogènes, le nombre de type de neurones au sein d'une même simulation n'est pas limité.

C'est le thread CPC qui attribue les événements à traiter aux Objets Événementiels.

4.3.3 CPC : Computation controller

Le thread CPC porte ce nom du fait de la signification anglaise de cet acronyme que l'on peut traduire par : « gestionnaire de calculs ». Ce nom lui a été donné car

CPC est un thread qui utise le temps qui lui est alloué pour effectuer les traitements d'événements locaux, c'est-à-dire ceux concernant les OE (les neurones) hébergés sur le LP local (voir figure 4.1).

Lors de l'exécution d'une simulation, l'algorithme 4 (page 61) est effectué par tous les LP. Dans le simulateur DAMNED, le CPC effectuera la partie de cet algorithme concernant la mise à jour de l'état du LP (algorithme 5). L'état du LP est fourni par l'état de tous les neurones hébergés par celui-ci. La mise à jour de l'état du LP consiste donc à mettre à jour les variables d'état des neurones impactés par les événements à traiter.

Algorithme 5 : Algorithme de base pour un CPC dans DAMNED

tant que *!fin* **faire**
  Récupérer le prochain événement à traiter.
  Associer l'événement à l'OE cible.
  Récupérer l'événement émis s'il a lieu.
fin

Détaillons les trois étapes de cet algorithme (illustrées sur la figure 4.3) afin de comprendre le principe du traitement des événements dans DAMNED.

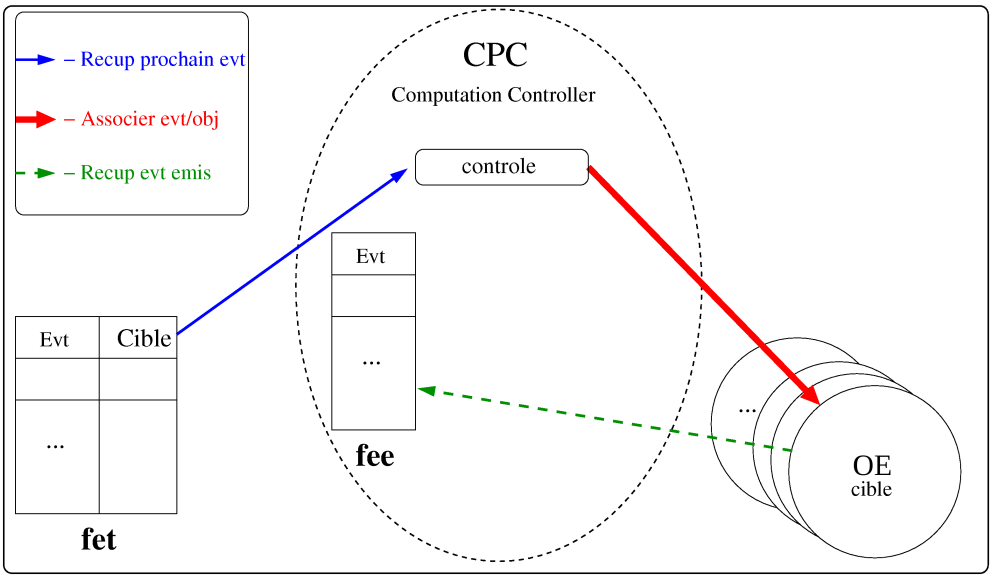

FIG. 4.3 – *Les trois étapes de l'algorithme effectué par CPC sont illustrées. Récupération éventuelle du prochain événement (en bleu, trait fin), association de l'événement à l'OE cible (en rouge, trait épais) et récupération de l'événement émis par l'OE cible (en vert, trait pointillé)*

Le prochain événement à traiter

Le CPC récupère le prochain événement à traiter depuis une file de priorité : la « file d'événements à traiter » (*fet*) dans laquelle sont insérés les événements reçus des autres LP de la simulation ou du LP local. Dans la *fet*, les événements stockés sont tous couplés à une cible donnée. CPC récupère donc un couple (événement,cible).

La récupération du prochain événement à traiter est soumise à un contrôle strict (entre la fet et les OE cibles de l'événement sur les figures 4.1 et 4.3). Un tel contrôle est nécessaire pour maintenir la cohérence temporelle désirée à travers tous les LP de la simulation. Ce point fait l'objet de la section 4.5. Si un événement est obtenu lors de la requête, l'événement en question est alors retiré de la file et stocké temporairement (en bleu sur la figure 4.3).

Associer l'événement à sa cible

Une fois la paire (événement,cible) récupérée, le CPC recherche la cible dans l'ensemble des OE que le LP héberge et le CPC fournit l'événement à l'OE cible. Ce traitement peut être effectué à l'aide d'un *thread* dédié (voir section 4.7), ou bien exécuté séquentiellement par le *thread* CPC.

Récupérer l'événement émis

À la suite du traitement d'un événement, l'OE peut générer un événement (par exemple, en réponse à la stimulation). Si c'est le cas, l'événement généré est placé dans la « file d'événements à émettre » (fee) (à gauche dans CPC sur la figure 4.3). Cette seconde file de priorité effectue un classement temporel des événements en attente d'émission. Ces émissions peuvent être à destination d'autres LP ou à destination du LP local. C'est le CMC qui sera chargé d'effectuer l'envoi des événements générés.

Structures de données

Le CPC est défini par son fonctionnement, mais également par les structures de données qu'il contient. Tout d'abord, il contient un stock d'OE. C'est une structure de type *table de hachage* est utilisée pour conserver les OE hébergés par le LP. Ce type de structure permet un accès aux éléments à partir d'une clef. Grâce à ces clefs, ces conteneurs dits *associatifs* sont capables d'effectuer des recherches d'éléments de manière optimisée. Les opérations de recherche se font généralement avec un coût logarithmique seulement, ce qui reste généralement raisonnable même lorsque le nombre d'éléments stockés devient grand. Les conteneurs associatifs sont donc particulièrement adaptés lorsqu'on a besoin de réaliser un grand nombre d'opérations de recherche. La librairie standard (STL) C++ fourni un ensemble de conteneurs associatifs (e.g. std : :set, std : :map).

Le CPC contient une file d'événements : la fee, pour les événements à émettre. Des détails d'implémentation de cette file sont fournis en section 4.6.

Le CPC contient une date représentant la date de traitement actuelle. Il contient également deux *mutex*, le premier pour l'accès à la file d'événements et le second pour l'accès à la date de traitement actuelle. Un *mutex* permet de sécuriser l'accès à une ressource par plusieurs processus (voir section 4.7).

4.3.4 CMC : CoMmunication Controller

Le CMC porte ce nom du fait de la signification anglaise de cet acronyme qui signifie : « gestionnaire de communications ». À l'instar du CPC, CMC est un *thread*. Le *thread* CMC est en charge des communications entre les LP.

Lors de l'exécution d'une simulation, CMC effectue l'algorithme 6.

Algorithme 6 : Algorithme de base pour un CMC dans DAMNED

tant que *!fin* **faire**
  Émettre des messages
  Réceptionner des messages.
  Préparer l'émission des événements.
fin

Détaillons les différentes étapes du travail effectué par CMC (illustrées sur la figure 4.4).

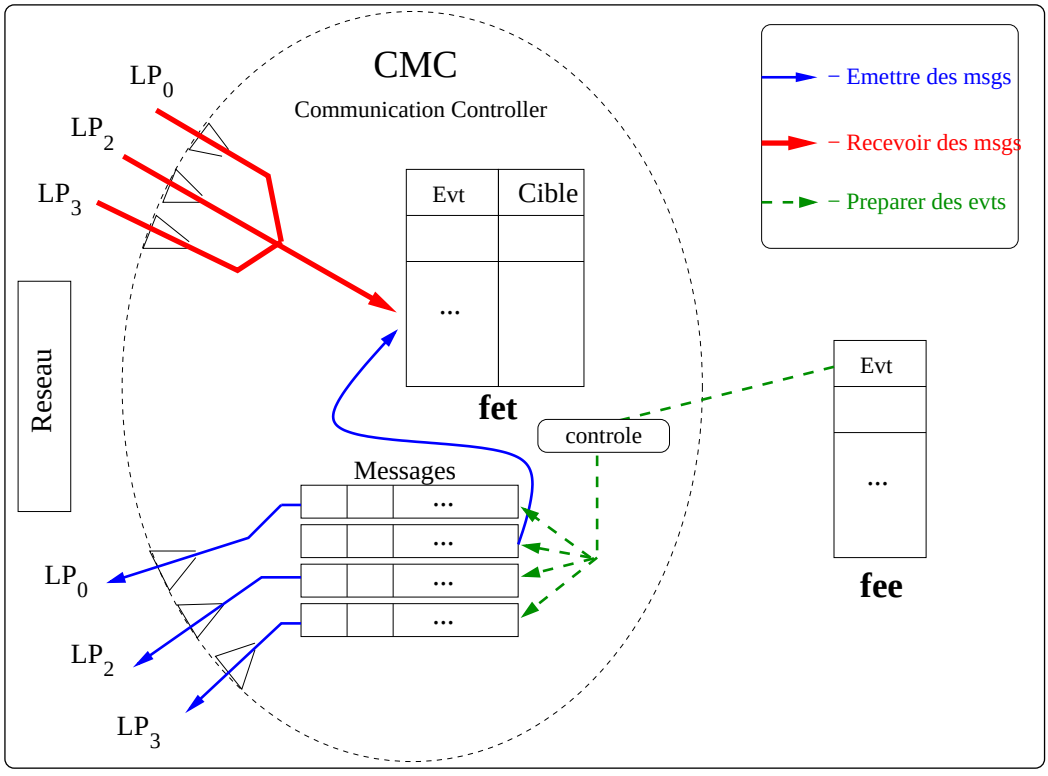

FIG. 4.4 – *Le fonctionnement du LP 1 est illustré. Une simulation contenant 4 LP est considérée. Les trois étapes de l'algorithme effectué par CMC sont représentées : émission de messages vers les autres LP en trait fin (bleu), réception des messages venant des autres LP en trait épais (rouge) et préparation des événements à émettre en trait pointillé (vert).*

Émettre des messages

Le CMC est chargé de l'émission des messages. C'est donc entre les CMC de la simulation que se font les communications. Les protocoles utilisés pour les communications sont basés sur la norme MPI 2.0 (voir section 3.2.1).

Dans sa tâche d'émission, le CMC effectue deux phases :

1. vérifier s'il y a des messages à envoyer à d'autres CMC (d'autres LP)
2. vérifier s'il y a des événements locaux destinés à des OE locaux.

Ces messages sont stockés dans des tableaux regroupant les événements par LP destinataire. Tous les envois et réceptions de messages sont non-bloquants. Cela signifie que pour chaque opération d'envoi ou de réception, une requête spécifique est créée. Une unique requête d'envoi (respectivement réception) peut être créée pour chaqueLP du réseau. Il existe donc un canal d'émission (flèches sortantes (bleues) vers les LP_i) et un canal de réception (flèches entrantes (rouges) depuis les LP_i) entre chaque LP de la simulation (voir figure 4.4). Lorsque la requête est honorée, une nouvelle requête du même type peut être créée.

Dans la phase 1, si aucune requête d'émission vers les LP concernés n'est en cours, une nouvelle requête d'émission vers chacun des LP concernés est créée et l'envoi commence dès que possible.
Dans la phase 2, tous les couples (événement,cible) destinés à des cibles locales sont transférés dans la fet locale, où le CPC les récupérera (figure 4.4, flèche bleue vers fet). Le délai temporel (en temps virtuel) de transmission d'un événement entre l'OE émetteur et l'OE récepteur, tel que défini dans le réseau simulé, est pris en compte par CMC avant l'insertion dans la fet.

Recevoir des messages

Les messages reçus des autres LP de la simulation contiennent des couples (événement,cible) qui sont directement extraits vers la fet (figure 4.4, flèche vers fet (rouge)). Il sont récupérés plus tard par le CPC. En plus des événements, chaque message contient des informations sur le temps virtuel global vu par le LP émetteur du message (voir section 4.5).

Préparer des événements pour l'émission

Avant d'être effectivement émis, les événements générés sur un LP doivent en premier lieu passer un contrôle strict (voir section 4.5) permettant de déterminer si l'événement peut ou non être émis. Ensuite, le CMC récupère les cibles atteintes par l'objet émetteur depuis une table de connexions partielle déterminée a priori à la création du réseau. Cette table est partielle : seules les connexions du réseau utiles à un LP (les connexions utilisées par les OE lors de l'émission de PA) sont stockées dans cette table.

Une fois extraite la liste des OE cibles, les couples (événement,cible) sont stockés dans des messages en attente d'envoi (voir figure 4.4). Chaque message est étiqueté en fonction du numéro du LP auquel il est destiné. Ces messages sont ensuite émis par le CMC dès que possible (section 4.3.4).

Structures de données

Le CMC contient tout d'abord une file d'événements : la fet. Cette file contient des couples (événement,cible) qui sont destinés au traitement local par le CPC. Des détails d'implémentation de cette file sont fournis en section 4.6.

Le CMC contient également un tableau d'horloges et un tableau de booléens *horloge_mise* (voir section 4.5.3) contenant chacun autant d'éléments qu'il y a de LP dans la simulation.

Le CMC contient trois *mutex* pour les accès concurrents à la fet, à l'horloge virtuelle et au tableau de booléens *horloge_mise*. Il contient également une table de hachage associant à chaque numéro de LP une liste d'événements destinés à ce LP en attente d'émission.

Deux tables de hachages supplémentaires sont utilisées : la première, pour associer les OE, hébergés par le CPC, à leurs cibles respectives : la table des connexions, et la seconde pour associer les OE connus du CMC au numéro du LP qui les héberge : la table de propriétés. Les OE connus du CMC sont tous les OE qui sont cibles des OE hébergés par le CPC, pour qu'à chaque émssion autorisée d'événement, le CMC puisse y associer les différentes cibles et leurs LP hôtes respectifs.

4.3.5 Entrées-Sorties et processus Environnement

Pour engendrer une dynamique dans le réseau de neurones modélisé, il est nécessaire de pouvoir envoyer des stimuli au réseau durant la simulation.

Dans ce but, l'un des LP, lors d'une simulation, est considéré comme un LP « Environnement ». Ce LP est dédié à l'exécution d'un processus environnement défini par le modélisateur, pour le réseau de neurones (par exemple un environnement virtuel de type proie/prédateur). Cet environnement sera chargé de fournir les « entrées » du réseau et de récupérer les activités désirées en tant que « sorties ». Cet environnement est un *thread* et peut interagir avec le réseau durant la simulation. Le rôle de ce LP, au sein de la simulation DAMNED, n'est pas différent des autres, il émet et reçoit des messages et il traite les événements qui lui sont destinés. Cependant, le LP porteur du *thread* Environnement ne dispose pas d'un *thread* CPC. Au démarrage de la simulation, le CMC et l'environnement sont exécutés sur ce LP.

Les stimulations appliquées par le *thread* Environnement utilisent des « cellules d'entrée » connectées au réseau de neurones. De la même manière, des « cellules de sortie » sont connectées aux neurones du réseau (selon le modèle implémenté).

Dans une simulation DAMNED, le réseau de neurones modélisé est donc réparti sur $P-1$ LP. Le LP_0 étant dédié aux entrées-sorties et aux interactions utilisateur. Cependant, la gestion interne de tous les LP est identique, c'est pourquoi nous considérerons le LP_0 comme n'importe quel LP (sauf indication explicite).

4.4 Cycle d'un événement

Pour mieux comprendre le fonctionnement du simulateur, on peut décrire le parcours d'un événement PA à travers le simulateur, depuis son émission jusqu'à la réception des PPS générés.

La figure 4.5, décrit ce cycle. On considère un événement *evt* dans une simulation :

1. celui-ci est en premier lieu émis par un Objet Événementiel OE_{idi} ;
2. il est alors récupéré par CPC qui le place dans la *fee*. L'événement est inséré dans cette file en fonction de sa date ;
3. lorsque l'événement se retrouve en première position dans la file, il est alors récupéré par CMC (dans l'étape de préparation des événements à émettre de l'algorithme 6) ;
4. CMC recherche l'identité des cibles atteintes par l'émetteur ;
5. pour chaque cible, une instance de l'événement est ajoutée avec l'identité de la cible à un message en attente d'envoi. Il y a autant de messages en attente que de LP potentiellement atteints par le LP local. Dès que tous les couples ont été insérés dans les messages, l'événement d'origine est détruit ;
6. les messages contenant les copies de l'événement initial sont envoyés par CMC vers les autres CMC de la simulation (flèches pointillées (en bleu) sur la figure 4.5) ;
7. l'un de ces messages est destiné au nœud local ;
8. les couples (événement,cible) contenus dans ce message sont alors placés directement dans la *fet* par le CMC local. Avant l'insertion, CMC modifie la date de l'événement pour lui ajouter la valeur du délai de transmission (en temps virtuel) entre l'émetteur et la cible ;
9. si l'OE émetteur de l'événement a une connexion avec lui-même, CPC récupérera, en temps voulu, le PPS et fournira cet événement à l'OE pour traitement, ce qui achèvera le cycle de vie complet de l'événement.

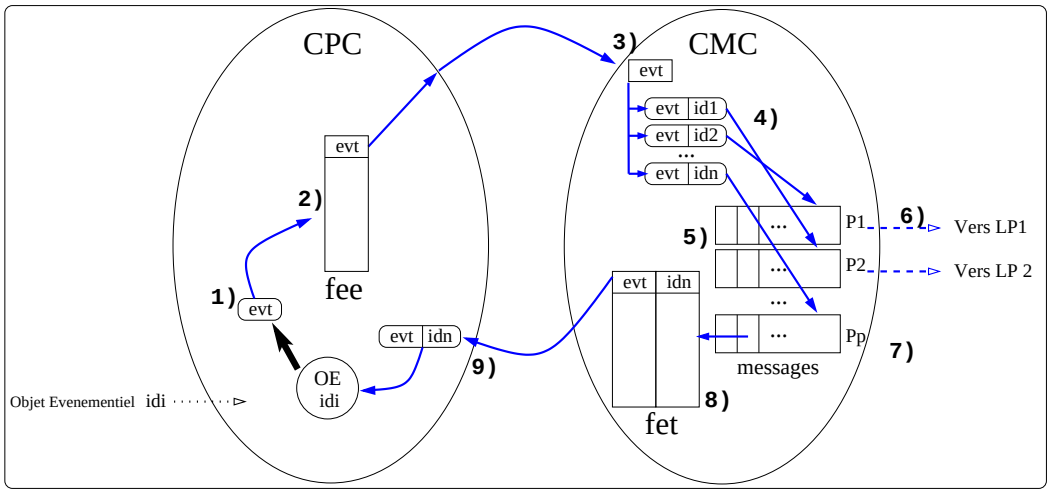

FIG. 4.5 – *Le cycle d'un événement dans le simulateur, de la génération au traitement. fee est la file d'événements à émettre, et fet est la file d'événements à traiter. Le cycle de l'événement evt émis (flèche noire épaisse) par l'OE id_i est illustré. Les CMC et CPC illustrés ici sont contenus dans le LP_p. Les étapes 1) à 9) du cycle sont détaillées dans la section 4.4).*

4.5 Gestion du temps virtuel

Le propre des neurones impulsionnels est d'introduire une notion temporelle dans le traitement connexionniste (voir section 1.1.3, page 5). L'ordre temporel (au sens du temps virtuel) des PA reçus par un neurone est donc particulièrement important dans les simulations de réseaux de neurones impulsionnels. En effet, cet ordre influence directement la dynamique des neurones et des synapses. De plus, il est de plus en plus courant d'exploiter la topologie des délais de transmission dans ces réseaux de neurones. Le respect des topologies définies et la reproductibilité des dynamiques sont donc d'une importance particulière pour les simulateurs.

Dès lors, il est essentiel que le simulateur DAMNED assure cet ordre temporel à un niveau global afin que la simulation se déroule exactement comme le réseau de neurones modélisé le définit. Si celui-ci est entièrement déterministe, alors deux simulations successives doivent générer exactement le même résultat. Si le réseau simulé introduit de l'indéterminisme (par exemple par le biais de variables aléatoires au niveau des délais, des synapses, des membranes, etc), alors cet indéterminisme seulement doit influencer la simulation. Pour assurer cette rigueur, un contrôle strict est appliqué sur l'évolution du temps virtuel global en fonction des événements émis et traités au sein de chaque LP de la simulation. Ce contrôle sera effectué par les *threads* CMC et CPC. Les méthodes utilisées sont inspirées des méthodes conservatives asynchrones proposées pour les simulations parallèles événementielles (voir chapitre 3).

4.5.1 Simulations conservatives asynchrones

Les simulations effectuées avec DAMNED sont de type asynchrones. Chaque LP de la simulation fonctionne indépendamment des autres. Chaque LP se compose d'un *thread* CMC et d'un *thread* CPC. Chaque LP a donc sa propre vision du temps virtuel global qui doit être régulièrement confrontée à celles des autres LP pour assurer le bon déroulement des simulations. Aucune synchronisation prédéfinie, comme par exemple des envois de messages à pas de temps fixes, n'est implémentée dans DAMNED. L'activité interne de chaque LP (due à l'activité du réseau de neurones sous-jacent) déterminera les synchronisations nécessaires au maintien du temps virtuel global (section 4.5.3).

Ces simulations sont d'autre part conservatives car DAMNED fait le choix de traiter les événements uniquement dans leur ordre chronologique. Au niveau local à un LP, dès lors qu'un événement dont la date d'émission est d_e a été retiré de la file, aucun événement dont la date $d'_e < d_e$ ne pourra être inséré dans la file.

Utiliser ce type de méthode de gestion du temps virtuel implique qu'aucune détection et/ou récupération d'erreurs n'est nécessaire au simulateur DAMNED pour assurer une simulation correcte, sans perte d'événements ni inversions temporelles, et ceci sans aucun « interblocage » (*deadlock*). Ce type de méthode assure que le temps virtuel global est strictement respecté. Tous les événements émis sont considérés comme effectivement reçus.

Par ailleurs, les réseaux physiques de processeurs supportant la simulation sont définis de manière statique, comme par exemple des réseaux locaux (ou LAN). Ils peuvent être hétérogènes mais restent fixes et utilisables pendant toute la durée de la simulation ainsi que le réseau de communications physique. Ainsi on considère

que toute communication entamée entre deux processeurs se terminera correctement. Dans sa version actuelle, le simulateur DAMNED n'est donc pas destiné à des réseaux dont le nombre et l'identité des machines utilisables sont variables.

Assurer une méthode de gestion du temps conservative et asynchrone nécessite en contrepartie un contrôle strict des envois et traitements d'événements et ainsi, souvent, une moindre anticipation des LP quant à l'avancement possible du temps virtuel par rapport à une gestion optimiste (voir section 3.3.3). Cependant, nous avons vu (section 3.3.4) que certaines propriétés des réseaux de neurones simulés, notamment en termes de distribution des délais, peuvent permettre une anticipation conséquente (voir section 4.5.2) sur les futures arrivées d'événements.

Gestion distribuée du temps virtuel global

La méthode conservative implémentée permet d'éviter les interblocages pour les LP de la simulation. Cette méthode repose sur une gestion distribuée du temps virtuel global plutôt que grâce à un LP « contrôleur central ». Ceci évite le goulot d'étranglement que représente un gestionnaire central. En effet les communications seraient en majorité concentrées vers ce LP, ralentissant ainsi ce dernier, or tout « ralentissement » de ce LP pénalise immédiatement l'ensemble des LP.

En contrepartie une gestion distribuée augmente sensiblement le nombre de communications entre les LP (voir section 3.2.4, page 57). Cependant au niveau de chaque LP le calcul et l'envoi ne dépendent plus d'un unique LP de la simulation mais de tous. Cela peut sembler gênant puisque l'on est alors bloqué par le plus lent de tous les LP. Pourtant, à l'inverse, en tenant compte des informations venant de tous les autres processeurs, on multiplie les chances de débloquer le LP local avant la réception des informations du LP le plus lent. L'architecture du réseau de machines sous-jacent a alors une influence.

Minimisation de la mémoire et de la charge du processeur

Adopter une stratégie conservative évite la gestion de fenêtres temporelles de « sauvegarde » lourdes en ressources mémoire (voir section 3.3.3, page 62). Le stockage et la récupération suite aux erreurs représenterait d'une part, une quantité de mémoire supplémentaire puisque dupliquant les situations successives, et d'autre part une charge computationnelle supplémentaire pour les processeurs afin de récupérer les erreurs. Les ressources mémoire économisées à ce niveau pourront notamment servir pour la simulation de réseaux de neurones plus étendus.

Flexibilité pour les objets événementiels

Tout Objet Événementiel défini pour être utilisable par le simulateur doit répondre aux critères exigés par ce dernier. Dans notre cas, adopter une stratégie conservative évite aux OE d'implémenter une récupération en cas d'erreur. En effet, si une erreur commise dans la chronologie des événements était prise en compte par le simulateur, alors un OE pourrait se voir fournir un événement erroné dont il devrait annuler l'effet en cas de récupération, ce qui impliquerait, par exemple, la mise à jour des variables internes de l'OE. Le simulateur garantit ici qu'aucun événement erroné ne sera fourni à un OE. Ceci permet donc aux modélisateurs de créer des modèles d'OE

dans lesquels aucune interaction directe avec le simulateur due à des erreurs n'est nécessaire.

Asynchronisme des simulations

Par essence, les simulations de neurones impulsionnels sont asynchrones. Chaque neurone évolue théoriquement parallèlement à tous les autres. Ainsi, à un instant donné de la simulation, les différents LP peuvent théoriquement effectuer simultanément et de manière indépendante, des calculs à partir de données différentes. L'évolution du temps virtuel global est donc soumise à la dynamique du réseau de neurones. C'est pourquoi une gestion distribuée des horloges de la simulation a un intérêt particulier dans ce type de simulations.

4.5.2 Horloges virtuelles

Pour que les événements soient conservés dans leur ordre chronologique (virtuel), DAMNED doit donc respecter précisément les dates de ces événements. De plus, les simulations informatiques sont contraintes de discrétiser le temps. La précision de cette discrétisation varie selon les simulations, en moyenne cette précision va de la milliseconde jusqu'à la nanoseconde environ.

Les LP_i de la simulation considèrent par défaut que leur date actuelle d'émission te_i est celle du prochain événement à émettre (le premier événement de la fee).

Chaque LP_i possède une version locale de l'horloge virtuelle globale. Cette horloge virtuelle est constituée d'un tableau contenant les dates d'émission spécifiques de chaque LP de la simulation (voir figure 4.6).

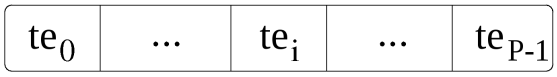

FIG. 4.6 – *L'horloge conservée sur chaque LP_i de la simulation. Il s'agit d'un tableau dont chaque case j ($0 < j < P - 1$) contient le temps virtuel local (date du prochain événement à émettre) de chaque LP_j, connu par le LP_i à un instant donné de la simulation. Ici P est le nombre total de LP dans la simulation.*

L'horloge virtuelle globale, à un instant donné de la simulation, peut donc être définie comme un vecteur contenant les horloges virtuelles locales de tous les LP.

4.5.3 Synchronisations des horloges virtuelles

Afin que le temps reste cohérent au niveau global de la simulation, c'est-à-dire entre les LP, il est nécessaire de mettre en place une méthode de « mise à jour » des horloges virtuelles locales. Le simulateur DAMNED procède à une synchronisation des horloges locales par échanges de messages.

Gestion distribuée des horloges

La gestion distribuée des horloges implique que des messages, porteurs d'informations relatives aux horloges, soient échangés par les LP. Pour limiter les communications, le simulateur DAMNED transmet ces informations, autant que possible, dans

les messages contenant des événements. Lorsque c'est nécessaire, certains messages, ne contenant que des informations d'horloges, seront transmis.

Sur un LP, deux files d'événements cohabitent, la fee et la fet. Il y a donc deux dates distinctes représentant la date actuelle d'émission te et la date actuelle de traitement tt, correspondant respectivement à la date du dernier événement extrait de la fee et celle du dernier événement extrait de la fet.

À chaque émission par un LP_i d'un message, contenant des événements, vers un LP_i, le tableau d'horloges du LP_i est envoyé au LP_j en tête de message.

Dans le simulateur DAMNED, le temps débute à la date $t_0 = 0$ et croit jusqu'à la date de fin de simulation. Les conventions suivantes sont utilisées pour décrire l'évolution des dates te et tt :

1. lorsqu'un LP a été autorisé à émettre (respectivement traiter) à une date t alors te (respectivement tt) est strictement positive et vaut t ;
2. lorsqu'un LP n'a pas été autorisé à émettre (respectivement traiter) à une date t alors te (respectivement tt) est strictement négative et vaut $-t$;
3. lorsque la fet est vide, la date tt est nulle. Lorsque la fet et la fee sont vides, la date te est nulle. ;

Ces conventions permettent aux LP de se transmettre une information supplémentaire dans les messages, informant sur l'état des files d'événements locales .

Pour émettre des événements, localement ou vers d'autres LP, les LP doivent s'assurer au préalable que les horloges locales de chaque LP soient telles qu'aucun des autres LP ne peut émettre un événement qui viendrait interdire l'émission.

Un contrôle est donc effectué par CMC (section 4.3.4) pour confirmer ou non l'émission du prochain événement en attente dans la fee.

Lorsque sur un LP_i la fee est vide, la date te envoyée aux autres LP, est calculée par l'algorithme 7.

Algorithme 7 : File d'événements à émettre vide

si $tt_i \neq 0$ **alors**
  $te_i \longleftarrow -|tt_i - \delta_{min}|$
sinon
  $te_i \longleftarrow 0$

Ceci permet à un LP n'ayant plus rien à émettre de ne pas bloquer le déroulement de la simulation. Dans le cas où la date de traitement n'est pas nulle, soit cette date est négative et cela signifie que le CPC est en attente d'un traitement à la date $-tt_i$, soit elle est positive et cela signifie que le CPC a déjà été autorisé à effectuer des traitements à la date tt_i. Dans les deux cas, la date actuelle de traitement peut être « en avance » de δ_{min} sur les dates d'émission. Donc le CMC peut « afficher » une demande d'émission virtuelle à la date $te_i = -|tt_i - \delta_{min}|$. Le CMC indique ainsi, aux autres LP lors de l'envoi de l'horloge virtuelle locale, une demande d'émission à cette date. Si la date de traitement est nulle, alors ce LP n'a plus rien à faire pour l'instant, il l'indique donc en passant à une horloge virtuelle nulle.

Durant la simulation, il arrive qu'un LP_i n'ait plus d'événements à émettre (lorsque sa file fee est vide). Afin de conserver la cohérence du temps virtuel glo-

bal de la simulation, des messages contenant uniquement les horloges connues par le LP_i pourront alors être transmis.

La conservation de la cohérence temporelle nécessite donc parfois l'envoi de certains messages, qu'on dira de type « null message » car ne contenant que des horloges, lorsque les horloges virtuelles d'un LP ne sont pas à jour. Dans ces cas, un LP peut être temporairement bloqué, en attente de nouvelles mises à jour de son horloge virtuelle, lors de la réception de prochains messages, contenant ou non des événements.

Lorsque le CPC (resp. CMC) d'un LP_i est amené à changer sa date actuelle de traitement tt_i (resp. d'émission te_i), un contrôle vérifiant la cohérence des horloges virtuelles locales doit être effectué. En effet, tant qu'aucun changement de date n'intervient, les LP peuvent continuer à traiter (resp. émettre) les événements en attente.

Le contrôle à l'envoi

Lorsque le CMC d'un LP_i, ayant te_i pour date d'émission actuelle, teste la possibilité d'émettre le prochain événement en attente dans fee, celui-ci récupère la date d'émission d_e de cet événement et effectue l'algorithme 8 pour confirmer ou infirmer l'autorisation d'émission.

Algorithme 8 : Autorisation d'émission d'un événement

```
1  si !evt.valide alors
2  |  détruire evt
   |  → émission !Ok
   sinon
   |  si d_e = te_i alors
   |  |  → émission Ok
   |  sinon
3  |  |  si (tt_i > 0) et (tt_i ≥ d_e + δ_min) alors
   |  |  |  te_i ⟵ d_e
   |  |  |  → émission Ok
   |  |  sinon
   |  |  |  si horloge_emise[j] et ((te_j = 0) ou (d_e ≤ |te_j| + δ_min)), 0 ≤ j < P alors
4  |  |  |  |  te_i ⟵ d_e
   |  |  |  |  → émission Ok
   |  |  |  sinon
5  |  |  |  |  te_i ⟵ −d_e
   |  |  |  |  → émission !Ok
```

Si l'émission est autorisée, alors l'événement est extrait de fee. Les cibles de l'événement émis (les neurones post-synaptiques) sont récupérées par CMC et les événements générés sont stockés dans des paquets prêts à l'envoi (voir section 4.3.4).

La date actuelle d'émission te_i du LP_i est ensuite mise à jour à partir de la date d'émission de l'événement d_e.

Un test *evt.valide* (ligne annotée numéro 1 de l'algorithme 8), permet aux OE de générer des événements qu'ils peuvent ensuite invalider avant leur émission (dans les cas de prédictions d'émission par exemple, voir section 2.4.5). Dans le cas où

l'événement en tête de file a été invalidé (par un OE lors de la réception d'un autre événement), celui-ci est détruit (ligne annotée numéro 2).

La valeur du délai minimum de transmission d'un événement δ_{min}, depuis un OE vers un autre (ligne annotée numéro 3) est définie à la création du réseau.

Lorsqu'un changement de date est nécessaire pour émettre le prochain événement de la file, deux alternatives sont possibles.

Soit la date d'émission de l'événement est telle qu'il est impossible qu'un prochain événement reçu doive être traité avant cette date (ligne annotée numéro 4). Dans ce cas, la date actuelle du LP peut être augmentée vers la date d'émission d_e de l'événement et l'événement peut être émis.

Soit la date d'émission se situe suffisamment loin dans le futur pour qu'il soit possible qu'un prochain événement reçu doive être traité avant (ligne annotée numéro 5) et aie donc la possibilité d'annuler ou de différer l'événement actuellement contrôlé. Dans ce cas, l'événement contrôlé ne peut pas être émis, la date actuelle du LP est alors positionnée à $-d_e$.

Le principe des « null messages » repose ici sur un tableau de booléens *horloge_emise* confirmant que l'horloge virtuelle locale a bien été envoyée aux autres LP.

À chaque modification de sa date actuelle d'émission, chaque LP positionnera les booléens à $faux$ afin de forcer l'émission de l'horloge virtuelle locale à tous les LP, même si aucun événement ne doit être envoyé. De plus, lors de ces modifications, un LP qui souhaite changer sa date actuelle doit être assuré que sa connaissance du temps virtuel global est suffisante. Pour ce faire, lorsqu'un LP est autorisé à changer sa date actuelle d'émission, il positionnera toutes les dates d'émission contenues dans son horloge virtuelle à la valeur $d_e - \delta_{min}$ (si cela n'implique pas l'augmentation des dates d'émission connues des autres LP). Ce faisant, le LP s'interdit lui-même tout nouveau changement de date immédiat. Il devra attendre d'avoir reçu suffisamment d'informations des autres LP pour être de nouveau autorisé à changer sa date actuelle d'émission.

Ces contrôles permettent d'éviter les interblocages, par exemple dus au fait qu'un LP possède une horloge virtuelle dont l'une des dates n'est pas à jour et bloque le traitement ou/et l'émission. En contrepartie, cette méthode génère de nombreuses émissions de messages. Le surcoût relatif de cette méthode dépend des modèles de réseaux d'OE simulés et en particulier de leur connectivité et de leur activité. Si l'activité est équitablement répartie et qu'il y a toujours des événements à envoyer à tous les LP, alors le surcoût se limite au coût de l'ajout du tableau d'horloges aux messages émis. Aucun message supplémentaire n'est nécessaire. À l'inverse, si l'activité est concentrée sur un seul LP, alors tous les envois d'horloges virtuelles effectués par les autres LP seront effectués à l'aide de messages vides et le surcoût sera alors maximal.

À chaque réception d'un message, l'horloge virtuelle reçue est utilisée pour mettre à jour l'horloge virtuelle locale. Seule la date d'émission locale n'est pas modifiée (le LP local possède la valeur la plus à jour pour celle-ci). La date d'émission d'un LP_k émetteur de ce message est mise à jour si elle est différente de la date te_k^i connue localement par le LP_i récepteur.

Si la date d'émission te_j^k d'un LP_j, reçue par LP_i dans un message provenant de LP_k, est positive et supérieure à la date te_j^i de LP_j connue localement, alors la date te_j^i

est mise à jour. La mise à jour de la date d'émission d'un LP_j dans l'horloge virtuelle d'un LP_i peut donc être effectuée lors de la réception d'un message provenant de n'importe quel autre LP_k de la simulation du fait que chacun envoie son horloge virtuelle complète (contenant sa propre connaissance des dates d'émission de chaque LP).

À chaque émission effective d'un message l'horloge virtuelle locale, jointe au message, est envoyée vers les autres LP. Les booléens *horloge_emise* correspondants sont positionnés à *vrai*.

Le contrôle au traitement

À l'instar du CMC, le CPC d'un LP_i effectue un contrôle sur les événements. Ce traitement s'effectue avant de lancer le traitement de l'événement par l'OE cible. Le CPC doit s'assurer que l'événement peut être retiré de la fet en étant sûr qu'aucun événement antérieur ne pourra arriver dans le futur.

Lorsque la fet est vide, la date de traitement actuelle est mise à jour avec $tt_i = 0$.

Lorsque le CPC, ayant une date de traitement actuelle tt_i, souhaite effectuer le traitement d'un événement à une date d_e, l'algorithme 9 est exécuté pour confirmer l'autorisation d'effectuer le traitement.

Algorithme 9 : Autorisation de traitement d'un événement

```
1 si (tt_i = d_e) ou
  (horloge_emise[j] et ((d_e ≤ |te_j| + δ_min) ou
  (te_j = 0)), 0 ≤ j < P) alors
      tt_i ⟵ d_e
2     si (te_i = 0) alors
          te_i ⟵ −tt_i
      → traitement Ok
  sinon
3     tt_i ⟵ −d_e
      → traitement !Ok
```

Si le traitement est autorisé, le CPC extrait le couple (événement,cible) de la fet puis récupère l'OE cible depuis son stock d'OE (voir section 4.3.3). L'événement est alors fourni à l'OE pour traitement et un événement PA émis par l'OE peut être récupéré en retour. Si tel est le cas, l'événement récupéré est inséré dans la file fee et sera ensuite émis par le CMC.

Si la date de l'événement à traiter ne nécessite pas de changement de la date de traitement alors l'autorisation de traitement est obtenue.

Si la date de l'événement nécessite un changement de date de traitement, l'autorisation n'est possible que si l'horloge virtuelle locale a été communiquée aux autres LP. De plus, l'autorisation de traitement n'est fournie que si la date de l'événement garantit qu'aucun événement antérieur ne pourra arriver par la suite (ligne annotée numéro 1). Dans ce cas la date de traitement est augmentée vers la date de l'événement et le traitement est autorisé.

Dans le cas contraire, la date de traitement est positionnée à la date de l'événe-

ment, en valeur négative (ligne annotée numéro 3), indiquant son désir d'effectuer un traitement à cette date. Le traitement est interdit dans ce cas.

Une mise à jour de la date actuelle d'émission du LP peut être effectuée lorsque le changement de date de traitement est autorisé (ligne annotée numéro 2) si la date d'émission actuelle est nulle. Ceci permet d'informer les autres LP, lors de l'envois de messages, que des événements à la date tt pourront potentiellement être envoyés, mais aucun n'est actuellement en attente (le signe négatif indique l'abscence d'événements en cours d'envoi à la date tt).

4.6 Gestion des événements

Tous les événements qui transitent dans une simulation sont stockés dans les files d'événements fee ou fet. Classiquement, il s'agit de tableaux à une dimension dont l'ajout d'un élément nécessite un tri. Le coût d'une insertion dépend du nombre d'éléments dans la file et est en $O(nlog(n))$ où n est le nombre d'éléments dans la file.

Les coûts d'insertions dans des files de priorité sont donc relativement élevés, tout particulièrement lorsque les files contiennent un grand nombre d'événements. Cependant, ces coûts sont dûs aux priorités entre les événements. En analysant ces priorités, nous allons voir qu'il est possible de s'affranchir de certains de ces coûts.

4.6.1 Priorités

Les événements doivent d'une part être ordonnés selon la date qu'ils portent. Par ailleurs, le simulateur DAMNED offre également la possibilité de trier les événements selon leur nature. Dans les simulations de neurones, le caractère « excitateur » ou « inhibiteur » du neurone émetteur peut influer sur le traitement effectué par le neurone récepteur. En effet pour un ensemble d'événements E_i donné à une même date t pour un neurone n_i. Si E_i contient des événements inhibiteurs et excitateurs le comportement du neurone peut différer selon que l'on traite les événements inhibiteurs prioritairement sur les excitateurs ou non. Par exemple, si suffisament d'événements excitateurs sont traités pour entrainer le franchissement du seuil par le neurone, ce dernier émettra un PA. Cette émission peut alors être ensuite annulée du fait du traitement d'événements inhibiteurs en nombre suffisant (tous ces événements impactant à la date t). Des annulations et créations d'événements peuvent donc avoir lieu sans un tri supplémentaire des événements. Par ailleurs, cette influence peut se répercuter sur les synapses impliquées dans certains cas, notamment si un apprentissage dynamique est en place (voir section 1.9) ou si les variables internes des synapses évoluent en fonction de l'activité (voir section 1.7). DAMNED propose alors de considérer les événements inhibiteurs comme prioritaires sur les événements excitateurs.

Afin de réaliser un telle gestion des priorités des événements et de minimiser le coût computationnel des ajouts/suppressions d'événements, une structure de données spécifique est créée.

4.6.2 Structure de données

Cette structure repose sur le fait que toute simulation est réalisée avec une précision temporelle donnée. Dans le cadre des simulations événementielles, celle-ci peut être arbitrairement précise (voir chapitre 2).

Il est alors possible de représenter une fenêtre temporelle d'une durée donnée par un tableau à une dimension dont chaque case représente un pas de temps discret.

Par exemple si la simulation est effectuée à l'échelle de la milliseconde, un tableau de mille cases constituera une bonne représentation d'une fenêtre temporelle d'une seconde. Si la précision est de l'ordre de la nanoseconde et que la fenêtre temporelle souhaitée est identique, le tableau devra alors contenir 10^6 cases.

La figure 4.7 illustre la structure de données mise en place dans le simulateur en tant que file de priorité. Nous nommons ce type de file une « file à délais ». Elle est constituée d'un tableau de pointeurs sur des listes. Les listes utilisées sont celles de la STL du langage C++.

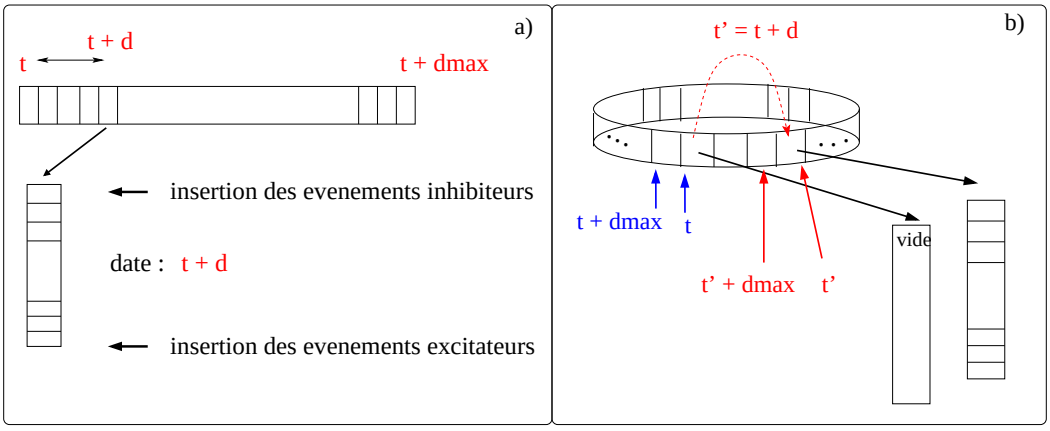

FIG. 4.7 – *Structure de données utilisée pour stocker et trier les événements locaux à un LP de la simulation.*

Lorsqu'un événement portant une date t est inséré dans une file de ce type, l'insertion se fait, sans tri, dans l'une des listes pointées par la case du tableau correspondant à cette date. La première case du tableau indique toujours la date actuelle.

Dans une simulation avec le simulateur DAMNED, le délai de transmission du PA, entre le neurone émetteur et le neurone récepteur, est appliqué par le CMC du LP hébergeant le neurone récepteur. Ainsi, pour l'événement émis à t, l'insertion sera alors effectuée dans la case du tableau d'indice $t + d$. Ce fonctionnement nécessite que le simulateur ait connaissance du délai maximal δ_{max} possible entre deux OE dans une simulation donnée. Un événement émis à t ne pourra donc pas avoir d'impact plus tardif que $t' = t + d_{max}$.

Dès lors, le tableau utilisé peut être vu comme un tableau circulaire : à chaque changement de date actuelle de la file, un pointeur indiquant la case de la date actuelle se décale et décale ainsi la dernière case du tableau circulaire, en écrasant les cases libérées (voir figure 4.7 , b). Ceci permet d'optimiser l'utilisation du tableau tout en minimisant l'espace mémoire requis.

Pour permettre la mise en place de priorités entre événements inhibiteurs et excitateurs, les insertions d'événements inhibiteurs se feront en tête d'une liste et les

événements excitateurs en queue de cette même liste pour un même délai de transmission, comme indiqué sur la figure 4.7, a). Lors de l'extraction, les événements sont extraits par la tête des listes afin que les événements inhibiteurs soient traités en premier.

Lors d'un accès à l'événement en tête de file, si la liste correspondant à la date t actuelle est vide, l'événement en tête de la première liste non vide du tableau circulaire (en partant de la position à la date actuelle).

C'est lors de la suppression d'un événement que l'indicateur de la date actuelle de la file est incrémenté jusqu'à la date de l'événement supprimé. En effet les suppressions d'événements se font toujours en tête de file, donc on est sûr qu'aucun événement n'est oublié lors de l'incrémentation.

De plus, le simulateur DAMNED garantit qu'aucun événement ne sera inséré en retard dans une file. C'est ce qui permet à la file, lors de la suppression du dernier événement de la liste, à la date courante, d'effectuer une « rotation » jusqu'à la date du prochain événement de la file. La tête de file (pointeur de date actuelle) et la fin de file (à δ_{max} pas de temps de la date actuelle) sont alors décalés dans le tableau de délais.

La file d'événements à émettre fee de chaque LP est constitué d'une file à délais dont les éléments sont des événements. La file d'événements à traiter fet de chaque LP est constituée d'une file à délais dont les éléments sont des couples (événement,cible).

4.7 Simulateur multithreadé

Chaque LP de la simulation se compose d'un *thread* CMC affecté aux communications, et d'un *thread* CPC affecté aux traitements d'événements (voir sections 4.3.4 et 4.3.3 respectivement).

Un *thread* est un processus dit léger. Il partage notamment sa mémoire dynamique avec son père, ce qui réduit les temps d'allocation et de libération de mémoire d'un *thread* par rapport à un processus. Les *threads* utilisés sont les POSIX Threads (*pthread*) Linux fournis dans la librairie GNU C. Une surcouche en C++ a été développée pour les besoins du simulateur.

L'utilisation de ces deux *threads* permet de recouvrir dans la mesure du possible les temps de communication et de calcul au sein d'un même LP. Cette propriété est particulièrement utile lors de l'utilisation de machines hyperthreadées ou multiprocesseur pour héberger chaque LP. Pour des machines ni hyperthreadées, ni multiprocesseurs (qui sont de plus en plus rares), le gain est moindre mais l'équilibrage communications/calculs reste intéressant surtout lorsque les temps de travail de chacun des *threads* ne sont pas équilibrés car l'ordonnanceur de tâches du système permet alors un ré-équilibrage.

Lors d'une simulation, ces deux *threads* concurrents effectuent des accès concurrents à des structures de données partagées en mémoire. Les structures partagées sont les suivantes :

1. la file d'événements à émettre fee ;
2. la file d'événements à traiter fet ;
3. l'horloge du LP ;

4. la date actuelles de traitement ;
5. le tableau de booléens horloge_emise.

Les opérations de « lecture » et d'« écriture » dans les zones mémoire correspondantes peuvent être effectuées simultanément (en théorie) par CPC et CMC. Les accès en « écriture » sont effectués dans les cas suivants :
- insertion d'un événement dans une file ;
- suppression d'un événement dans une file ;
- modification d'une des dates de l'horloge ;
- modification de la date de traitement actuelle ;
- modification des booléens du tableau horloge_emise.

Les accès en « lecture » sont effectués dans les cas suivants :
- accès au prochain événement de l'une des files ;
- accès à une des dates de l'horloge ;
- accès à la date de traitement actuelle ;
- accès aux booléens horloge_emise.

Dès lors, il est indispensable de mettre en place des méthodes de synchronisation des accès afin qu'aucune opération sur ces structures ne se fasse en concurrence, et ainsi éviter les erreurs d'accès. Pour cela, des mécanismes d'exclusion mutuelle sont définis pour chacune de ces structures, à partir des *mutex* fournis dans la librairie des *pthread*.

Une autre utilisation possible des *threads* est implémentée dans le simulateur DAMNED. Elle consiste à utiliser des *threads* pour permettre le traitement concurrent de plusieurs événements sur un LP. Cette fonctionnalité est à réserver à des simulations dans lesquelles les traitements effectués par les OE sont suffisamment lourds pour contrebalancer les coûts de création et de fonctionnement des *threads*. D'autre part le gain potentiel en comparaison du coût d'exploitation de multiples *threads* reste à préciser.

4.7.1 Exclusions mutelles

Le *thread* CPC qui partage sa file d'événements à émettre fee (voir figure 4.1) et sa date de traitement actuelle tt avec le *thread* CMC possède respectivement un « mutex de file » et un « mutex de date ».

Le *thread* CMC partage trois structures avec le *thread* CPC que sont sa file d'événements à traiter fet, son horloge virtuelle et son tableau de booléens *horloge_emise*.

Les *mutex* de CPC n'autorisent qu'un seul verrouillage, ce qui signifie que seul l'un des deux *threads* pourra accéder à la ressource à un instant donné.

4.8 Échanges de messages

Dans les phases d'émission et de réception, les CMC de chaque LP s'échangent des messages par l'intermédiaire de la librairie MPI.

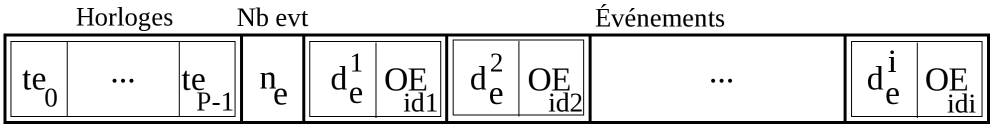

FIG. 4.8 – *Contenu des messages échangés entre les LP. L'horloge virtuelle est fournie, puis les événements en commençant par le nombre d'événements envoyés.*

4.8.1 Structure des messages

Le format des messages défini par le simulateur est tel qu'illustré sur la figure 4.8. Les messages contiennent tout d'abord l'horloge virtuelle locale du LP émetteur (voir section 4.5.2). Les messages contiennent également des événements. Le nombre d'événements contenus dans un message varie selon l'émetteur, le récepteur et l'activité du réseau de neurones ayant entraîné ce message. La librairie MPI permet de réaliser une telle définition du format des messages.

4.8.2 Mode de communication

Deux principaux modes de communication sont possibles entre les noeuds d'une exécution MPI. Il s'agit de communications dites « bloquantes » et de communications dites « non-bloquantes ». Une communication est bloquante si le nœud qui l'initie ne peut continuer ses traitements que lorsque la communication est terminée. Cette communication peut-être soit une émission, soit une réception.

Une communication est « non-bloquante » si le nœud initiant cette communication peut continuer ses traitements immédiatement après l'appel à l'émission ou la réception de messages.

Les communications bloquantes sont plus aisées à manipuler puisqu'elles servent en même temps de synchronisations entre les noeuds. Cependant elles contraignent les nœuds appelants à devoir attendre pendant toute la durée de chaque communication. Or, il est possible, selon le problème considéré, pour un nœud donné d'une simulation MPI, d'effectuer par exemple des calculs pendant que les données sont transmises sur le réseau.

Dans le cas du simulateur DAMNED, nous faisons le choix de l'utilisation de communications non-bloquantes. Dès lors, le CMC d'un LP donné dans les simulations émet et reçoit les messages en mode non-bloquant. Ainsi, dès qu'une opération de communication se termine, elle peut être récupérée par le CMC. Celui-ci teste la terminaison des opérations de communication en cours, dans les phases d'émission et de réception (voir algorithme 6). Dans des simulations composées de nombreux LP, un accès aux communications terminées est préférable à une attente de terminaison ciblée sur un LP.

4.9 Création des simulations

Le simulateur DAMNED est conçu pour être en mesure d'effectuer des simulations définies par l'utilisateur. Il ne s'agit donc pas uniquement de proposer des modèles préconstruits, mais il s'agit plutôt de fournir la possibilité au modélisateur d'utiliser

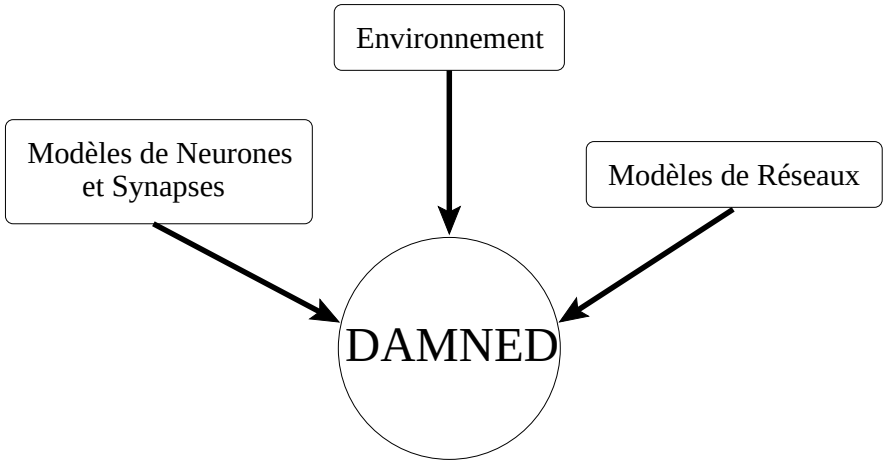

FIG. 4.9 – *Schéma représentant les différentes parties configurables dans le simulateur DAMNED.*

ses propres modèles. Le simulateur DAMNED fournit donc la possibilité de « configurer » les simulations selon plusieurs aspects illustrés sur la figure 4.9 :

- Définir des modèles de neurones et de synapses.
- Définir des modèles de réseaux.
- Définir des environnements de stimulation.

Le simulateur DAMNED manipule des OE (Objets Événementiels) qui sont plus génériques que les neurones, ainsi, la seule contrainte imposée aux modèles de neurones, outre le fait d'être des modèles événementiels, est de respecter le fonctionnement des OE. Dans le langage C++, cela signifie que les modèles de neurones développés devront hériter de la classe OE pour être utilisables par DAMNED. Les modèles de synapses ainsi que les éventuelles règles d'apprentissage locales, pourront être utilisés par les neurones ainsi définis sans contrainte supplémentaire. Respecter le fonctionnement d'un OE consiste à définir quelques fonctionnalités utilisées par le simulateur pour effectuer la simulation :

1. ajouter des connexions ;
2. tester l'existence d'une connexion ;
3. obtenir le délai d'une connexion ;
4. déterminer la priorité par rapport à un autre OE ;
5. traiter un événement.

La définition de modèles de réseaux dans le simulateur DAMNED passe par un fichier de configuration. Les réseaux sont configurés sous forme de populations de neurones. Le fichier de configuration commence par une définition du nombre de populations avec par exemple :

```
#DMD    NBPOP    4
```

Chaque population a une taille et une proportion donnée de neurones inhibiteurs (entre 0 et 1). Pour chaque population on trouvera donc des définitions du type :

```
#DMD  TAILLE_POP_1  400
#DMD  TAUX_I_POP_1  0.2
```

Les populations projettent les unes vers les autres. Les projections spécifiques d'un modèle de réseau sont énumérées dans le fichier de configuration avec une syntaxe de la forme :

```
#DMD  TYPE_POP_1_POP_2  1
```

Plusieurs types de projections différents sont initialement disponibles dans le simulateur.

Ensuite, pour chaque projection définie, les probabilités de connexion entre les neurones de la population source et ceux de la population cible peuvent être configurés à l'aide d'instructions du type :

```
#DMD  PROB_POP_1_POP_2  0.05
```

Les poids des connexions sont configurés de la façon suivante :

```
#DMD  POIDS_POP_1_POP_2  0.05
```

Autour de cette valeur moyenne, les poids des connexions peuvent être aléatoirement « dispersés » à l'aide des instructions :

```
#DMD  JIT_POIDS_MIN  0.05
#DMD  JIT_POIDS_MAX  0.2
```

Où l'instruction *M*IN définit la diminution maximale du poids d'une connexion lors de la dispersion et *M*AX définit l'augmentation maximale. Les délais sont définis pour toutes les populations. À l'instar des poids, une valeur de base est définie pour les délais, puis une dispersion aléatoire autour de cette valeur peut être définie. Par exemple :

```
#DMD  BASE_DELAI      100
#DMD  JIT_DELAI_MIN   80
#DMD  JIT_DELAI_MAX   40
```

À partir de ces informations, lors d'une simulation, chaque LP exécuté génère uniquement le sous réseau dont il a la charge afin de limiter les ressources computationnelles nécessaires.

Enfin, pour que la simulation soit prête à être exécutée, le modélisateur peut définir l'environnement de stimulation. L'environnement héberge un ensemble de cellules d'entrée et de sortie (dont les populations sont définies dans le fichier de configuration). L'environnement est un OE particulier. Le simulateur DAMNED recherche cet OE à la construction du réseau (selon l'identité de l'OE spécifiée à la configuration) et exécute la fonction *run* spécifique à l'environnement, associée à un *thread*, pour toute la durée de la simulation.

Cet environnement pourra alors :

– récupérer des événements en attente (déposés dans la fet par le CMC) ;

– ajouter des événements à émettre (dans la fee du CMC).

L'environnement peut également accéder aux cellules d'entrée et de sortie (déclarées comme telles dans le fichier de configuration). Ainsi, on peut très simplement implémenter un environnement de type « monde artificiel » tenant compte des activités des cellules de sortie pour définir les activités des cellules d'entrée au cours de

la simulation, sans que le décours temporel virtuel au sein du simulateur DAMNED ne soit affecté.

En effet, le CMC associé à l'environnement, sur le LP_0, prendra garde à ce que ce dernier ne soit pas « en traitement » lors des contrôles modifiant l'horloge virtuelle locale (sous peine de voir des événements « incorrects » temporellement être insérés dans la *fee* par la suite). Les phases pendant lesquelles l'environnement est « en traitement » sont à définir par l'environnement.

4.10 Conclusion

Nous avons présenté dans le détail le fonctionnement du simulateur DAMNED. Les différentes structures de données utilisées, les algorithmes et le mode de représentation du réseau de neurone simulé ont été développés. Le simulateur proposé allie une stratégie événementielle et un fonctionnement distribué pour accélérer les simulations de réseaux de neurones impulsionnels. Mais ce simulateur permet surtout d'exploiter de multiples ressources matérielles afin de rendre possible la réalisation de simulations dont les ressources mémoire et les ressources computationnelles nécessaires vont au-delà de celles requises par les réseaux simulables sur de simples stations de travail. La taille des réseaux, la complexité des modèles ou des règles d'apprentissage influent notamment sur ces paramètres nécessaire à une simulation donnée.

Le simulateur DAMNED est actuellement le seul simulateur de réseaux de neurones impulsionnels qui fasse un tel usage de multiples ressources matérielles à l'aide des *threads* et des échanges de messages dans des simulations distribuées. L'élaboration de ce simulateur a donné lieu à une publication acceptée (Mouraud et al. 2006) et une soumise. De nombreuses évolutions sont envisagées pour ce simulateur, certaines d'entre elles font l'objet de recherches en cours (voir section 6.8).

Évaluation du simulateur DAMNED

5

Afin d'évaluer les performances du simulateur DAMNED et d'en exploiter au mieux les capacités pour des simulations nécessitant la mise en place de grands réseaux de neurones impulsionnels, nous étudierons principalement les temps d'exécution. Contrairement à une application séquentielle classique, évaluer le simulateur DAMNED signifie l'évaluation d'un système distribué « localement » grâce aux *threads* CPC et CMC présents sur chaque LP ; distribué « globalement » par la mise en interaction de plusieurs LP ; distribué « physiquement » par l'utilisation conjointe de plusieurs ressources (processeur/mémoire).

Compte tenu de cette complexité, il est également important de noter que dans le cas de DAMNED, les temps d'exécution et l'occupation mémoire peuvent varier notamment selon le modèle de réseau simulé, selon les modèles de neurones et de synapses, selon la dynamique du réseau simulé, selon les processeurs utilisés, selon la mémoire disponible pour chaque processeur et enfin selon le réseau de communications physique utilisé.

D'autres critères influencent les temps d'exécution comme par exemple le nombre de LP utilisés par DAMNED pour réaliser la simulation et leur adéquation au réseau de processeurs physiques sous-jacent.

Au vu de ces nombreux paramètres influant sur le déroulement d'une simulation, il est nécessaire de fixer certains d'entre eux afin de restreindre le champ d'investigations de cette évaluation. Pour ce faire, nous préciserons le type de support matériel utilisé (section 5.2) et la description des modèles de réseaux et des protocoles utilisés pour cette évaluation (voir ection 5.3).

5.1 Profilage du simulateur

Dans un premier temps, on souhaite observer finement le déroulement d'une simulation au sein de DAMNED, les appels de fonctions et les parts respectives des différentes fonctions implémentées dans les temps d'exécution.

Pour ce faire, nous utilisons l'outil de profilage de programme *Valgrind* (Nethercote & Seward 2007). Il s'agit d'un logiciel libre composé de plusieurs outils dont *Callgrind* qui permet d'enregistrer tous les appels de fonctions effectués lors de l'exécution d'un programme. Ces données sont stockées dans un fichier qui peut être visualisé à l'aide d'une interface graphique (nous utilisons *KCachegrind*). L'utilisation de tels outils alourdit les simulations et multiplie par un facteur de 3 à 10 les temps d'exécution.

Pour permettre l'utilisation de ces outils, il est alors nécessaire de simuler de petits réseaux. Nous avons donc choisi un réseau contenant environ 1500 neurones. Le modèle de réseau utilisé n'est pas prépondérant ici puisque l'objectif est d'observer le déroulement de la simulation et la répartition de la charge computationnelle.

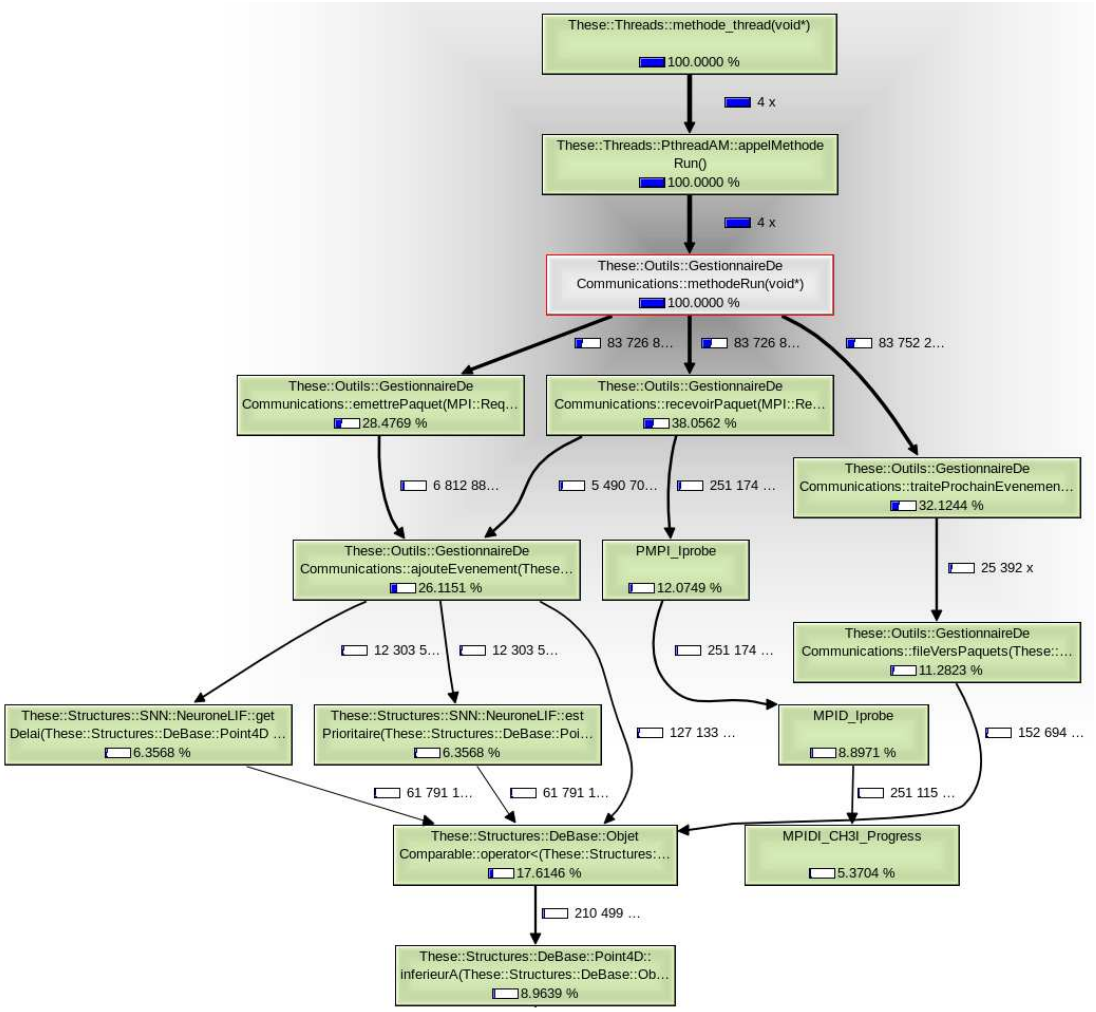

FIG. 5.1 – *Les principaux appels de fonctions effectués lors de la simulation, à partir de la fonction principale (methodeRun()) du CMC de chaque LP. Les détails des légendes sont fournis dans le texte.*

Cependant, précisons qu'il s'agit du modèle présenté dans le chapitre 6 qui implémente des interactions entre différentes populations de neurones pour la production de saccades oculaires par le tronc cérébral. Le protocole de stimulation consiste en une unique stimulation d'une durée de 150ms à une fréquence de 750Hz (voir section 6.5, la stimulation est appliquée en position 30).

Les exécutions ont été effectuées sur une station de travail bi-processeur munie de 2Go de mémoire, en utilisant 4 LP pour la simulation.

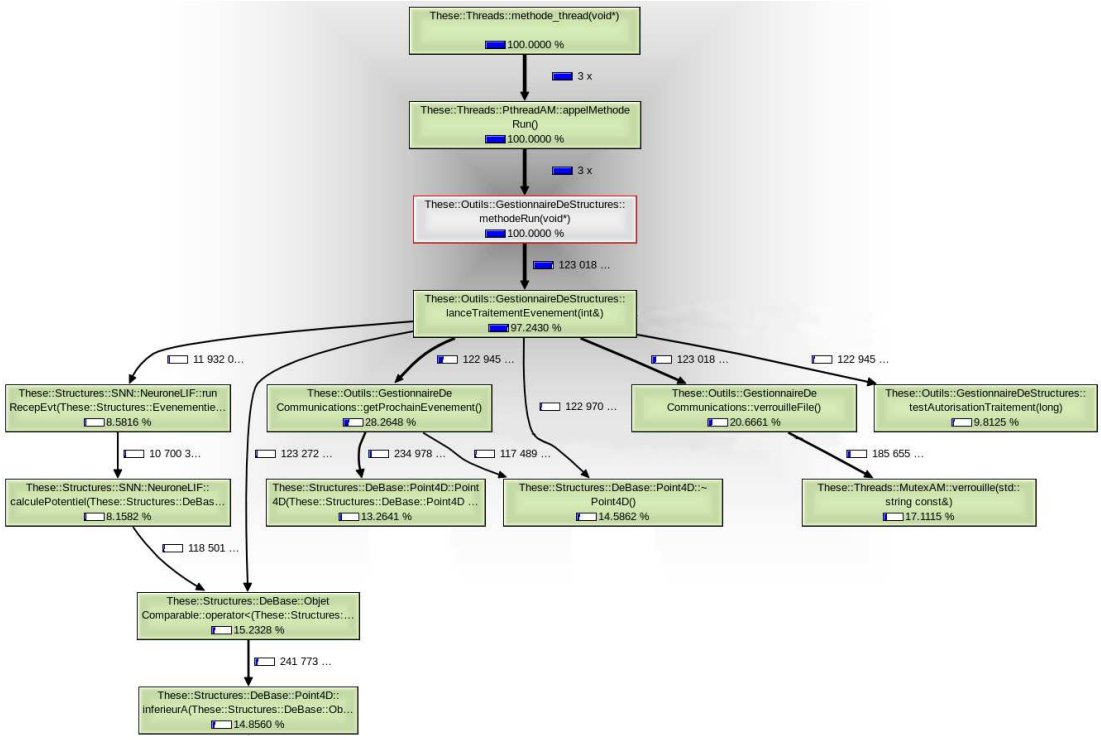

FIG. 5.2 – *Les principaux appels de fonctions effectués lors de la simulation, à partir de la méthode principale (methodeRun()) du CPC de chaque LP. Les détails des légendes sont fournis dans le texte.*

5.1.1 Contenu des graphes d'appels de fonctions

Les figures 5.1 et 5.2 présentent les graphes d'appels de fonctions obtenus suite aux simulations du réseau présenté plus haut. Les figures contiennent les données concernant le déroulement de chaque LP (le total des 4 LP est affiché sur la figure). La figure 5.1 illustre le graphe des appels des CMC dénomés : « Gestionnaires de Communications » et la figure 5.2 illustre le graphe des appels des CPC dénommés : « Gestionnaires de Structures ».

Sur chaque figure, l'épaisseur des flèches représente le poids des appels dans la charge totale de la fonction appelante. En pourcentages sont indiquées ces charges, respectivement pour chaque fonction appelée. Le nombre exact d'appels aux différentes fonctions est également indiqué (suivi d'un x, tronqué pour les nombres à plus de 6 chiffres). Les appels représentés se limitent aux appels de fonctions DAMNED et MPI. Le nom de chaque fonction est indiqué dans les cadres. La profondeur des appels est limitée à 5 et les fonctions apparaissant doivent représenter au moins 2% de la charge de la fonction appelante. La croix ombrée sur le fond de l'image indique la fonction sur laquelle est centré le graphe d'appels (la fonction *GestionnaireDeStructures : :methodeRun()*, pour le *thread* CPC).

Les 4 LP sont chacun constitués d'un CMC (GestionnaireDeCommunications) et d'un CPC (GestionnaireDeStructures), excepté le LP_0 qui, à la place d'un CPC, contient un processus environnement en charge des entrées / sorties. C'est pourquoi la fonction *methodeRun()* du CMC est appelée 4 fois comme indiqué sur la figure 5.1 et la fonction *methodeRun()* du CPC est appelée 3 fois (figure 5.2).

5.1.2 Graphe des CMC

Sur la figure 5.1 les appels effectués par la fonction *methodeRun()* qui consomment le plus de temps sont les appels aux trois fonctions principales :

1. la fonction *emettrePaquet* qui effectue les envois de messages vers les autres LP et qui ajoute les événements locaux dans la fet locale ;
2. la fonction *recevoirPaquet* qui teste les requêtes de réception de messages depuis les autres LP et qui ajoute les événement reçus dans la fet locale ;
3. la fonction *traiteProchainEvenement* qui prépare les événements (autorisé pour l'envoi) en paquets spécifiques à chaque LP de la simulation.

On constate alors que ces trois fonctions occupent chacune environ 30% de la charge computationnelle totale allouée au *thread* CMC.

Par ailleurs, on remarque que la comparaison d'objets, qui est utilisée dans les recherches des cibles d'un événement (fonction *ObjetComparable : :operator<*, en bas, au centre), représente environ 17% du travail total du CMC. Cette proportion importante est pourtant minimisée par la mise en place de files d'événements ne nécessitant pas de tris (voir section 4.6).

On observe également sur cette figure que la charge computationnelle utilisée pour la réception des événements (fonction *recevoirPaquet*, au centre) est clairement répartie entre l'ajout d'événements (fonction *ajouteEvenement*) et les tests de terminaison des opérations de réceptions de messages (fonction MPICH : *PMPI_Iprobe*). Ces tests représentent environ 12% de la charge computationnelle totale d'un CMC.

Le reste du temps, le CMC effectue d'autres travaux d'émission ou réception. Dans un programme utilisant des communications bloquantes, ces tests ne seraient pas effectués, mais ils seraient remplacés par une attente passive du *thread* CMC dès lors que l'opération de communication initiée ne serait pas encore terminée.

Enfin, on constate sur cette figure que la préparation des événements émis en paquets spécifiques de chaque LP (fonction (*fileVersPaquets*) représente environ 11% de la charge computationnelle totale utilisée par le CMC. Ceci signifie que les 20 % de la charge computationnelle du CMC restant dans le traitement des événements à émettre (fonction *traiteProchainEvenement*) représentent principalement la charge computationnelle nécessaire aux contrôles précédant l'émission (voir section 4.5).

Au vu de cette répartition de la charge de calcul entre les différentes tâches au sein du CMC, nous pouvons dire qu'un équilibre est atteint puisque les trois principales tâches représentent chacune environ 30% de la charge totale. On note par ailleurs que dans ces trois tâches, les tests de comparaison d'OE (pour trouver les cibles d'un OE) et de terminaison de communications représentent une proportion importante de la charge computationnelle utilisée par le CMC.

5.1.3 Graphe des CPC

Sur la figure 5.2, les principaux appels de fonctions effectués par CPC sont illustrés de la même manière que pour CMC.

La phase de récupération du prochain événement à traiter de l'algorithme 5 du CPC (fonction *getProchainEvenement*) représente environ 30 % de la charge totale du CPC auxquels il faut ajouter les 20 % de verrouillage de la fet (fonction *verrouilleFile*).

Le verrouillage est nécessaire avant l'accès au prochain événement de la fet. En effet, CMC et CPC y ayant accès en concurrence, les *mutex* mis en place permettent d'éviter les erreurs.

Les contrôles effectués pour obtenir l'autorisation de traiter un événement représentent moins de 10% de la charge computationnelle allouée au CPC (fonction *testAutorisationTraitement*).

Les traitements d'événements par les OE à proprement parler (fonction *runRecepEvt*, à gauche sur la figure) représentent, dans cet exemple, moins de 10 % de la charge du CPC. Cette proportion dépend directement du nombre d'événements à traiter, en moyenne sur chaque CPC et de la complexité des modèles de neurone et de synapse utilisés.

La phase d'association d'un événement à sa cible fait appel à une recherche dans le stock d'OE du CPC, ce qui implique des comparaisons entre objets (fonction *operator<*, en bas à gauche de la figure) qui représentent près de 15 % de la charge totale du CPC. Dans l'exemple utilisé ici, les comparaisons sont aussi effectuées par les neurones implémentés dans cet exemple, lorsqu'ils recherchent la dendrite sur laquelle arrive l'événement en cours de traitement.

La phase de récupération des événements après traitement n'est que très peu coûteuse en temps du fait que la création de l'événement est effectuée par l'OE (donc dans la fonction *runRecepEvt*). La simulation présentée a été effectuée sans utilisation de *threads* indépendants pour les traitements des OE. Les temps de récupération effectifs sont négligeables dans ce cas.

5.2 Support matériel

Toutes les simulations effectuées pour cette évaluation sont exécutées sur un réseau de 35 stations de travail. Chaque station est une machine double-cœur à $2,20GHz$ possédant une mémoire vive de $1Go$.

Un réseau à 100 Mégabits/s supporte les communications. Le temps de réponse moyen à un *ping* (message vide permettant de tester l'existence de la connexion et la latence de la réponse) sur le réseau de stations est d'environ $0,150 \times 10^{-3}$ secondes. Les stations fonctionnent sous un système d'exploitation Linux version 2.6.22.9. Ces ordinateurs occupent des salles de Travaux Pratiques d'Informatique des étudiants de l'Université Antilles Guyane et sont à ce titre dotés de systèmes d'exploitation complets, non optimisés pour le calcul distribué. Ceci implique que les machines hébergent d'autres applications (démons de services, interfaces graphiques ...). Par ailleurs, l'ordonnancement des *threads* et des processus est effectué par le système ce qui ne nous permet donc pas une utilisation optimale des machines. Il s'agit donc d'une configuration réaliste, non optimisée.

Dans les simulations, chaque LP sera associé à une station autant que possible.

Les données de simulation récupérées (accès fichiers) et les affichages (accès à l'écran) sont des facteurs influant sur les temps d'exécution. Pour les simulations effectuées dans le cadre de cette étude, aucune donnée de simulation ne sera récupérée, excepté les temps d'exécution eux-mêmes.

La dynamique, la topologie et les modèles de neurones et de synapses des réseaux simulés ont une forte influence sur le temps d'exécution. Nous procéderons à la défi-

nition des réseaux utilisés pour toutes les simulations afin que les comparaisons des temps d'exécution soient effectuées à partir de simulations identiques.

5.3 Protocoles de test

Pour effectuer les simulations, nous considérons des réseaux de neurones impulsionnels dont les modèles sont de type LIF (la définition a été fournie au chapitre 1). Les valeurs des paramètres des neurones utilisés sont fournies en annexe B.

Les réseaux modélisés contiennent 20% de neurones inhibiteurs.

Les stimulations sont appliquées par des PA émis par des cellules d'entrée de l'environnement, à fréquence constante ($600Hz$) de sorte que, sur toute la durée de simulation, l'activité moyenne du réseau est maintenue à une moyenne constante. Deux régimes d'activité moyens sont testés pour les réseaux simulés : 1Hz et 10Hz.

La durée des simulations est de $1s$ de temps biologique et la précision est de l'ordre du dixième de milliseconde. Le délai moyen implémenté dans les connexions neuronales est de $10ms$ et varie selon les connexions jusqu'à plus ou moins $2ms$.

Chaque réseau est simulé sur différents nombres de LP, pour chaque régime d'activité. Le LP_0 est toujours affecté à l'environnement et héberge les cellules d'entrée et de sortie.

La quantité de mémoire vive diponible sur les machines utilisées ($1Go$ par machine) limite le nombre de machines minimum utilisables pour exécuter certaines simulations (stockage du réseau et des événements générés).

Les performances du simulateur DAMNED, en termes de temps d'exécution sur le support matériel utilisé, ont une importance particulière pour la validation du simulateur en tant qu'outil efficace de simulation distribuée de réseaux de neurones impulsionnels et donc son adéquation avec ses objectifs.

En effet, le premier enjeu est de permettre l'exécution de grands réseaux ($> 10^4$). Un autre objectif est alors de permettre une accélération suffisante des simulations de réseaux de grande taille par rapport à une exécution séquentielle (théorique) pour permettre des tests rapidement analysables. Pour chaque simulation effectuée, plusieurs indicateurs temporels sont placés dans le code source afin d'enregistrer les temps désirés.

Des précisions quant au placement exact de ces indicateurs temporels dans le code source sont fournis en annexe 6.8. Les temps sont moyennés sur 5 essais pour chaque réseau.

Dans une première partie, nous présentons la méthode de calcul de l'accélération utilisée. Nous étudions ensuite les temps d'exécution et les temps de création des réseaux de neurones simulés avec DAMNED.

5.4 Calcul de l'accélération

Bien que des réseaux d'une taille de l'ordre de 10^3 neurones avec une connectivité de $0,1$ soient exécutables sur une unique machine, les ressources mémoire nécessaires à l'exécution de simulations de réseaux de 10^4 ou 10^5 neurones, avec une même connectivité, rendent impossibles de telles simulations sur une seule des machines (parmi celles dont nous disposons).

Nous avons donc doté l'une de ces machines du maximum de mémoire adressable sur un processeur 32*bits*, soit 4*Go* afin de permettre la simulation de réseaux allant jusqu'à 10^4 neurones possédant chacun 10^3 synapses. Pour des réseaux de l'ordre de 10^5 neurones il est impossible d'exécuter les simulations sur une machine 32*bits*. De plus l'espace mémoire requis dépendant principalement du nombre de synapses dans le réseau, celui-ci évolue en n^2 où n est le nombre de neurones du réseau. L'utilisation d'une machine 64 bits ne permettrait pas de résoudre le problème.

À l'aide du simulateur DAMNED, l'utilisation de plusieurs machines permet de repousser cette limite et donne accès à des simulations de très grands réseaux.

L'accélération (ou *speed-up*) des temps d'exécution et de création des réseaux est étudiée afin de connaître le gain obtenu par l'utilisation de plusieurs processeurs à l'aide du simulateur DAMNED, en regard d'une simulation séquentielle pour un réseau équivalent, en théorie. En effet, compte tenu du fait que les tailles de réseaux d'intérêt ne sont pas simulables sur une seule machine, nous devons élaborer une extrapolation théorique des temps d'exécution séquentiels pour ces réseaux.

L'accélération, se calcule selon l'équation :

$$A = \frac{T_1}{T_P} \tag{5.1}$$

où T_1 correspond au temps d'exécution du meilleur algorithme séquentiel possible, et T_P correspond au temps d'exécution de l'algorithme parallèle avec P processeurs.

Dans notre cas, le temps de référence séquentiel est obtenu en exécutant le simulateur DAMNED, sur une seule machine, avec un seul LP hébergeant le réseau de neurones et un LP pour l'environnement.

Nous extrapolons les temps d'exécution théoriques sur une machine dont la mémoire vive est portée à 4*Go* pour ces tests, à partir de simulations de réseaux de 1000, 2000, 3000, 4000, 7000 et 10000 neurones. Pour ces tailles de réseaux, la simulation sur une unique machine fournit un temps d'exécution de référence.

Lorsque l'on fait varier la taille du réseau, le temps d'exécution d'une simulation, pour une connectivité et une activité moyenne données, évolue principalement en fonction du nombre de synapses soit en n^2 avec n le nombre de neurones.

Nous recherchons une fonction $f(x) = \alpha \times x^2 + \beta \times x + \gamma$ décrivant l'évolution du temps d'exécution en fonction du nombre de neurones. Nous utilisons la méthode des K-moyennes ($K - means$ (Steinhaus 1956)) à partir des temps d'exécution obtenus.

Les *speed-ups* des temps de simulation seront calculés par rapport aux temps d'exécution théoriques de référence sur une machine (deux cœurs), extrapolés à partir des fonctions obtenues.

Bien que l'étude des temps d'exécution soit indispensable à l'évaluation d'un simulateur, les nombreuses sources de variabilité dans les temps d'exécution en font, malgré tout, un indicateur de performance limité, en particulier dans le cas de machines non-dédiées.

5.5 Temps d'exécution des simulations

L'étude des temps d'exécution est effectuée pour deux catégories de connectivités des réseaux. Dans un premier temps, des réseaux de connectivité $0,1$ où les neurones

auront en moyenne 10% de chances d'avoir une connexion avec chacun des autres neurones. Dans un second temps, une connectivité de 0,01 sera utilisée. Nous distinguerons ces deux catégories par les termes de connectivité « forte » pour 10% et connectivité « faible » pour 1%.

5.5.1 Connectivité forte

Temps séquentiel

Des réseaux de 1 à 10^4 neurones sont exécutés pour établir la relation décrivant l'évolution du temps d'exécution en fonction de la taille du réseau, pour une connectivité de 0,1, sur une seule machine. Deux régimes d'activité sont testés : $1Hz$; et $10Hz$. Des réseaux de 1000, 2000, 3000, 4000 et 7000 neurones sont exécutés avec une activité moyenne de $10Hz$. Des réseaux de 1000, 2000, 3000, 4000, 7000 et 10000 neurones sont exécutés pour une activité moyenne de $1Hz$.

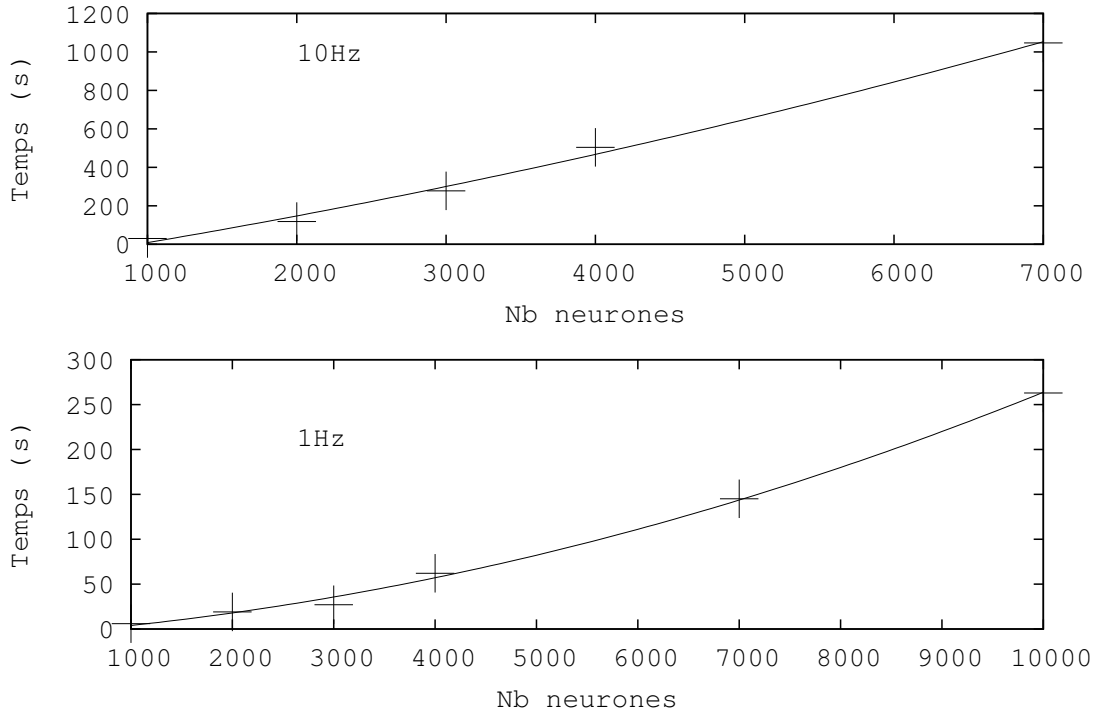

FIG. 5.3 – *Évolution des temps d'exécution sur une machine, pour des réseaux de 1000 à 10000 neurones avec une connectivité de* 0,1. *Les temps d'exécution enregistrés sont indiqués par des croix (en vert). La courbe tracée (en rouge) est obtenue à partir de la fonction* $f(x) = \alpha \times x^2 + \beta \times x + \gamma$. *Les paramètres spécifiques de chaque courbe sont fournis dans le texte. A : l'activité moyenne des réseaux est de* $10Hz$. *B : l'activité moyenne des réseaux est de* $1Hz$.

La figure 5.3, illustre ces temps d'exécution pour des réseaux à $10Hz$ (A, en haut) et $1Hz$ (B, en bas). Les croix indiquent les valeurs obtenues lors des simulations. Les courbes indiquées correspondent aux fonctions :

$$f(x) = \alpha \times x^2 + \beta \times x + \gamma \tag{5.2}$$

où les paramètres ont pour valeur :
- $\alpha = 7,01 \times 10^{-6}$, $\beta = 0,118$ et $\gamma = -116$, lorsque l'activité moyenne est de $10Hz$;
- $\alpha = 1,86 \times 10^{-6}$, $\beta = 8,42 \times 10^{-3}$, $\gamma = -6,33$, lorsque l'activité moyenne est de $1Hz$;

Sur ces courbes, on observe de légères variations autour de la valeur obtenue par le calcul. Les variations des débits dans le réseau physique peuvent expliquer en partie ces variations, compte tenu de l'ordre de grandeur des temps d'exécution. Un réseau de 1000 neurones, émettant à $1Hz$ en moyenne, représente 1000 événements PA et 100000 événements PPS (la connectivité étant de $0,1$). Les régimes d'activité sont stabilisés autour de $1Hz$ (respectivement $10Hz$). Une faible variation dans le nombre de PA émis peut influencer les temps d'exécution à cette échelle. Par exemple, un réseau émettant 1100 PA (respectivement 11000) sur $1s$ de temps biologique simulé, peut être considéré comme émettant à environ $1Hz$ en moyenne. Compte tenu de la connectivité, le nombre d'événements PPS traités passe à 110000 PPS (respectivement 1 100 000), ce qui représente une variation de 10000 PPS (respectivement 100000) dans le traitement, ce qui a donc un fort impact sur le temps d'exécution.

Accélération sur plusieurs machines

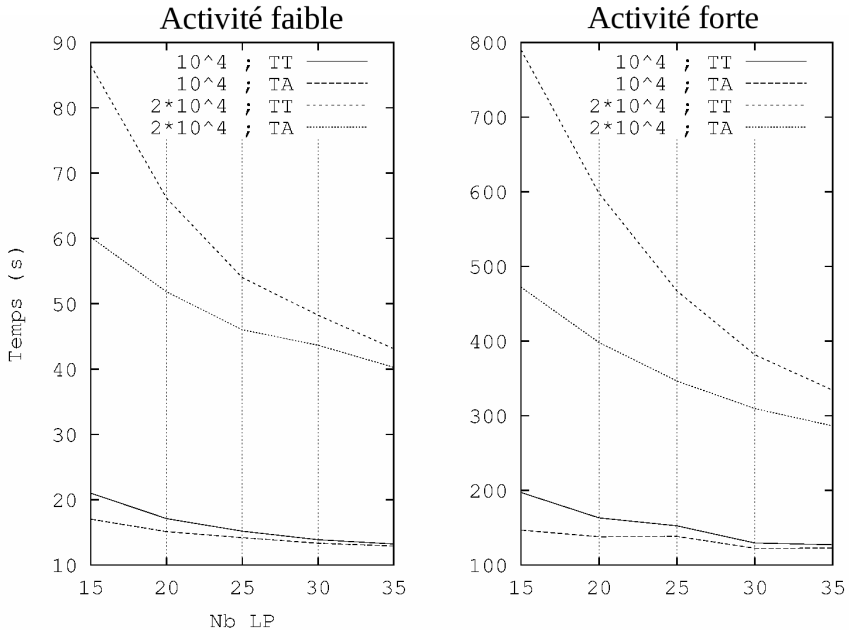

FIG. 5.4 – *Évolution des temps d'exécution, en fonction du nombre de LP, pour des réseaux de 10000 et 20000 neurones. La connectivité est de* $0,1$. *Pour chaque réseau, le temps total d'exécution (TT) ainsi que les temps d'activité (TA) des threads CMC correspondant sont indiqués.*

Deux tailles de réseaux sont implémentées pour étudier l'accélération procurée par le simulateur DAMNED dans les exécutions de réseaux de grande taille fortement connectés.

Un réseau de 10^4 neurones et un réseau de 2×10^4 neurones sont réalisés afin d'étudier l'impact de l'augmentation de la taille du réseau et de l'activité moyenne.

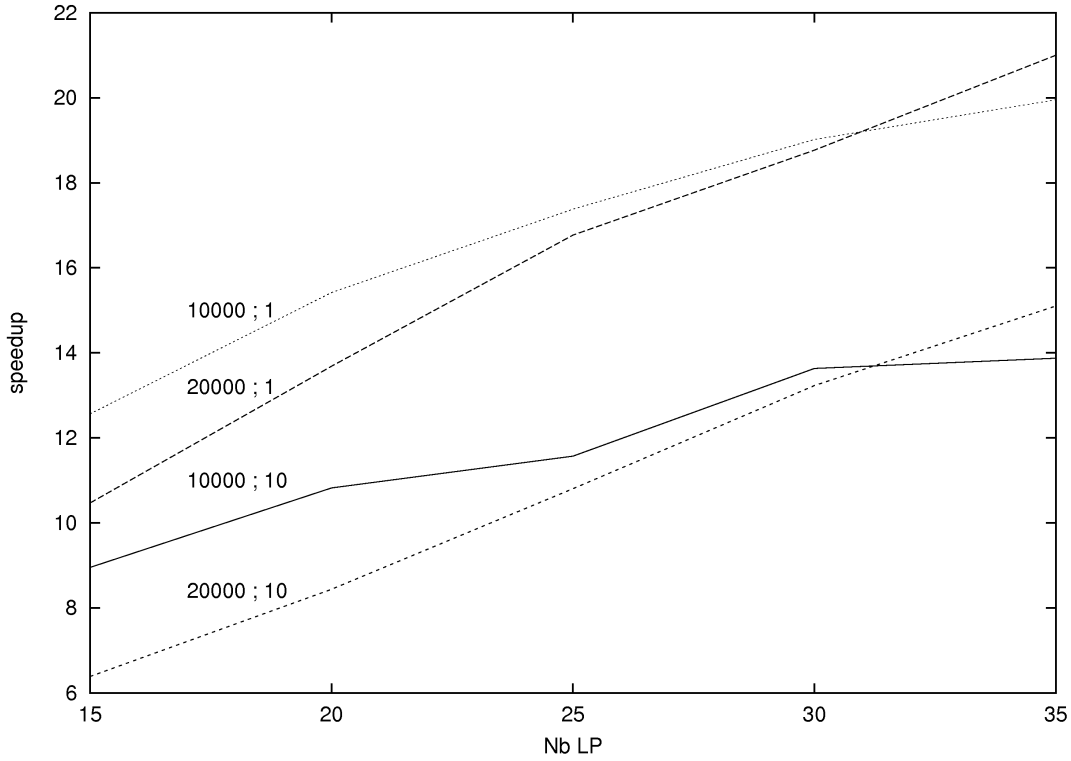

FIG. 5.5 – *Les speed-ups sont illustrés pour des tailles de réseaux de* 10000 *et* 20000 *neurones, à des activités moyennes de* $1Hz$ *et* $10Hz$.

La figure 5.4 illustre l'évolution du temps d'exécution des simulations en fonction du nombre de LP (1 par machine : 2 cœurs, $1Go$ de RAM) utilisés pour la simulation. Les temps d'exécution pour les deux tailles de réseau sont indiqués pour des activités « forte » (à $10Hz$ en moyenne) et « faible » (à $1Hz$ en moyenne).

La figure 5.5 illustre les *speed-ups* en fonction du nombre de LP utilisés (un LP par machine), pour les deux tailles de réseaux et les deux types d'activité.

On constate une accélération du temps d'exécution par l'utilisation de plusieurs machines. Cette accélération est supérieure à 12 en utilisant 30 machines quel que soit le réseau. De plus, on constate que la variation de l'accélération en fonction du nombre de LP est plus rapide pour des réseaux de 2×10^4 neurones que pour des réseaux de 10^4 neurones. Dès que l'on atteint 30 machines, l'accélération obtenue pour un réseau de 2×10^4 neurones devient plus grande que celle obtenue pour un réseau de 10^4 neurones. On s'attend donc a avoir une accélération d'autant plus grande que la taille du réseau est grande.

Pour une taille donnée de réseau, l'augmentation de l'activité du réseau (de $1Hz$ à $10Hz$) diminue l'accélération, mais sa variation en fonction du nombre de LP conserve la même allure.

5.5.2 Connectivité faible

Temps séquentiel

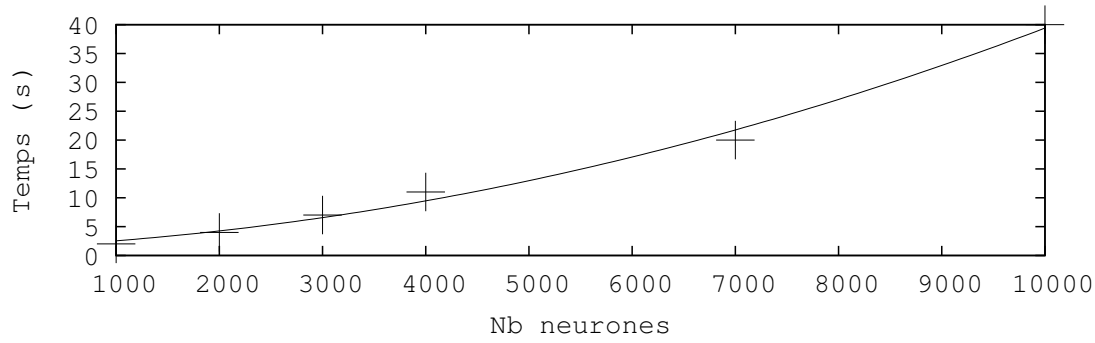

FIG. 5.6 – *Évolution des temps d'exécution sur une machine, pour des réseaux de 1000 à 10000 neurones. Les temps d'exécutions enregistrés sont indiqués par des croix. La courbe est obtenue à partir de l'équation* $f(x) = \alpha \times x^2 + \beta \times x + \gamma$. *Les paramètres sont fournis dans le texte. L'activité des réseaux est de* $1Hz$.

Pour les réseaux de faible connectivité, des réseaux de 10^3 à 10^4 neurones sont exécutés pour établir la relation décrivant l'évolution du temps d'exécution en fonction de la taille du réseau, pour une connectivité de $0,01$, sur une seule machine. Les activités moyennes des réseaux sont de $1Hz$.

L'espace mémoire requis pour l'étude de réseaux de 10^5 neurones à une connectivité de $0,1$, ainsi que pour une activité de $10Hz$, dépasse les ressources mémoire disponibles sur le réseau de stations utilisé. C'est pourquoi, nous ne présenterons que les résultats obtenus pour une connectivité de $0,01$ et une activité de $1Hz$ (faible). Les limites mémoires limitent également à 27 machines le nombre minimum de machines nécessaires pour l'exécution de réseaux de 10^5 neurones de connectivité $0,01$ à un régime de $1Hz$. Afin d'illustrer malgré tout l'évolution des temps d'exécution et de l'accélération pour des réseaux de cette taille, nous présentons deux tailles de réseau : 80×10^4 et 10^5 neurones pour 27, 29, 31 et 35 LP (un par machine).

La figure 5.6, illustre les temps d'exécution des réseaux de 10^3 à 10^4 neurones ayant une activité moyenne de $1Hz$ sur une seule machine. Les croix indiquent les valeurs obtenues lors des simulations. Les paramètres obtenus pour la fonction correspondante sont les suivants : $\alpha = 2,98 \times 10^{-7}$, $\beta = 8,17 \times 10^{-4}$, et $\gamma = 1,43$;

Accélération sur plusieurs machines

Les temps d'exécution des simulations de réseaux de 80×10^4 et 10^5 neurones sont présentés sur la figure 5.7. Le régime d'activité maintenu est de $1Hz$ et la connectivité est de $0,01$. On observe que le temps d'exécution diminue avec l'augmentation du nombre de LP. Le temps d'exécution est divisé par deux en augmentant le nombre de machines de 27 à 35.

Les accélérations obtenues par l'utilisation du simulateur pour les réseaux de 80×10^4 et 10^5 sont illustrées sur la figure 5.8.

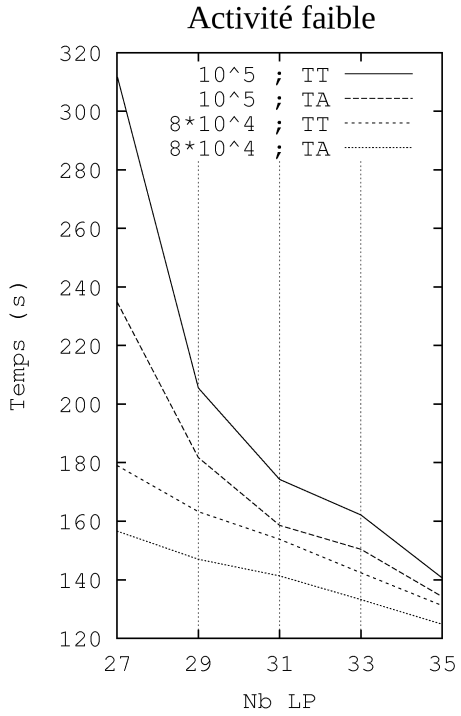

FIG. 5.7 – *Évolution des temps d'exécution, en fonction du nombre de LP, pour des réseaux de 80000 et 100000 neurones. La connectivité est de* $0,01$. *Pour chaque réseau, le temps total d'exécution (TT) ainsi que les temps d'activité (TA) des threads CMC correspondant sont indiqués.*

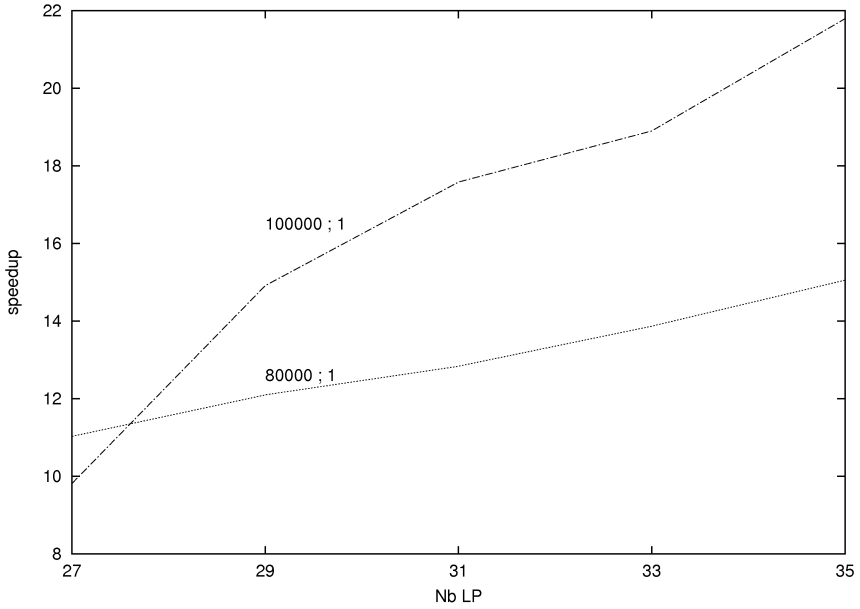

FIG. 5.8 – *Speedups obtenus pour les réseaux de* 8×10^4 *et* 10^5 *neurones avec une connectivité de* $0,01$.

L'influence de la taille du réseau de neurones sur l'accélération observée dans la section précédente se confirme dans ce cas. L'accélération obtenue pour les réseaux

de 10^5 neurones est plus grande que pour les réseaux de 80×10^4 neurones dès l'utilisation de 29 LP.

5.6 Temps de création

La création du réseau de neurones est effectuée à partir du fichier de configuration défini pour l'expérience. Chaque machine récupère les informations contenues dans ce fichier, et crée les neurones à héberger localement, ainsi que leurs connexions.

Les temps de création sont étudiés pour des réseaux de 10^4 avec une connectivité de $0,1$ et des réseaux de 10^5 neurones avec une connectivité de $0,01$.

Pour les raisons exposées à la section 5.5.2, les réseaux de 10^5 neurones avec une connectivité de 0.1 ne peuvent être étudiés à l'aide des machines disponibles.

5.6.1 Temps séquentiel

On procède, comme pour les temps d'exécution, à une extrapolation du temps séquentiel à partir des temps de référence obtenus sur une machine avec un LP contenant le réseau de neurones complet plus un LP pour l'environnement.

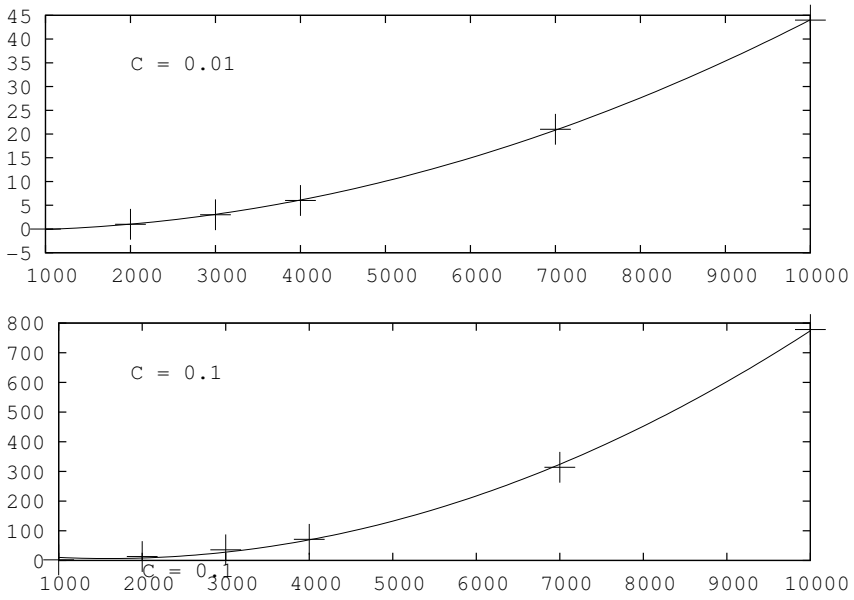

Fig. 5.9 – *Évolution des temps de création des réseaux, sur une machine, en fonction de la taille du réseau. Les temps de création pour des connectivités de* 0.1 *et* 0.01 *sont illustrées.*

Des réseaux de 1000, 2000, 3000, 4000, 7000 et 10000 neurones sont utilisés pour cette extrapolation. À partir des temps de création obtenus pour ces réseaux, sur une seule machine, il est possible d'extrapoler un polynôme du second degré (du type de la fonction 5.2) qui décrive les données et permette de prédire le temps de création séquentiel, en fonction de la taille du réseau. La figure 5.9 présente ces temps de création ainsi que la fonction obtenue à l'aide des paramètres suivants :

– $\alpha = 1,09 \times 10^{-5}$, $\beta = -0,0346$ $\gamma = 34,0$, lorsque la connectivité est de $0,1$;

– $\alpha = 4,74 \times 10^{-7}$, $\beta = 3,02 \times 10^{-4}$, $\gamma = -0,268$, lorsque la connectivité est de $0,01$;

Accélération

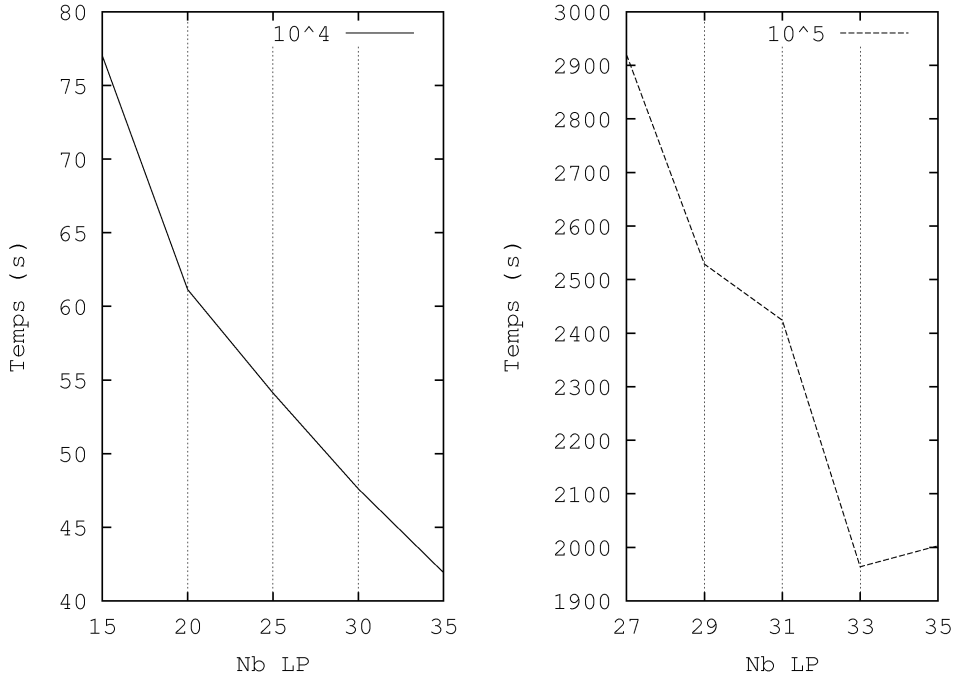

FIG. 5.10 – *Évolution des temps de création des réseaux en fonction du nombre de LP dans la simulation. Les temps de création sont indiqués pour des réseaux de* 10^4 *neurones avec une connectivité de* 0.1*, à gauche et des réseaux de* 10^5 *neurones avec une connectivité de* 0.01*, à droite.*

L'évolution des temps de création des réseaux de 10^4 neurones avec une connectivité de 0.1 et des réseaux de 10^5 neurones avec une connectivité de 0.01 est présentée sur la figure 5.10. La diminution du temps de création des réseaux lors de l'augmentation du nombre de LP est confirmée. Cependant on constate que le temps de création pour les réseaux de 10^4 neurones (connectivité $0,1$) est divisé par deux en augmentant le nombre de LP de 27 à 35 alors que pour le réseau de 10^5 neurones (connectivité $0,01$) n'est diminué que d'un tiers.

À partir des temps séquentiels extrapolés, il est possible de calculer l'accélération des temps de création des réseaux simulés. La figure 5.11 illustre les accélérations obtenues pour ces réseaux.

On constate que l'accélération des temps de création est de l'ordre de 15 pour les réseaux de 10^4 neurones (connectivité $0,1$). L'accélération est de l'ordre de 2 pour les réseaux de 10^5 neurones (connectivité $0,01$). Pour ces deux types de réseaux, l'accélération augmente lorsque le nombre de LP (machine) augmente.

Le nombre de synapses d'un réseau de 10^4 neurones avec une connectivité de $0,1$ est de $10^4 \times 10^3 = 10 \times 10^6$ synapses. Le même nombre de synapses existe dans un réseau de 10^5 neurones avec une connectivité de $0,01$. Ainsi, ce n'est pas le nombre de synapses du réseau qui influe ici sur le temps de création.

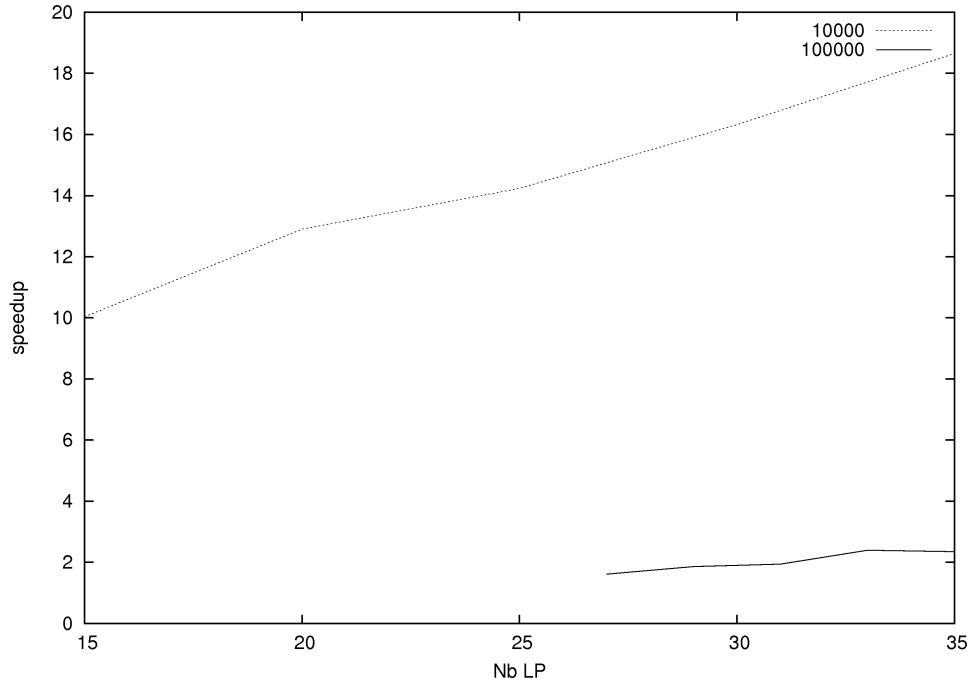

FIG. 5.11 – *L'accélération des temps de création des réseaux de* 10^4 *neurones de connectivité* 0.1 *et un réseau de* 10^5 *neurones de connectivité* 0.01 *sont présentés en fonction du nombre de LP utilisés.*

Cependant, au niveau d'un LP, la taille des tables de connexions et de propriétés des threads CMC (voir section 4.3.4, page 76) dépend étroitement du nombre de neurones « connus » par le LP. Cela dépend donc du nombre de neurones du réseau ayant des connexions post-synaptiques ou pré-synaptiques avec des neurones locaux. Dans un réseau de 10^4 (respectivement 10^5) neurones où chaque LP héberge n neurones, le nombre maximum de neurones connus par chaque LP est donc de n*10^3 (respectivement n*10^4) si chaque connexion (post ou pré) de chaque neurone se fait avec un neurone différent. Dans le cas des réseaux de 10^4 neurones, le nombre de neurones connus est borné par 10^4 alors que pour les réseaux de 10^5 ce nombre est borné par 10^5.

Une étude plus détaillée du mapping des réseaux sur les LP et des tailles des tables des CMC obtenues pourrait renseigner sur l'influence de ce paramètre sur l'évolution des temps de création et leur accélération.

5.7 Discussion

Le simulateur DAMNED permet la simulation de grands réseaux de neurones impulsionnels en tirant parti de supports matériels distribués et du multithreading. Dans le cadre de l'évaluation du simulateur, des réseaux de 10^3 à 10^5 neurones impulsionnels avec un nombre de connexions allant de 10^4 à 10^8 ont été réalisés pour des activités de $1Hz$ et $10Hz$ représentant de 10^3 PA et 10^4 PPS (pour un réseau de 10^3 neurones avec une connectivité de $0,01$ et émettant à $1Hz$) à 10^5 PA et 10^8 PPS (pour un réseau

de 10^5 neurones avec une connectivité de $0,1$ émettant à $1Hz$) générés au cours de la simulation (qui dure une seconde).

Le manque de mémoire vive rend impossible la simulation de grands réseaux (e.g. 10^5 neurones) sur une seule machine. Ce travail montre tout d'abord que le simulateur DAMNED permet de réaliser d'augmenter les tailles des réseaux implémentables grâce à des ressources matérielles distribuées.

D'autre part, l'utilisation du simulateur et de plusieurs machines accélère les temps d'exécutions. Il apparaît également que l'accélération augmente avec le nombre de machines utilisées.

Impact de la taille du réseau

Cette augmentation est d'autant plus forte que le réseau est étendu (voir figures 5.5 et 5.8), à nombre de machines, activité et connectivité identiques. Nous avons vu à la section 5.1.3 (page 96) que la récherche des neurones cibles dans le stock d'OE occupe une part importante des traitements réalisés par les *threads* CPC. Lorsque le nombre de neurones stockés augmente, les recherches d'OE sont plus longues et ainsi, le recouvrement entre les communications et le traitement (recouvrement du travail de CMC et CPC) est potentiellement plus grand. L'augmentation du nombre de LP pour effectuer l'exécution a donc un impact plus fort pour des réseaux plus grands : à 25 LP un réseau de 10^4 est accéléré d'un facteur $17,4$ tandis qu'un réseau de 2×10^4 est accéléré de $16,8$. À 35 LP, les mêmes réseaux sont accélérés d'un facteur $19,9$ (pour 10^4) et 21.0 (pour 2×10^4).

Ceci s'explique par le fait que l'augmentation des communications due à l'augmentation du nombre de LP est d'autant mieux compensée par les traitements que le nombre de neurones hébergés sur chaque LP est grand. Parmi les traitements, le temps de recherche des neurones dans les stocks d'OE locaux peut expliquer en partie cette « meilleure » compensation lors de l'augmentation de la taille du réseau.

Impact de la dynamique du réseau

Par ailleurs, la dynamique des réseaux a également une influence sur les temps d'exécution. On a constaté (voir figure 5.5) que, à taille de réseau constante, l'accélération des temps de simulation est d'autant plus forte que l'activité est faible. La variation de l'accélération des temps d'exécution, lors de l'augmentation du nombre de processeurs, reste cependant similaire pour des activités forte ou faible (la pente de l'accélération reste d'autant plus forte que la taille du réseau est grande). Pour une taille de réseau fixe, augmenter le taux de décharge moyen des neurones augmente le nombre d'événements traités. Cette augmentation est répercutée sur la charge de traitement d'événements par les *threads* CPC et sur la charge de communications effectuées par les *threads* CMC. Si l'augmentation du nombre d'événements diminue l'accélération, cela signifie donc que l'impact sur l'augmentation des temps de traitement est plus faible que l'impact sur l'augmentation des temps de communication. Il existe donc une borne du taux de décharge moyen maximum pour lequel l'utilisation du simulateur DAMNED permet d'accélérer les temps de simulation. Compte tenu des résultats obtenus, cette borne de la fréquence moyenne de décharge devrait être d'autant plus grande que la taille du réseau est grande.

Par ailleurs, une fréquence de décharge moyenne de $10Hz$, sur tout le réseau est comparable à une moyenne biologiquement plausible.

Le débit des réseaux de communication utilisés a, dans ce cas, un rôle très important. Dans cette étude, les réseaux sont de type Ethernet, avec un débit de $100Mbits$ est les machines sont centralisées sur un switch.

L'accélération maximum obtenue lors de cette évaluation des temps d'exécution est de 21.8 pour un réseau de 10^5 neurones ayant chacun 10^3 synapses et émettant en moyenne à $1Hz$, exécuté avec 35 LP, où chaque LP est hébergé par une machine (2 cœurs, $1Go$ de RAM).

Les temps de création des réseaux sont également accélérés par le simulateur lors de l'utilisation de plusieurs LP comme le montre la figure 5.10. Cependant, l'accélération est limitée pour les réseaux de 10^4 neurones et devient médiocre pour les réseaux de 10^5 neurones puisqu'on obtient seulement une accélération de 2 avec 35 LP.

Les méthodes de création des réseaux n'ont pas encore été optimisées. L'ensemble des connexions potentielles du réseau sont testées sur chaque LP afin de déterminer quelles connexions il doit créer localement. Une première optimisation pourrait limiter, sur chaque LP, le parcours des connexions à celles qui seront utiles localement.

5.8 Conclusion

Dans ce chapitre, nous avons montré que le simulateur DAMNED ouvre l'accès aux simulations de grands réseaux de neurones à moindre frais. Un réseau de stations quelconque suffit pour être en mesure de réaliser de grand réseaux. La barrière de la mémoire n'en est donc plus réellement une.

Nous avons également montré que les temps d'exécution des simulations sont fortement accélérés lors de l'utilisation du simulateur DAMNED et de plusieurs machines.

La faible accélération observée pour les temps de création des réseaux les plus étendus montre que la parallélisation de la création n'est pas optimale dans l'état actuel. Cependant, une charge mémoire supplémentaire est nécessaire au simulateur. Cette charge est d'autant plus lourde qu'il y a de machines dans la simulation. De plus la topologie des réseaux et le mapping sur les machiens ont une forte influence sur la quantité de mémoire nécessaire pour stocker les sous réseaux. Les connexions entre deux neurones hébergés pas des LP différents sont plus lourdes en termes de mémoire que les connexions locales.

Les tests effectués sur différentes tailles de réseaux, différentes connectivités ainsi que différents régimes d'activités ont permis de montrer que le simulateur DAMNED offre un support efficace pour la simulation de grands réseaux de neurones impulsionnels.

Les méthodes de gestion distribuée du temps virtuel et la stratégie événementielle de simulation présentées au chapitre précédent, montrent ici qu'elles ne provoquent pas de goulot d'étranglement qui empêcherait l'augmentation du nombre de machines.

Ce travail de thèse apporte donc à la communauté le premier simulateur distribué entièrement événementiel pour des réseaux de neurones impulsionnels, pour une utilisation libre par la communauté scientifique.

Un modèle connexionniste du système saccadique

6

Le chapitre 5 nous a permis d'apporter des précisions sur les performances du simulateur DAMNED. Mais au-delà des temps d'exécution bruts, la première vocation du simulateur est de permettre l'utilisation des ressources matérielles disponibles afin d'effectuer des simulations permettant l'étude de réseaux de neurones impulsionnels. La meilleure façon d'en illustrer les aptitudes passe par l'utilisation du simulateur pour l'étude d'une modélisation connexionniste biologiquement plausible.

Le projet *A*NR MAPS : « Mapping, Adaptation, Plasticity and Spatial Computation », mené en collaboration par quatre partenaires que sont :

- Le Laboratoire d'InfoRmatique en Image et Systèmes d'Information (LIRIS, UMR-CNRS 5205) de Lyon
- Le LORIA (Institut National de Recherche en Informatique et en Automatique (INRIA) Lorraine), de Nancy
- Le laboratoire Mouvement et Perception (UMR-CNRS 6152) de Marseille
- L'Institut de Neurosciences Cognitives de la Méditerranée (INCM, UMR-CNRS 6193) de Marseille,

a donné lieu à la réalisation d'un modèle de contrôle oculomoteur sous-cortical, entièrement connexionniste, directement inspiré de données biologiques.

Dans un premier temps, une rapide présentation du système oculomoteur ainsi que certaines hypothèses sur son fonctionnement nous permettront de proposer un modèle de contrôle saccadique sous-cortical basé à la fois sur la neuroanatomie et sur la neurophysiologie des structures étudiées. Nous testerons alors les aptitudes du modèle à reproduire le comportement de populations de neurones biologiques enregistrées lors de microstimulations électriques. Plusieurs propriétés des structures sous-corticales étudiées seront retrouvées dans le modèle proposé.

6.1 Système oculomoteur

La rétine n'est pas une structure homogène : son centre, la fovéa, permet de prélever des informations visuelles avec une résolution beaucoup plus importante que la zone périphérique. Ainsi, au cours de l'évolution, en même temps que se développait ce capteur visuel, la motricité des globes oculaires se développait également afin de

pouvoir déplacer et aligner cette zone à forte résolution avec les objets d'intérêt à analyser finement.

Le système oculomoteur est le système sensori-moteur qui permet de déplacer les globes oculaires de façon à stabiliser l'image sur la rétine - mouvements oculaires de compensation - ou à orienter la fovéa vers les objets d'intérêt - mouvements oculaires d'orientation. Notre intérêt se portera dans le présent travail sur le contrôle des mouvements d'orientation, et particulièrement sur ceux appelés « saccades oculaires ». Les saccades sont les déplacements très rapides et très précis que les globes oculaires réalisent pour changer le point de fixation du regard. Ce sont par exemple les petits mouvements brusques des yeux que l'on réalise lorsqu'on lit un texte. L'être humain réalise en moyenne de 1 à 3 saccades par seconde. Si l'on considère seulement les saccades oculaires réalisées sans mouvement de la tête, la gamme d'amplitude de ces mouvements est de 1 à 40° environ. Leur durée est de 30 à 100ms et la vitesse maximale atteinte lors de ces déplacements peut aller jusqu'à 800° par seconde.

Aux niveaux psychologique et neurophysiologique, le système de production des commandes motrices pour réaliser des saccades en réponse à des stimuli visuels est, parmi les systèmes sensori-moteurs, celui sur lequel nous avons le plus d'informations (Findlay & Walker 1999). En particulier, en neurophysiologie, de très nombreuses données sont disponibles sur l'organisation anatomique et sur le fonctionnement physiologique des réseaux nerveux du système saccadique (Büttner & Büttner-Ennever 2006).

Malgré cette richesse au niveau des informations disponibles sur le fonctionnement du système saccadique, il persiste encore de nombreuses zones d'ombre sur le fonctionnement fin de ce système. Vu la complexité des interactions régnant entre les diverses structures neuronales en jeu, l'approche modélisation dans ce domaine de recherche est très importante. De nombreux modèles du système saccadique ont déjà été réalisés (voir pour revue Girard & Berthoz 2005).

6.1.1 Neurophysiologie du système saccadique

Nous offrons ici une brève présentation (anatomique et physiologique) du système saccadique afin d'introduire par la suite notre travail de modélisation.

Nous limiterons notre modélisation, et donc notre description, aux circuits localisés dans le tronc cérébral.

Motoneurones et muscles extra-oculaires

En partant de la périphérie : chaque globe oculaire est mobilisé par 3 paires de petit muscles, appelés « muscles extra-oculaires » . Grossièrement, une paire de muscles gère la dimension horizontale, une deuxième paire gère la dimension verticale et une troisième gère une dimension torsionnelle (voir figure 6.1).

La contraction ou le relâchement de ces muscles sont contrôlés par des motoneurones localisés dans trois noyaux du tronc cérébral : le noyau abducens, le noyau trochléaire et le noyau oculomoteur commun.

L'activité des MotoNeurones (MN) lors de la production d'une saccade correspond à un « *pulse-step* » . Le *pulse* est une phase haute fréquence qui a pour but de vaincre les forces de résistance au mouvement (visco-élasticité) et de déplacer l'œil

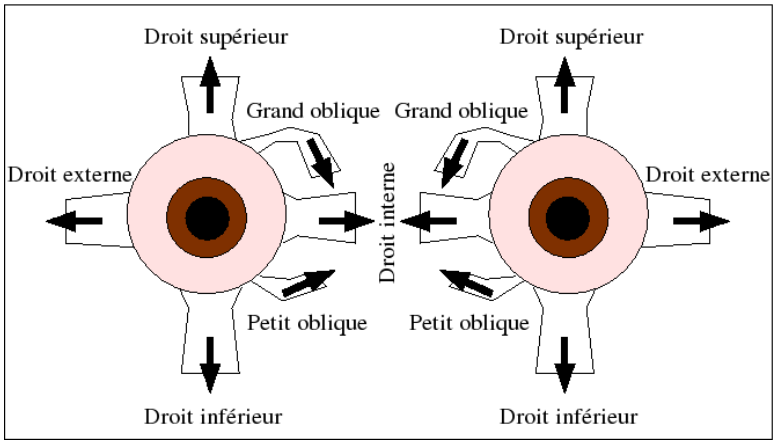

FIG. 6.1 – *Les mouvements de l'œil sont dirigés par 3 paires de muscles : Droit Supérieur, Droit Inférieur et Droit Interne, Droit Externe, Grand Oblique et Petit Oblique.*

très rapidement. Le *step* est une activité tonique permettant de maintenir, après son déplacement saccadique, l'œil en position excentrée.

Pour produire cette activité *pulse-step*, les MN reçoivent les axones de deux types de neurones : des neurones qui déchargent en bouffée (EBN pour *Excitatory Burst Neuron*) [production du *pulse*] et des neurones dont l'activité est tonique (TN pour *tonic neuron*) [production du *step*].

Codage temporel au niveau des EBN

Les caractéristiques du *burst* EBN sont fonction de l'amplitude de la saccade à réaliser. En fait, le profil de cette décharge est très proche du profil de vitesse de l'œil lors d'une saccade oculaire.

Les TN ont une activité dont la fréquence dépend de la position de l'œil. La fréquence de cette activité est obtenue par une intégration, au sens mathématique du terme, de l'activité des EBN. L'activité des EBN est reliée à la vitesse du déplacement (Van Gisbergen et al. 1981).

Nous comprenons donc que le contrôle de la décharge des EBN permet le contrôle de l'amplitude des saccades. À ce niveau, le codage de l'amplitude des saccades est un codage temporel (la durée et la fréquence de décharge influent sur l'amplitude des saccades). L'ensemble des EBN est activé pour chaque saccade.

Codage spatial au niveau du colliculus

Les EBN reçoivent des projections en provenance d'une structure neuronale appelée Colliculus Supérieur (CS) qui forme l'étage en amont du système. Au niveau du CS, le codage de l'amplitude du mouvement à réaliser est spatial. En effet le CS est organisé en carte motrice. Le CS est une structure plane contenant plusieurs couches de neurones. Ce sont les couches intermédiaires et profondes qui forment la carte motrice. À ce niveau c'est la localisation d'une population de neurones activés qui va déterminer l'amplitude (et la direction) du mouvement selon le mapping de la figure 6.2. Lorsque l'activité est localisée vers le pôle rostral (vers la gauche sur la

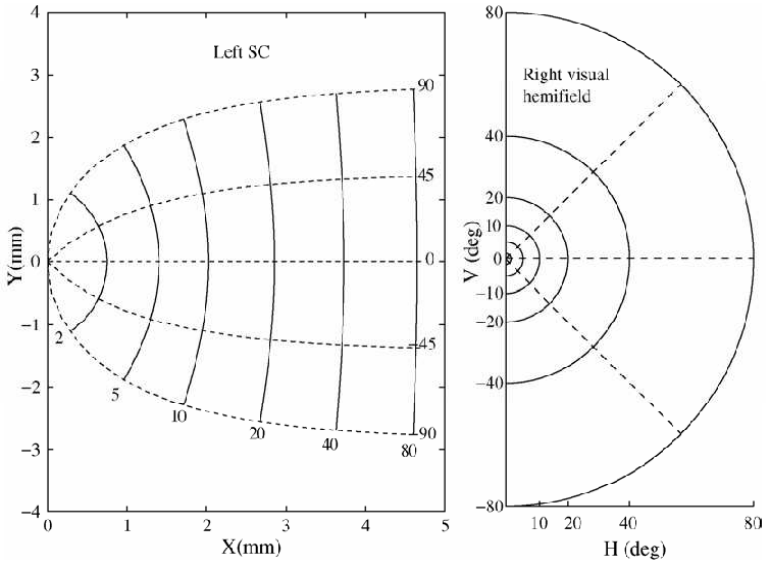

FIG. 6.2 – *D'après (Girard & Berthoz 2005), correspondance entre la carte motrice en coordonnées cartésiennes à gauche, et le champ visuel en coordonnées polaires. Le colliculus gauche et l'hémichamp visuel droit sont présentés sur cette figure.*

figure), la saccade produite est de faible amplitude. Plus la population se déplace vers le pôle caudal (à droite), plus l'amplitude de la saccade augmente. Ensuite si la population est localisée de façon médiane sur la structure, la saccade sera dirigée vers le haut. Elle sera dirigée vers le bas lorsque l'activité sera localisée de façon latérale sur le CS.

D'un codage spatial vers un codage temporel

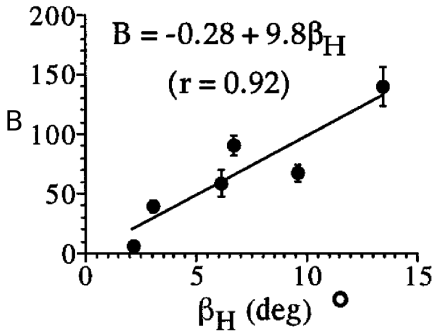

FIG. 6.3 – *D'après (Moschovakis et al. 1998), relation entre le nombre B de boutons synaptiques sur les EBN et l'amplitude des saccades β_H. La relation entre B et β_H est fournie par la droite de régression dont l'équation est insérée sur la figure, r étant le coefficient de corrélation.*

Ainsi, entre l'étage CS et l'étage EBN, une transformation spatio-temporelle doit avoir lieu.

Cette transformation repose en partie sur l'organisation des projections du CS vers les EBN : plus on recule sur la carte motrice (en direction du pôle caudal), plus le nombre de boutons synaptiques formés par les neurones colliculaires sur les EBN

est important (Moschovakis et al. 1998). La figure 6.3 présente la relation entre le nombre de boutons synaptiques dans la zone de la formation réticulée contenant les EBN et l'amplitude de la saccade effectuée lorsque la zone correspondante du CS est stimulée. Toutefois, un autre élément contribue également au contrôle de l'amplitude des saccades : une boucle de rétroaction.

6.1.2 Boucle de rétroaction pour le contrôle de l'amplitude

Il a été proposé assez tôt (Robinson 1975) et démontré par la suite (Zee et al. 1976, Jurgens et al. 1981, Barton et al. 2003) que la production des saccades mettait en jeu le fonctionnement d'une boucle de rétroaction. Un signal de déplacement désiré est généré à partir de la perception de la position de la cible. La saccade va être lancée et tout au long du mouvement le déplacement déjà effectué va être comparé au déplacement désiré, produisant une erreur motrice (ce qu'il reste à parcourir). La saccade s'arrêtera lorsque l'erreur motrice sera nulle.

Position du colliculus dans la boucle de rétroaction

A l'heure actuelle, il est bien établi que le signal de déplacement désiré est présent au niveau du CS. Le déplacement déjà effectué serait obtenu en prenant une copie de ce qui sort des EBN pour aller vers les MN et en intégrant cette copie. Le point de comparaison entre déplacement désiré et déplacement déjà effectué reste plus problématique. Pendant un temps, il a été proposé que cette comparaison soit effectuée par le CS (ce qui plaçait le CS dans la boucle de rétroaction) (Voir Girard & Berthoz 2005, pour revue). La tendance actuelle est plutôt de faire se refermer cette boucle en aval du CS (Kato et al. 2006).

La cMRF dans la boucle de rétroaction

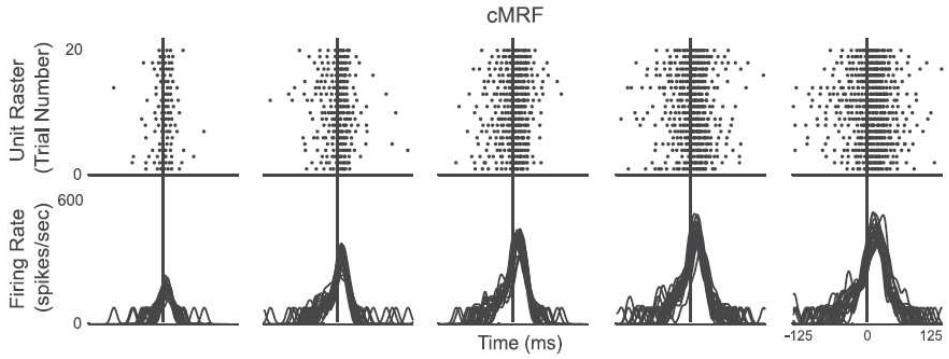

FIG. 6.4 – *D'après(Cromer & Waitzman 2007), exemples de profils de décharge pour les neurones de la cMRF lors de la production d'une saccade. En haut se trouve le raster des émissions de PA et en bas se trouve la fréquence de décharge instantanée des neurones. L'initiation de la saccade est indiquée par la ligne verticale. De gauche à droite se trouvent les réponses obtenues pour des saccades d'amplitudes de plus en plus grandes.*

Plusieurs études ont récemment permis de caractériser l'activité d'une zone particulière du tronc cérébral appelée « Formation Réticulée Mésencéphalique centrale »

(cMRF). Cette zone pourrait être le point de comparaison entre « déplacement désiré » et « déplacement déjà effectué ». En effet une catégorie de neurones localisés dans la cMRF montre une décroissance d'activité très bien corrélée à la décroissance de l'erreur rétinienne (Waitzman et al. 1996). La figure 6.4 illustre le profil de décharge des neurones de la cMRF lors d'une saccade.

Les neurones omnipause

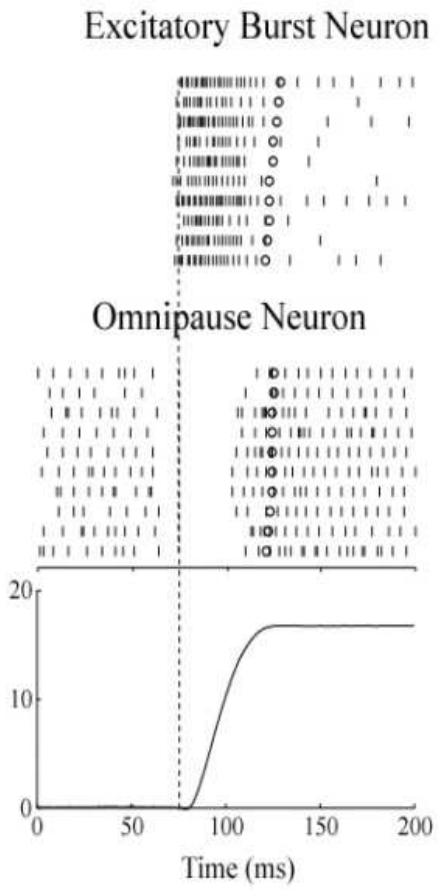

FIG. 6.5 – *Exemple de profils de décharge pour les neurones OPN et EBN lors d'une saccade.*

Enfin, il nous faut présenter un dernier type de neurones très important pour le contrôle des saccades. Il s'agit de neurones inhibiteurs qui influencent très puissamment les EBN. En fait ces neurones sont actifs de façon tonique (voir figure 6.5) et ils montrent une pause dans leur activité pour chaque saccade. Étant donné qu'il n'y a qu'une seule population de tels neurones, qui font une pause quelle que soit l'amplitude et la direction des saccades, ils ont été dénommés Neurones OmniPause (OPN). Ces neurones pourraient faire partie de la boucle de rétroaction décrite plus haut. Ces neurones forment une sorte de porte qui s'ouvre lorsqu'une saccade doit être produite. Par ailleurs leur action inhibitrice très forte sur les EBN entre les sac-

cades permettrait de favoriser la fixation en évitant la perturbation de l'activité des montoneurones par le bruit qui pourrait subsister dans le système.

6.2 Inspiration biologique du modèle et hypothèses de travail

6.2.1 Inspiration biologique

Guillaume & Pelisson (2001) ont étudié le fonctionnement du CS en réalisant des microstimulations électriques dans les couches intermédiaires-profondes de cette structure. Ce travail a été réalisé chez le chat en condition « tête libre ». Les mouvements des yeux et de la tête, évoqués par les microstimulations, ont été enregistrés. Ces auteurs ont caractérisé les influences de la position de la stimulation sur la carte colliculaire, de l'intensité de la stimulation et de la fréquence de la stimulation, sur l'amplitude des saccades évoquées. Ils observent que la position de la stimulation sur la carte « code » pour l'amplitude de la saccade avec une relation non-linéaire (voir figure 6.6 A). Une position rostrale entraînera des saccades de faible amplitude lorsqu'une stimulation caudale entraînera des saccades de grande amplitude. Ce travail met également à jour le fait que l'intensité de la stimulation, et donc la taille de la population activée dans le CS, a une influence sur l'amplitude des saccades et que cette influence est d'autant plus grande que la position de stimulation est caudale (ceci provenant de la non-linéarité de la carte). Ce travail montre également qu'une plage de fréquence de stimulation particulière (300-500 Hz) permet d'obtenir les plus grandes amplitudes de saccade.

L'effet de l'intensité du courant est largement discuté dans cette étude. Les auteurs proposent que la position et la taille de l'activation colliculaire code pour l'amplitude de la saccade. Dès lors, pour que la notion de carte topographique soit conservée, les tailles de population qui s'activent naturellement doivent être très contrôlées. Mais surtout cette proposition permet de mieux comprendre comment les neurones colliculaires pourraient agir sur les EBN, comment leur activité sera traitée dans la formation réticulée (ceci sera présenté dans la section qui suit). Cette meilleure compréhension permet également, dans le présent travail, de proposer une modélisation de ce système, uniquement à l'aide d'entités très proches de la biologie, les neurones impulsionnels.

Cette modélisation nous permet d'aborder les questions suivantes :

- Les propositions de Guillaume & Pelisson (2001) sont-elles viables et peut-on construire un modèle reproduisant les activités biologiques à partir de ces dernières ?
- L'amplitude des saccades produites peut-elle dépendre de la taille de la population activée dans le CS ?

Le travail de modélisation présenté ici est basé sur les protocoles de microstimulations électriques effectués par Guillaume & Pelisson (2001).

6.2.2 Traitement de l'information colliculaire

Deux hypothèses ont été proposées pour décrire le traitement de l'activité colliculaire effectué par le tronc cérébral : l'hypothèse « vector sum » (McIlwain 1976, Sparks et al. 1976), et l'hypothèse « vector average » (Lee et al. 1988, Hanes & Wurtz 2001).

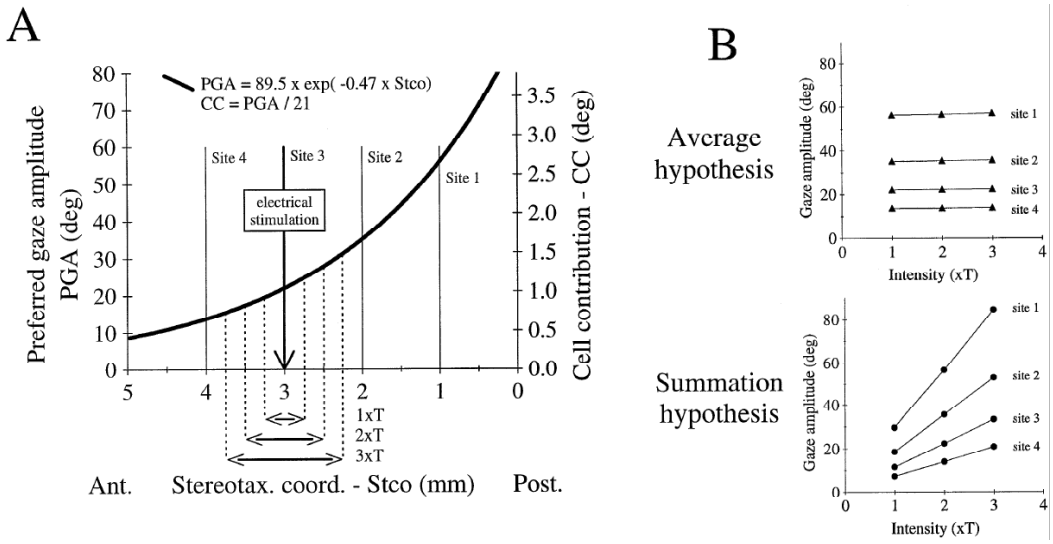

FIG. 6.6 – *D'après (Guillaume & Pelisson 2001). Effets de la variation de la taille de la population active du CS sur l'amplitude des saccades selon les deux hypothèses : sommation et moyennage. A, la carte colliculaire unidimensionnelle est orientée du rostral (Ant.) vers le caudal (Post.). L'amplitude observée (PGA : Preferred gaze amplitude) par Guillaume & Pelisson (2001) a une évolution non-linéaire en fonction de la position selon la relation indiquée en insert. CC est la contribution par cellule dans le modèle. La taille de la population activée par la variation de l'intensité est indiquée par les doubles flèches. B, amplitude de la saccade en fonction de l'intensité de la stimulation (taille de la population activée). T est l'intensité minimum nécessaire pour obtenir une saccade.*

Chaque neurone actif dans le CS code pour un vecteur de déplacement. Lors d'une saccade, une sous-population du CS est active (20% environ, en conditions naturelles). Selon la première hypothèse (sommation), l'amplitude de la saccade est « calculée » par sommation des vecteurs correspondants à partir des coordonnées des neurones de la population active sur la carte motrice de la figure 6.2. Selon la seconde hypothèse, il s'agirait d'un moyennage de ces vecteurs.

La figure **??** illustre les prédictions de ces deux hypothèses sur la manière dont la variation de la taille de la population du colliculus activée influence l'amplitude des saccades. Ces prédictions sont obtenues en fonction des données recueillies par Guillaume & Pelisson (2001).

Les deux propositions concernant l'interprétation des efférences colliculaires par les structures en aval (*vector sum* or *vector average*) conduisent à des prédictions différentes concernant l'effet de la variation de la taille de la population activée sur l'amplitude de la saccade (figure 6.6).

L'hypothèse de sommation fournit des prédictions proches des données obtenues par Guillaume & Pelisson (2001) quant à la variation de l'amplitude des saccades.

6.2.3 La boucle de rétroaction

Le modèle proposé dans notre travail permettra également d'étudier la possibilité d'une boucle de rétroaction excluant le colliculus pour la génération des saccades. L'activité colliculaire étant maintenue par stimulation artificielle, l'arrêt de la saccade ne doit pas être dépendant de l'arrêt de l'activité dans le CS. Le phénomène de « *stair-*

case » saccades observé lors de stimulations prolongées (Robinson 1972) montre que si la stimulation persiste, la saccade se termine quand même effectivement, puis une seconde saccade est initiée dont l'amplitude est souvent moindre. Un modèle montrant ce type de comportement permettrait de confirmer la plausibilité d'une telle boucle.

6.3 Un modèle colliculaire impulsionnel simple

De nombreux modèles de contrôle oculomoteur ont été développés afin de tester les hypothèses avancées. La plupart de ces modèles sont basés sur des modules fonctionnels interagissant de manière prédéfinie (par exemple Badler & Keller 2002, Arai & Keller 2005). Quelques rares modèles sont partiellement basés sur des réseaux de neurones (par exemple Gancarz & Grossberg 1998, mais ce modèle ne tient pas compte du colliculus). Ces modèles tentent alors de reproduire les comportements observés puis éventuellement de prédire certaines conséquences possibles.

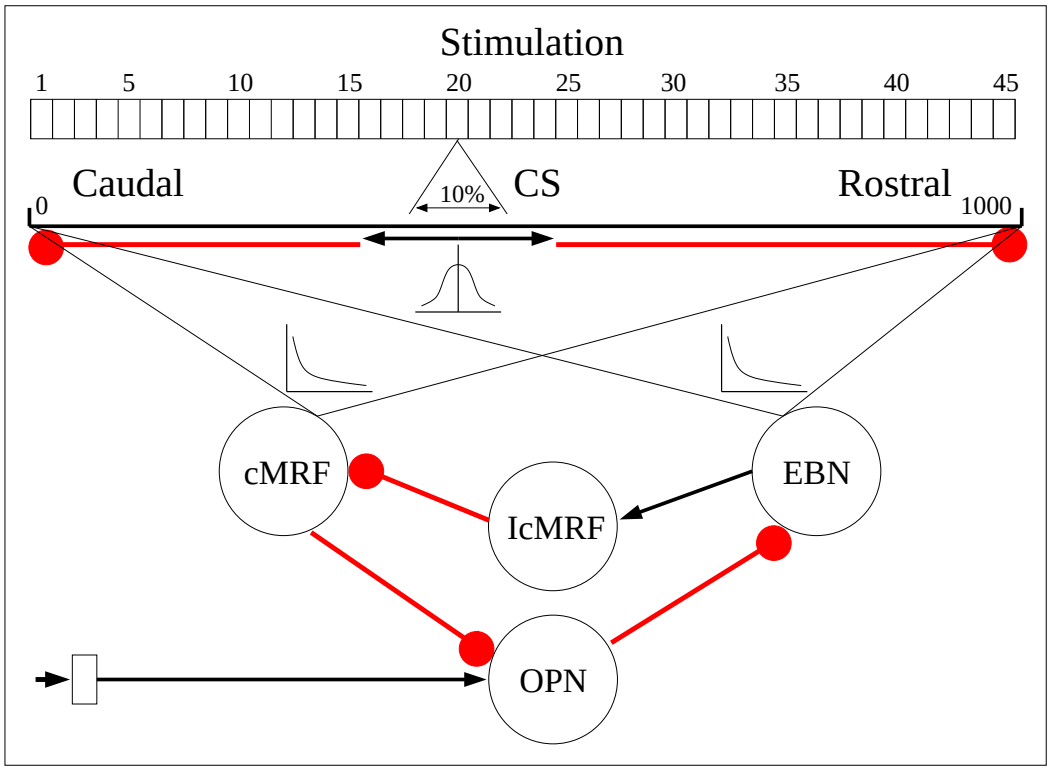

Fig. 6.7 – *Le modèle choisi pour les simulations contient : 1 population de neurones du Colliculus Supérieur (CS), 1 population pour les neurones omnipause (OPN), 1 population pour les EBN, qui constitue la sortie du réseau, et 1 population représentant la formation réticulée (cMRF). On y ajoute une population de neurones inhibiteurs, permettant aux EBN d'inhiber la cMRF, et une seconde pour les inhibitions de la carte colliculaire. Une population de cellules est définie pour stimuler le CS. Le CS est une carte unidimensionnelle de 1000 neurones organisés du pôle Caudal (0) au pôle Rostral (1000).*

Le travail de modélisation proposé ici, dans le cadre du projet ANR MAPS, et réalisé en collaboration avec Alain Guillaume et Emmanuel Daucé, consiste en un modèle de contrôle oculomoteur dans lequel la boucle de rétroaction permettant l'ar-

rêt des saccades, en condition de stimulation artificielle, ne passe pas par le colliculus. La boucle proposée repose sur les neurones ayant une activité de type *burst* durant la saccade (EBN), les neurones omnipause (OPN) et une population de neurones de la Formation Réticulée Mésencéphalique centrale (cMRF). Cette boucle se referme donc « en-dessous » du CS.

Le modèle proposé est entièrement constitué de neurones impulsionnels de type LIF (voir section 1.3, et en annexe 6.8) dont les spécifications sont fournies en annexe. Les simulations sont effectuées à l'aide du simulateur DAMNED développé dans cette thèse (voir chapitre 4). La topologie précise du réseau et les protocoles d'expérience utilisés sont expliqués dans les sections suivantes. Le fichier de configuration (DAMNED) correspondant aux spécifications énoncées dans ces sections est présenté en annexe.

Des tests préliminaires (non présentés) du modèle avec un unique neurone par population ont montré que le déroulement de la boucle EBN - cMRF- OPN est très sensible aux variations des poids synaptiques, des délais de transmission et de la temporalité des stimulations. Nous avons ensuite fait évoluer le modèle vers un modèle à populations dont les caractéristiques sont présentées dans les sections suivantes.

6.3.1 La carte du Colliculus Supérieur

Le modèle proposé, illustré par la figure 6.7, contient une population de neurones colliculaires associés à la carte motrice (couches intermédiaires et profondes) du Colliculus Supérieur. Cette carte contient 1000 neurones. Elle sera par la suite stimulée par une population de « cellules perceptives ».

La carte colliculaire projette des connexions vers une population de neurones de type EBN. Ces neurones représenteront la sortie motrice lue. Une intégration des PA émis par les EBN permet de traduire la position de l'œil. Les circuits neuronaux sous-tendant la transmission de ces informations aux muscles ne sont pas considérés dans ce modèle. Tous les neurones de la carte sont connectés avec les EBN selon une probabilité fixée à $prob = 0.3$ avec un poids de base des connexions synaptiques $poids = 9.5$. Ces poids diminuent exponentiellement avec l'augmentation de l'indice des neurones colliculaires, ce qui produit la différence de réponse Rostral/Caudal lors des stimulations (voir figure 6.8).

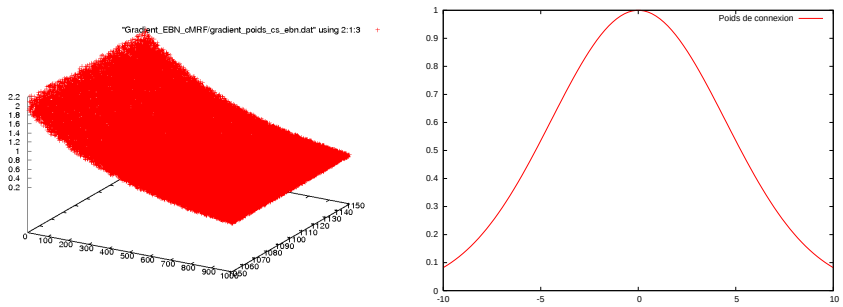

FIG. 6.8 – *A gauche : Valeurs du poids de la connexion (cote) en fontion du numéro d'un neurone pré-synaptique du CS (abscisse) et d'un neurone post-synaptique des EBN (ou de la cMRF) (ordonnée). A droite : Valeurs des poids excitateurs autour d'un neurone du CS placé en $x = 0$. L'écart type est de $\sigma = 0.05$ et la moyenne de $\mu = 0$*

La carte contient des connexions excitatrices internes. La position des neurones sur la carte est donnée par leur indice de 0 à $N-1$ dans l'ordre de leur création. Chaque cellule émet vers ses voisines avec un poids diminuant selon une gaussienne centrée sur la cellule (voir figure 6.8 et annexes C). Le poids de base affecté à ces connexions vaut $poids = 1.5$. Des connexions inhibitrices, par l'intermédiaire de 50 neurones inhibiteurs, permettent de maintenir une proportion d'environ 20% de neurones actifs en conditions normales. Les poids des connexions de CS sur les neurones inhibiteurs du CS vaut $poids = 1$ et la probabilité de connexion est de $prob = 0.05$. Les inhibiteurs projettent quant à eux sur le CS avec un poids synaptique : $poids = 0.3$ et une probabilité de connexion $prob = 0.5$.

Enfin le Colliculus Supérieur projette vers des neurones de la cMRF avec un poids de base $poids = 7$ et une probabilité de connexion $prob = 0.17$. Le même gradient de poids que celui des EBN est utilisé (figure 6.8).

6.3.2 Les neurones EBN

Dans le modèle proposé, les mouvements possibles de l'œil sont restreints à des mouvements horizontaux, afin de s'affranchir de la complexité des interactions entre les six paires de muscles. On ne conserve, dans notre modèle, que les muscles dirigeant les yeux de gauche à droite. De plus, on ne s'intéresse pour l'instant qu'aux saccades allant dans une seule direction, afin de limiter la complexité du réseau. Le fonctionnement est similaire pour le côté contra-latéral, et les EBN ipsi-latéraux inhibent les EBN contra-latéraux lors d'une saccade. La population d'EBN contient 100 neurones.

Ces neurones reçoivent des stimulations du CS, et sont inhibés par les neurones omnipause (OPN). Les neurones de la population EBN projettent vers la sortie motrice. Ils émettent également vers des neurones inhibiteurs projetant à leur tour sur les neurones de la cMRF. Les EBN ont une probabilité de connexion $prob = 0.4$ et un poids de base $poids = 1$ pour les connexions avec les inhibiteurs. Les inhibiteurs ont une probabilité de connexion avec les neurones de la cMRF $prob = 0.6$ et un poids de base $poids = 1$.

6.3.3 Les neurones OPN

Dans notre modèle, les neurones omnipause permettent, dans le cadre d'une stimulation électrique du Colliculus Supérieur, de stopper la saccade lors de leur remise en route. Dans notre modèle, les OPN sont stimulés, pendant toute la durée de l'expérience, par une unique cellule qui possède une connexion avec chacun des neurones OPN.

Les neurones OPN sont au nombre de 100. Ils projettent sur les neurones EBN, de manière inhibitrice. Leur probabilité de connexion aux EBN est de $prob = 1$ et le poids de base des connexions est de $poids = 10$.

6.3.4 Les neurones de la cMRF

La population de neurones de la cMRF représentée dans notre modèle, reçoit des projections excitatrices du Colliculus Supérieur et des projections inhibitrices des EBN (par l'intermédiaire d'une population d'interneurones).

Par ailleurs, la cMRF est composée de 100 neurones et projette vers les OPN. L'impact des neurones de la cMRF est inhibiteur. Ils permettent d'ouvrir la porte aux EBN pour émettre. En fonction de la saccade à effectuer (et donc de la position de stimulation sur le CS), la fréquence et la durée de l'activité de la cMRF varient. Lorsque cette activité a permis aux EBN d'émettre suffisamment de PA pour inhiber la cMRF en retour, on observe alors une diminution puis extinction de l'activité de la cMRF qui permet aux OPN de reprendre leur activité et d'inhiber suffisamment les EBN pour stopper leur activité (voir schéma sur la figure 6.7).

6.4 Principes de fonctionnement du modèle

Le modèle proposé ici définit des interactions entre les populations de neurones. Lors des simulations, on enregistre les activités de chacune de ces populations.

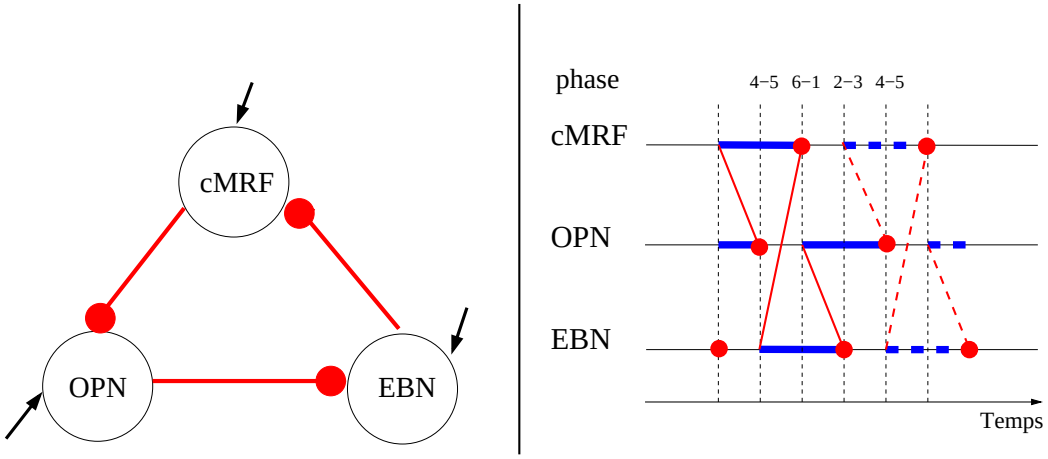

Fig. 6.9 – *Les populations cMRF, EBN et OPN interagissent cycliquement pour produire une saccade. À gauche, les trois populations sont stimulées (flèches noires). Les connexions en rouge (extrémité ronde) sont inhibitrices. À droite, décours temporel de l'activation du cycle. En bleu (trait épais), population en activité. Flèches rondes rouges (trait fin), impact inhibiteur de l'activité. En pointillé est indiqué le début du cycle suivant. Les six phases du cycle sont indiquées par les traits pointillés verticaux.*

On peut décrire le déroulement de la saccade illustré sur la figure 6.9 de la manière suivante. Au départ, seuls les OPN sont stimulés. Au bout de 5ms, la stimulation du CS débute. L'activité du colliculus est rapidement stabilisée (10ms après l'apparition d'une activité dans CS). Le CS excite à la fois les neurones EBN et cMRF. Les EBNs sont inhibés pas les OPN. La cMRF, stimulée par le CS, entre en activité (première ligne pointillée verticale sur la figure 6.9) et vient inhiber les OPN. Cette inhibition provoque l'arrêt des OPN relâche la contrainte sur les EBN qui entrent en activité (seconde ligne pointillée), démarrant la commande pour l'exécution de la saccade. Les EBN inhibent alors la cMRF dont l'inhibition sur les OPN diminue. Cette inhibition entraîne l'arrêt des neurones de la cMRF et permet la reprise de l'activité des OPN (troisième ligne pointillée) qui stoppent ensuite les neurones EBN (quatrième ligne pointillée).

Le maintien de la stimulation du CS permet à ce cycle d'inhibitions mutuelles de

se reproduire pour générer une nouvelle saccade. Le cycle se décompose donc en 6 phases :

1. Activation des OPN
2. Désactivation des EBN
3. Activation de la cMRF
4. Désactivation des OPN
5. Activation des EBN
6. Désactivation de la cMRF

La figure 6.9 illustre le décours temporel de ce cycle.

6.4.1 Exemple d'exécution

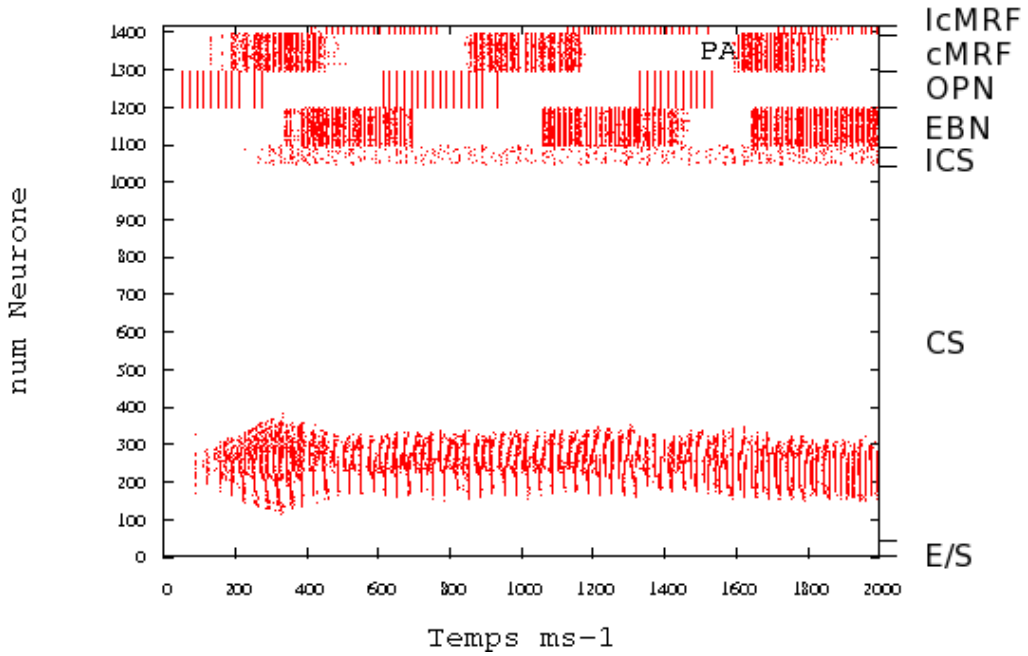

FIG. 6.10 – *Raster des activités enregistrées dans le modèle de contrôle des saccades oculaires. Le temps est en abcisse et chaque PA émis par un neurone n_i est indiqué par un point d'ordonnée i. Les lignes numéro 48 à 1047 sont les neurones de la carte colliculaire. Puis de 1048 à 1097 viennent les inhibiteurs du CS, de 1098 à 1197 les neurones EBN, de 1198 à 1297 les neurones OPN, de 1298 à 1397 les neurones de la cMRF et enfin de 1398 à 1417 on trouve les neurones inhibiteurs de la cMRF. Dans cet exemple, le CS est stimulé en position caudale (position 10).*

Pour chacune des simulations effectuées par la suite, la carte colliculaire est stimulée à l'aide de 45 cellules de stimulation ciblant chacune un champ récepteur d'un rayon de 5 % de la taille de la carte colliculaire. Ces cellules sont numérotées de 1 à 45. Les plus petits indices sont les plus caudaux. Une stimulation d'un rayon de 5 % entraîne, par excitation latérale, le recrutement d'environ 20 % de la population colliculaire avant de générer la saccade. La fréquence de stimulation représente le nombre d'événements PA émis par les cellules perceptives. Elle est de 300 Hz avec un poids

de 3,7 et elle entraîne une fréquence moyenne de décharge d'environ 500 - 600 Hz dans le Colliculus Supérieur.

Par ailleurs, les neurones omnipause (OPN) reçoivent également une excitation extérieure. La source biologique de cette excitation est floue, mais les OPN soutiennent une activité intense (environ 400Hz) avant le début d'une saccade et à la fin d'une saccade.

Classiquement, une stimulation extérieure est appliquée aux OPN. Elle se fait, dans notre modèle, par l'intermédiaire d'une unique cellule de stimulation (numéro 46). Elle possède une connexion avec chaque neurone OPN. Elle émet des PA à une fréquence de 500Hz avec un poids de 30 (pour assurer une émission à chaque stimulation (voir annexes C)).

Sur la figure 6.10, on peut observer le déroulement d'une exécution typique. L'enregistrement est effectué durant 200ms. L'échelle temporelle de la figure (abscisse) est le dixième de milliseconde (ms^{-1}). La stimulation du CS et des OPN est appliquée pendant 150ms. L'enregistrement des PA émis se poursuit jusqu'à 200ms. L'axe des ordonnées de gauche indique le numéro des neurones dans l'implémentation et l'axe de droite les populations correspondantes. On peut alors aisément observer l'enchaînement des six phases présentées précédemment (figure 6.9).

6.5 Protocole de stimulation

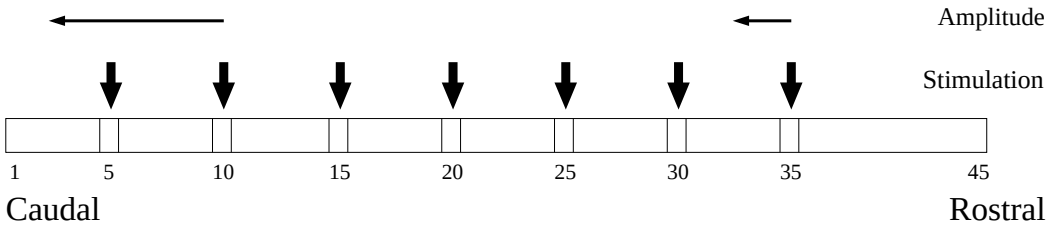

Fig. 6.11 – *Position des stimulations sur la carte colliculaire. Les sept stimulations sont appliquées aux positions 5, 10, 15, 20, 25, 30 et 35 (Flèches épaisses). Les stimulations rostrales (resp. caudales), à droite (resp. gauche) provoquent une saccade de faible (resp. grande) amplitude (flèches fines).*

Dans un premier temps, pour valider le modèle de carte colliculaire, on choisit sept positions sur la carte. Les positions de stimulation choisies sont indiquées sur la figure 6.11. Elles seront identifiées par le numéro de la cellule perceptive utilisée pour la stimulation : 5, 10, 15, 20, 30 et 35. La figure 6.11 représente la carte colliculaire sur une seule dimension. Elle correspond à une portion du colliculus droit, du pôle rostral (à droite sur la figure) au pôle caudal (à gauche sur la figure). Les amplitudes de saccade observées en biologie sont faibles pour les stimulations proches du pôle rostral et augmentent en s'éloignant vers le pôle caudal.

À partir du raster, la fréquence instantanée moyenne des neurones de chaque population est calculée, avec une fenêtre glissante d'une largeur de 10ms. La figure 6.12 présente un exemple de profil de fréquence de décharge des neurones des populations EBN, OPN et cMRF lors de l'exécution d'une saccade dans le modèle. L'étroite relation entre l'activité de burst des neurones EBN est le déplacement final de l'oeil, avérée en neurophysiologie, nous permet d'exploiter l'intégrale des émissions des EBN pour représenter une approximation du déplacement de l'oeil. Cette somme est

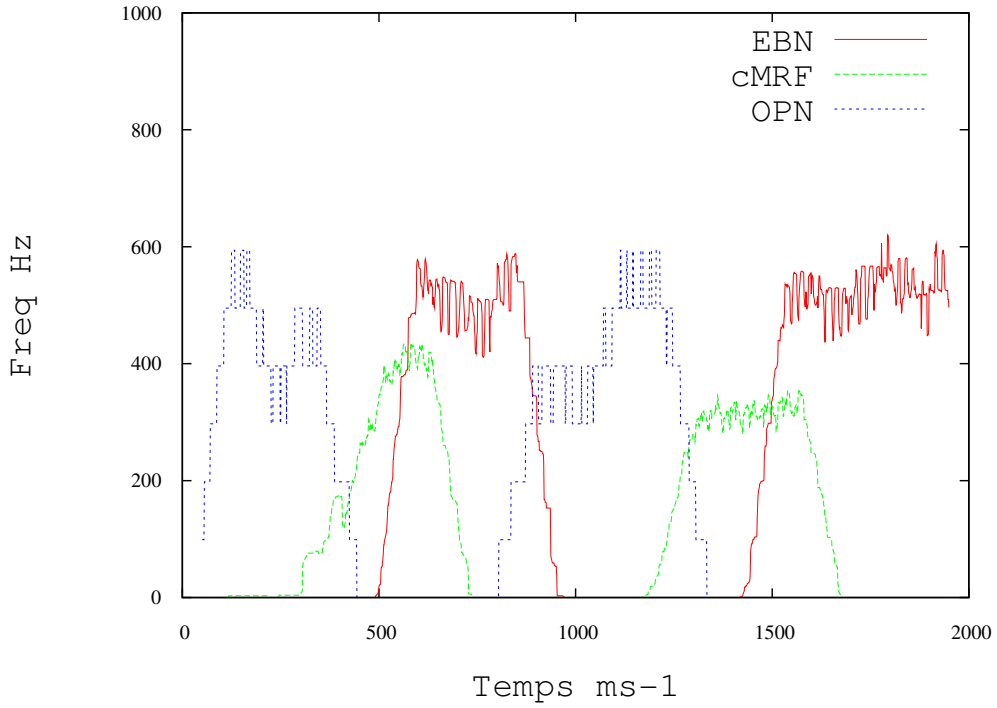

FIG. 6.12 – *Courbes de fréquences de décharge instantanée des neurones EBN (rouge), cMRF (vert) et OPN (bleu). Pour cet exemple, la stimulation était appliquée en position 35 sur la carte colliculaire.*

calculée à partir des instants d'émission des EBN. Le profil de décharge des EBN peut également être utilisé comme approximation de la vitesse de déplacement de l'oeil.

6.6 RÉSULTATS

6.6.1 Amplitude des saccades

Position de la stimulation

Les figures 24 à 30 fournies en annexe C, présentent les rasters, les courbes de fréquences des neurones EBN OPN et cMRF, ainsi que le déplacement de l'oeil pour chacune des positions. La figure 6.13 (A) représente la variation de l'amplitude des saccades effectuées en fonction de la position de la stimulation. Plus la stimulation est rostrale (vers la droite sur la figure), moins l'amplitude de la saccade est élevée. La courbe obtenue est comparable à celle obtenue lors de micro-stimulations électriques réelles chez l'animal (B). On retrouve bien la non-linéarité de la relation entre la position sur la carte colliculaire et l'amplitude de la saccade.

Sur la figure 6.14 A, trois des sept positions de stimulation sont représentées et comparées avec les observations lors d'enregistrements neurophysiologiques (figure 6.14 B) : la position la plus rostrale (position 35) ayant entraîné la saccade de plus faible amplitude, une position médiane (position 20) et la position la plus caudale (position 5) ayant entraîné la saccade de plus forte amplitude. Pour chacune de ces stimulations, les courbes de position de l'oeil et celle de fréquence de décharge

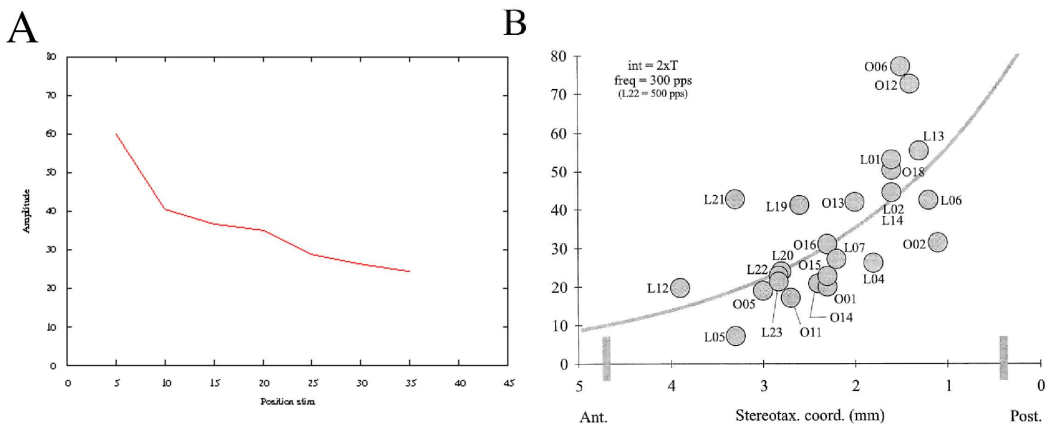

FIG. 6.13 – *Amplitude maximum de la première saccade générée (ordonnée) fonction de la position (abscisse) de la stimulation. Les données obtenues en neurophysiologie sur le chat sont présentées à droite (Guillaume & Pelisson 2001). À gauche les amplitudes données par le modèle. Les amplitudes du modèles sont normalisées par rapport à l'amplitude maximum enregistrée (position 5). À 1mm du pôle caudal, on observe des amplitudes d'environ 60 degrés (site L13 le plus proche de la courbe ajustée :* $f(x) = 89.5 \times e^{-0.47x}$*).*

des neurones EBN sont représentées. L'échelle temporelle est en dixième de milliseconde et la simulation dure 200ms. Sur cette figure, les amplitudes des saccades générées par le modèle ne sont pas normalisées. Ainsi les valeurs indiquées pour les amplitudes sont abstraites. La saccade la plus grande générée par le modèle (position 5) est l'amplitude de référence utilisée pour la normalisation de la figure 6.13.

Quelque soit le site de stimulation, les saccades se terminent malgré la stimulation continue, puis l'oeil reprend son déplacement pour effectuer une nouvelle saccade. C'est le phénomène de *staircase saccade* observé en neurophysiologie en conséquence d'une stimulation continue et prolongée du colliculus supérieur. La forme des *staircases* saccades obtenues avec le modèle n'est pas tout à fait identique, notamment au niveau de l'amplitude de la seconde saccade et des suivantes qui, biologiquement, est de moindre amplitude que la première saccade (voir section 6.2.3). Dans le modèle, les saccades suivant la première sont parfois plus grandes, parfois plus petites.

Notre modèle ne prend pas en compte un facteur important qui pourrait expliquer cette différence entre la biologie et la modélisation. L'amplitude des saccades évoquées dépend de la position de l'œil au moment de leur initiation. Ainsi, les secondes saccades évoquées sont d'amplitude plus faible parce que l'œil est déjà dévié dans l'orbite au moment de leur initiation. Cette influence de la position initiale de l'œil n'est pas intégrée dans notre modèle.

Par ailleurs, comme en neurophysiologie, la variabilité des réponses est grande. Les durées des saccades générées par le modèle sont présentées sur la figure 6.15 et montrent que plus les saccades sont caudales plus elles ont une durée importante.

D'un point de vue computationnel, la variation de fréquence de décharge des EBN s'explique par la forme de la distribution des poids des connexions entre les neurones de la carte colliculaire et les neurones EBN (voir figure 6.8, à gauche). On constate que la boucle de rétroaction mise en place (figure 6.9) est bien reproduite dans la dynamique des neurones. On constate également que l'équilibre de cette boucle est précaire, la forme des réponses obtenues et leur variabilité (introduite dans

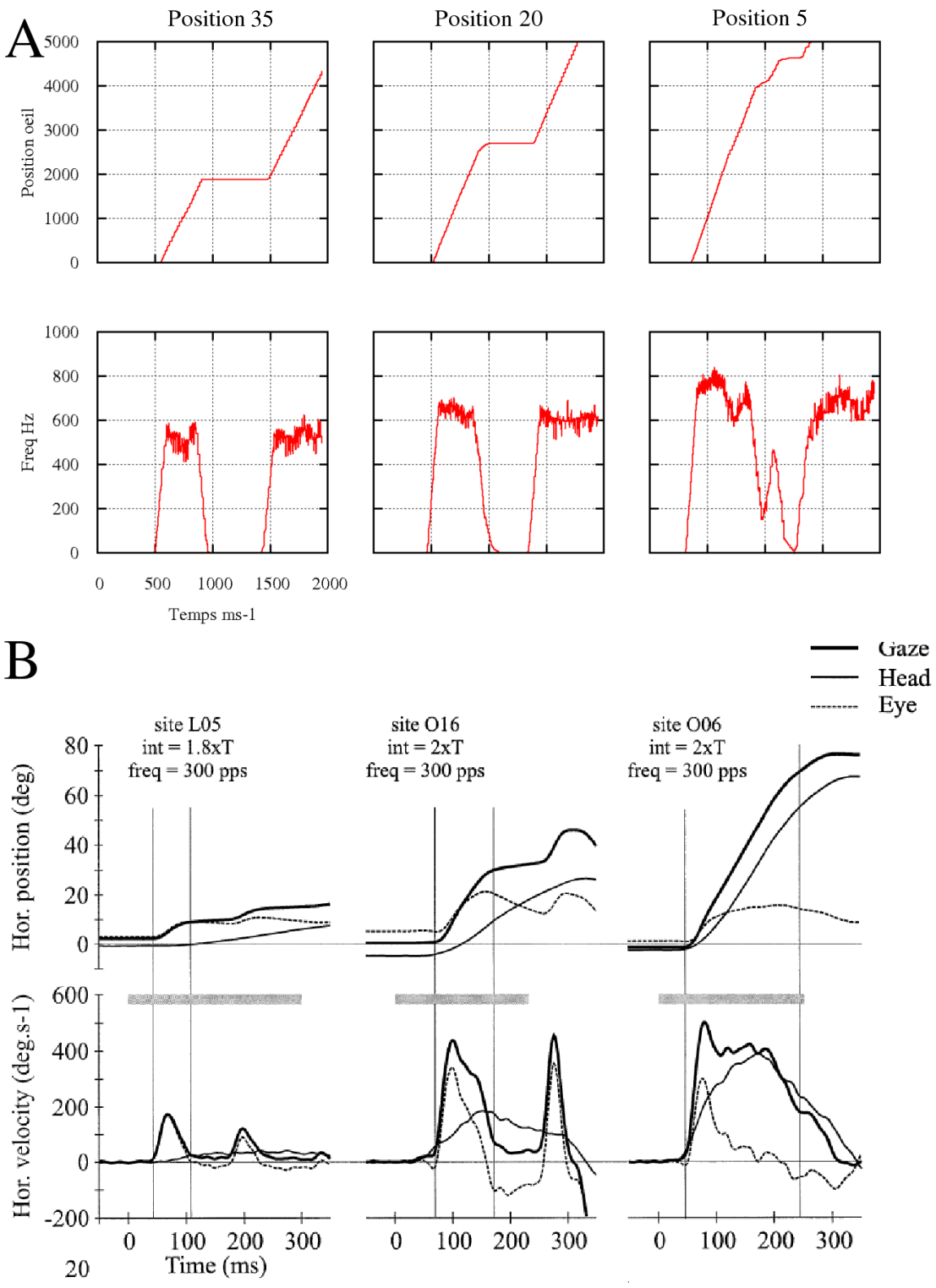

FIG. 6.14 – *Comparaison des données obtenues par simulation du modèle (A) et par enregistrement électrophysiologique (en condition tête libre) (B, d'après (Guillaume & Pelisson 2001)). Les tracés sont indiqués pour une position rostrale, une position médiane et une position caudale pour le modèle et pour les enregistrements.*

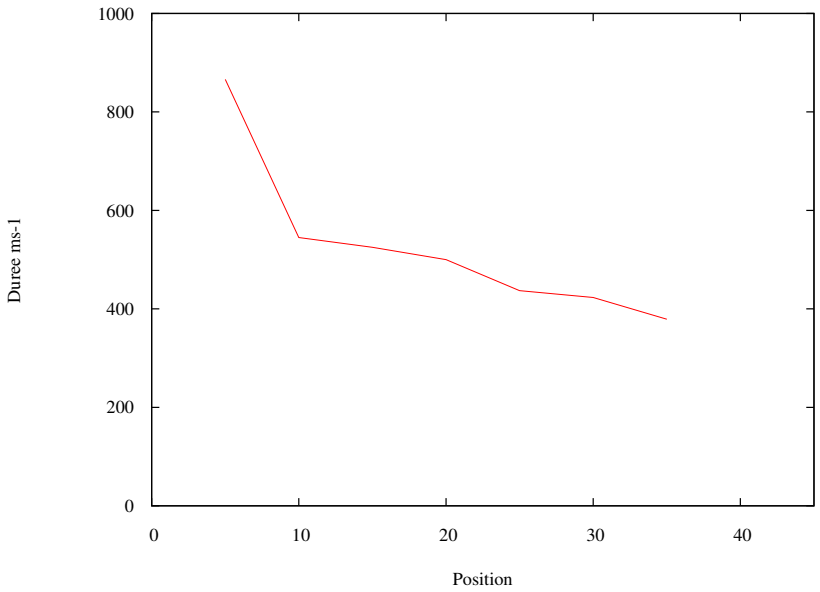

FIG. 6.15 – *Pour chaque position de stimulation (abscisse), la durée de la saccade observée est indiquée en ordonnée.*

les dispersions aléatoires des poids et délais autour des valeurs moyennes) indique clairement qu'il manque certaines influences dans le modèle tel qu'il est implémenté.

Quoi qu'il en soit, le modèle de carte colliculaire ainsi validé nous permet de poursuivre notre étude en testant l'influence de l'intensité de la stimulation sur les saccades générées.

Intensité de la stimulation

La figure 6.16 présente, comme précédemment, les amplitudes des saccades et les fréquences de réponse du modèle lorsque l'on fait varier l'intensité de la stimulation. Dans notre modèle, cette intensité est déterminée par le poids de la stimulation dans les connexions avec la carte colliculaire et par la population colliculaire impactée par la stimulation. En conditions normales, le poids de la stimulation est de $3,7$ et la population stimulée sur la carte colliculaire représente environ 10% de la carte (voir figure 6.7), puis par excitations latérales, environ 20% des neurones de la carte colliculaire sont activés. Par analogie avec les expériences en neurophysiologie, cette intensité constituera notre intensité $2 \times T$, où T représente l'intensité minimum nécessaire pour obtenir une saccade. Pour étudier l'influence de l'intensité, on applique alors une variation de l'intensité de la stimulation pour un site caudal (position 10) et un site rostral (position 30). Les intensités $1,3 \times T$ (poids : $2,8$, rayon 2.5%), $2 \times T$ (poids : $3,7$, rayon : 5%) et $6 \times T$ (poids : 11, rayon : 15%) sont appliquées.

Les variations des amplitudes des saccades observées en réponse à la variation de l'intensité sont limitées. Une explication vient du mode d'inhibition mis en place dans la carte colliculaire. En effet, l'inhibition totale (voir section 6.3.1) de la carte empêche, dans une certaine mesure, que l'augmentation de la stimulation engendre l'activation d'une plus grande population colliculaire. Sur les rasters 31 à 34 en annexe

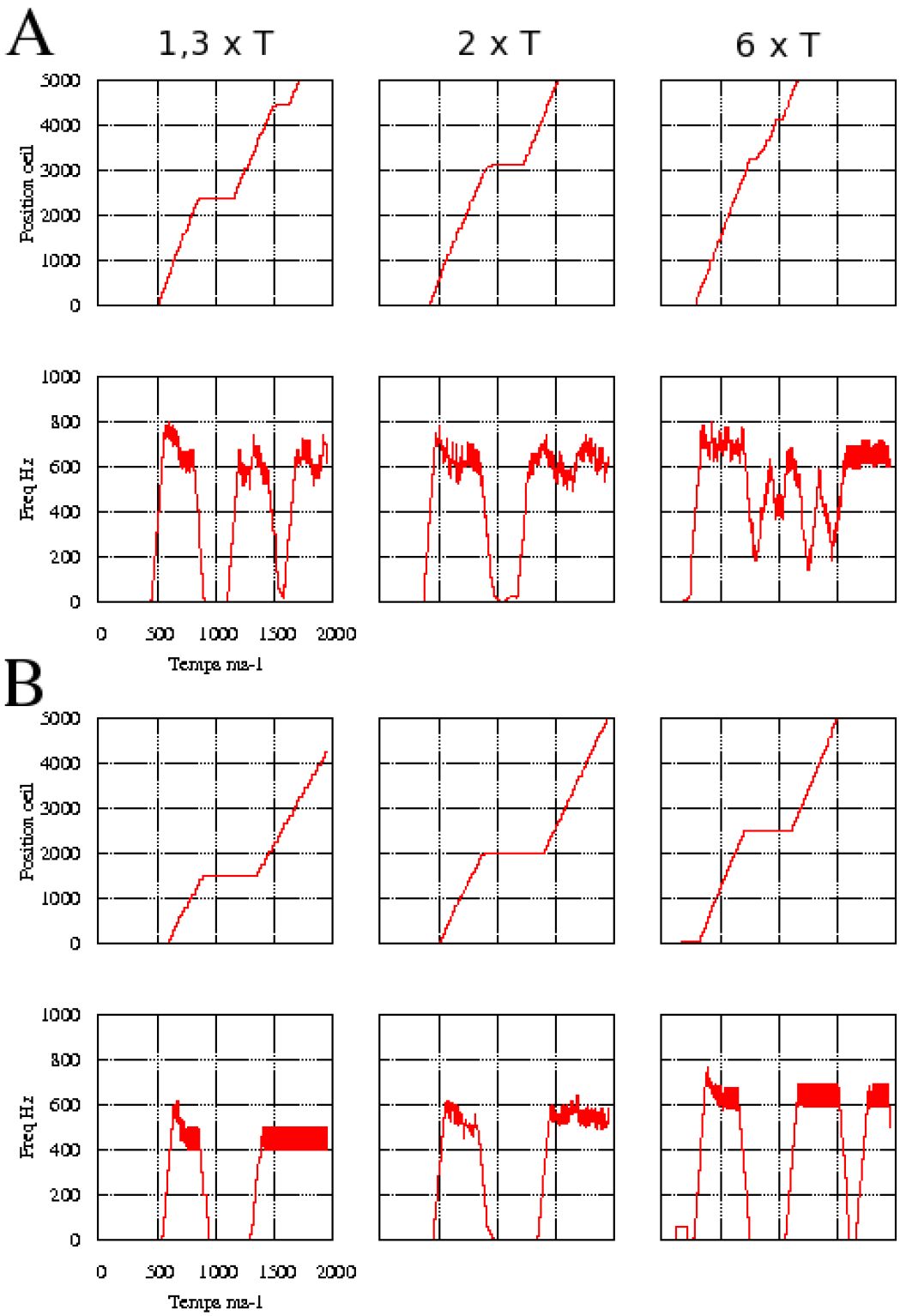

FIG. 6.16 – *Amplitude des saccades (position oeil) et vitesse de déplacement (freq. EBN) pour trois intensités de stimulation* $1,3 \times T$ *(à gauche),* $2 \times T$ *(au centre) et* $6 \times T$ *(à droite). A) Position 10, caudale. B) Position 30, rostrale.*

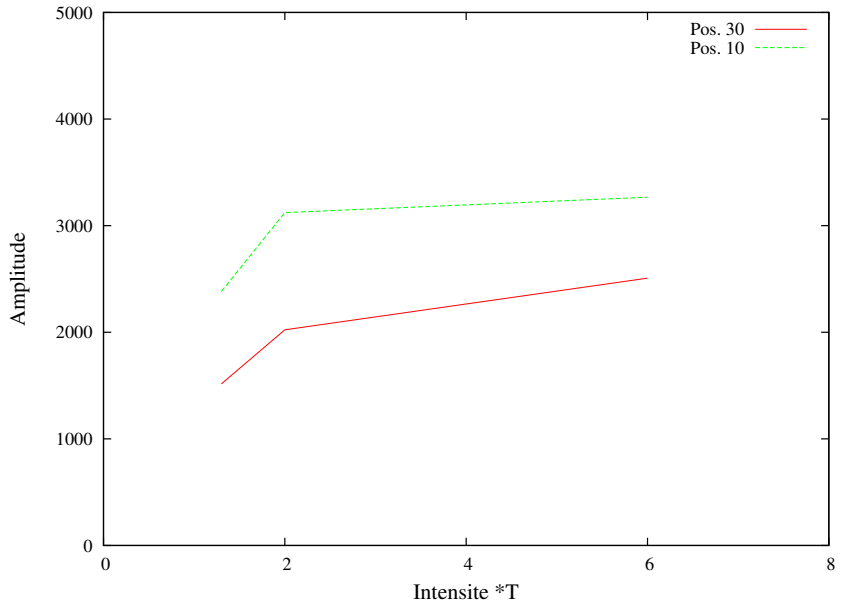

FIG. 6.17 – *Amplitude des saccades en fonction de l'intensité de la stimulation. Les amplitudes pour les deux positions de stimulation sont indiquées sur le même graphique.*

C, on observe bien le recrutement d'une plus grande population dans les premières millisecondes d'activité, mais l'inhibition réduit rapidement celle-ci.

Fréquence de la stimulation

Pour étudier l'influence de la fréquence de stimulation sur les amplitudes des saccades, nous conservons les deux positions de stimulation 10 et 30. En conditions normales, la fréquence de stimulation est de 300 Hz (voir section 6.4.1) à une intensité $2 \times T$. La figure 6.18 illustre la réponse du modèle à une stimulation aux fréquences de 150Hz, 300Hz, 500Hz et 750Hz avec une intensité $2 \times T$.

La figure 6.19 A montre l'évolution de l'amplitude des saccades générées par le modèle lorsque l'on fait varier la fréquence de stimulation. On constate que l'amplitude des saccades est plutôt stable entre 300Hz et 500Hz et cette amplitude diminue pour des stimulations à 150Hz et 750Hz. Ce résultat est très proche des données obtenues en électrophysiologie dans l'étude de (Guillaume & Pelisson 2001) (voir figure 6.19, B).

6.6.2 Latence des saccades

La figure 6.20 illustre les variations des latences des saccades selon différentes conditions. La latence correspond à la durée de stimulation nécessaire avant l'initiation de la saccade. La latence des saccades diminue à mesure que la position de stimulation est plus caudale. Les latences enregistrées sont d'environ 35 à 50ms (figure de gauche).

La latences observées lors de la variation de l'intensité et de la fréquence sont

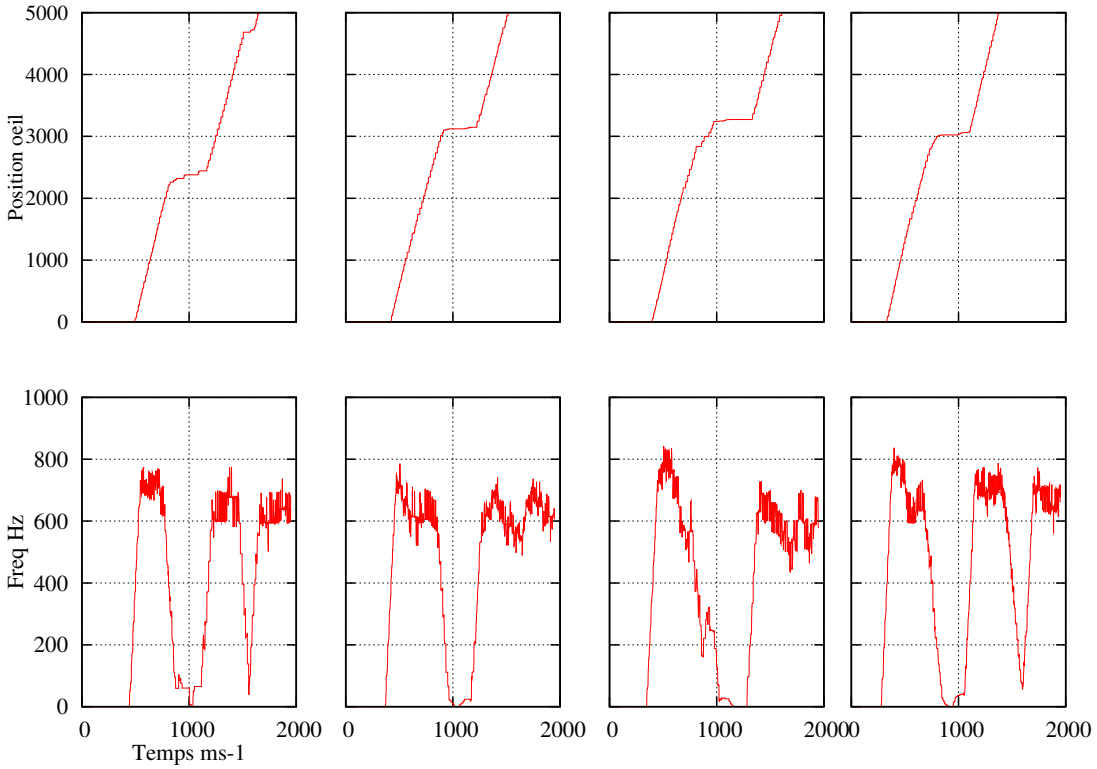

FIG. 6.18 – *Position de l'oeil et vitesse de déplacement pour quatre fréquences de stimulation* 150*Hz*, 300*Hz*, 500*Hz et* 750*Hz. La stimulation est appliquée en position 10.*

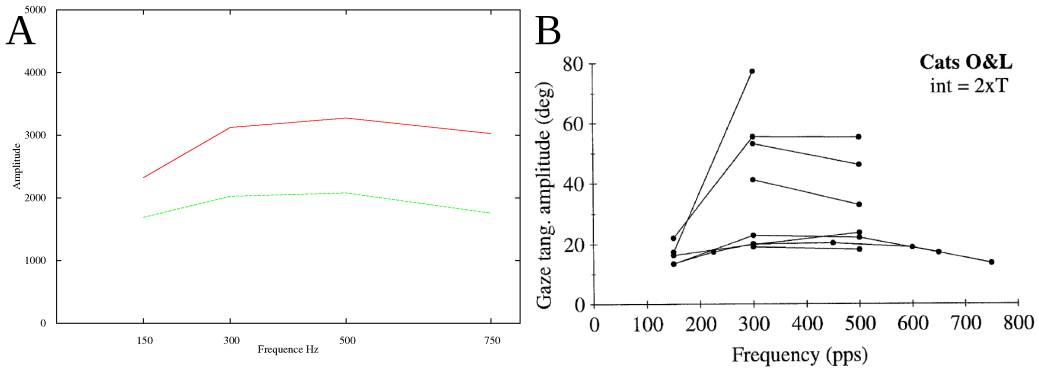

FIG. 6.19 – *A), Amplitude des saccades obtenues pour une stimulation en position 10. Les fréquences de 150, 300, 500 et 750 Hz ont été testées. B) Amplitude des saccades observées chez le chat dans (Guillaume & Pelisson 2001) en fonction de la fréquence de stimulation.*

illustrées sur les figure 6.20 au centre et à droite. Les positions de stimulation 10 et 30 sont illustrées.

6.7 Discussion

Le modèle de contrôle oculomoteur par le tronc cérébral proposé dans ce travail est basé sur les données anatomiques et physiologiques actuelles. De plus, notre modèle

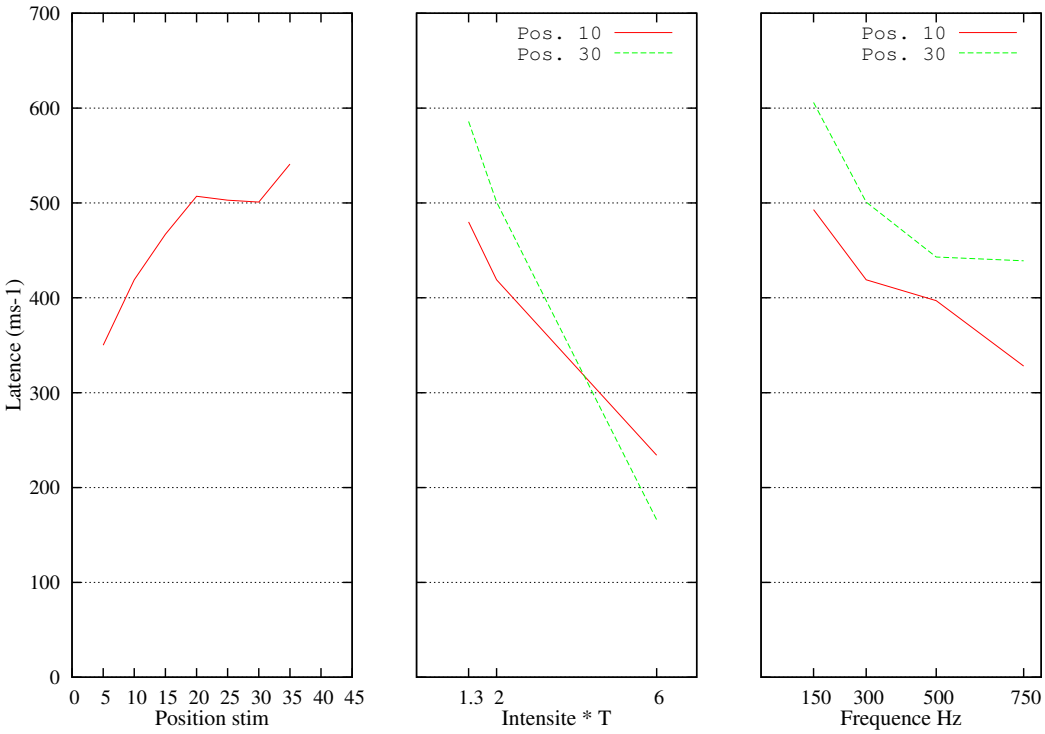

FIG. 6.20 – *Latence des saccades dans le modèle. À gauche, latences en fonction de la position de la stimulation sur le CS. Au centre, latence des saccades en fonction de l'intensité de la stimulation. Les positions de stimulation 10 et 30 sont illustrées. À droite, latence des saccades en fonction de la fréquence de stimulation. Les positions de stimulation 10 et 30 sont illustrées.*

propose que le colliculus ne fasse pas partie de la boucle de rétroaction permettant l'arrêt des saccades (voir section 6.1.2).

6.7.1 Comportement du modèle

Le comportement du modèle, en conditions normales de stimulation (voir section 6.6.1) est très proche des observations biologiques (pour des microstimulations électriques). Les plages de fréquences d'activités des neurones, les latences des saccades ainsi que les dynamiques des populations neuronales modélisées sont également comparables aux observations biologiques.

Les caractéristiques des mouvements, extrapolés à partir de l'activité des neurones EBN, générés par le modèle, sont proches des saccades obtenues par microstimulations électriques comme l'illustre la figure 6.14. Les plages de valeurs des amplitudes obtenues avec le modèle ne sont que théoriques et sont normalisées pour faciliter la comparaison (voir figure 6.13), mais ces plages de valeurs sont du même ordre que les variations enregistrées en électrophysiologie.

Le modèle de carte motrice colliculaire proposé est donc efficace pour traduire la variation de la position de la stimulation sur la carte en modulation de la fréquence de décharge des neurones EBN.

Par ailleurs, le modèle de boucle de rétroaction implémenté, excluant le colliculus, permet une terminaison efficace des saccades. Les durées des saccades qui dépendent

étroitement de cette boucle de rétroaction, sont comparables aux durées biologiques (15100ms,Büttner & Büttner-Ennever (2006)) puisqu'elles sont d'environ 3585ms dans le modèle (voir figure 6.15). Les neurones des populations EBN, cMRF et OPN obtenues avec le modèle (voir figure 6.10 et les figures 24 à 40 fournies en annexe), montrent des activités très proches de celles enregistrées lors de stimulations électriques (voir figures 6.4 et 6.5).

6.7.2 Influence de la taille de la population sur l'amplitude

Variation de l'intensité

Lorsque l'on fait varier la taille de la population du CS activée en faisant varier l'intensité de la stimulation, on observe bien une variation de l'amplitude des saccades (voir figure 6.16) mais cette variation est moins importante que dans les enregistrements de Guillaume & Pelisson (2001).

Dans notre normalisation des amplitudes des saccades du modèle, l'amplitude maximum obtenue dans les enregistrements est utilisée comme base pour la normalisation. Cependant, la borne minimum (stimulation en position 35 dans le modèle) n'a pas été calculée à partir de l'amplitude minimum enregistrée (voir figure 6.13) mais en fonction de sa « distance » à l'amplitude maximum. En envisageant une renormalisation des amplitudes en fonction de l'intervalle [min : max] biologique, les variations d'amplitudes obtenues par la modélisation seraient nécessairement plus importantes.

Par ailleurs, sur les figures 31 à 34 fournies en annexes, on constate que l'augmentation de la population colliculaire activée induite pas la variation de l'intensité de la stimulation n'est pas très importante. Compte tenu du mode d'inhibition implémenté dans le modèle, l'augmentation de la taille de la population activée implique une augmentation forte de l'inhibition sur toute la carte, qui limite très rapidement la taille de la population active dans le CS lors de l'augmentation de l'intensité de la stimulation.

Des tests, non présentés ici, ont montré qu'une variation du poids de l'inhibition sur la carte colliculaire permet une plus grande variation de la taille de la population. D'un point de vue computationnel, il est évident qu'en relâchant l'inhibition sur le réseau colliculaire cela entraîne immédiatement l'activation d'une plus grande population pour une même intensité de stimulation.

L'implémentation d'une inhibition distante progressive dans la carte colliculaire pourrait offrir un comportement plus proche du comportement biologique. Les informations neuroanatomiques et neurophysiologiques concernant le mode d'inhibition des couches profondes du Colliculus Supérieur manquent encore. Les enregistrements effectués lors des microstimulations électriques ne permettent pas d'enregistrer l'activité colliculaire en même temps, ce qui explique en partie le manque de données. On ne peut donc pas affirmer que la taille de la population activée dans le CS est constante pendant toute la durée des microstimulations.

Dans les travaux de Guillaume & Pelisson (2001), l'intensité utilisée lors de l'augmentation de l'intensité de stimulation est de $3 \times T$. Dans le modèle, en conservant les paramètres de simulation identiques (modèle de neurone, populations, connexions, poids, délais, stimulations), l'utilisation d'une intensité à $3 \times T$ n'engendre pas, ou très peu, de variations de la taille de la population activée. C'est pourquoi, dans le

modèle, l'intensité utilisée pour l'augmentation de l'intensité de stimulation est de $6 \times T$. À cette intensité, la taille de la population activée est bien d'environ 30% de la carte colliculaire. Les variations d'amplitude des saccades observées dans le modèle, à cette intensité, restent de moindre ampleur que dans les expériences de microstimulation, mais montrent néanmoins clairement l'impact de la variation de la taille de la population active dans le CS sur l'amplitude des saccades générées.

De plus, l'influence de la variation est, dans le modèle, plus importante pour les positions rostrales que pour les positions caudales (voir figure 6.17). Ceci est en accord avec les résultats des expériences en microstimulation où l'amplitude des saccades pour les sites caudaux est déjà à sa valeur maximale à une intensité de $2 \times T$ dans Guillaume & Pelisson (2001).

Ainsi, l'hypothèse de sommation vectorielle (voir section 6.2.2), dont les conséquences sur la variation de l'amplitude des saccades, lors de la variation de l'intensité de stimulation, sont illustrées sur la figure 6.6 B), est confirmée par les résultats du modèle (figure 6.16). En effet, la variation de l'intensité a un impact clair sur l'amplitude des saccades générées, contrairement aux prédictions de l'hypothèse de moyennage vectoriel.

Variation de la fréquence

Les variations appliquées sur la fréquence de stimulation du modèle engendrent, sur les amplitudes de saccades, des variations proches de celles enregistrées chez l'animal. Pour des fréquences de stimulation entre 300 et 500Hz les saccades sont d'amplitude équivalente. Mais si la fréquence de stimulation est plus faible (150Hz) ou au contraire, plus forte (750Hz), l'amplitude des saccades diminue.

Selon (Guillaume & Pelisson 2001), la variation de la fréquence de stimulation du colliculus pourrait, comme dans le cas de variation de l'intensité, entraîner une variation de la taille de la population activée dans le colliculus, à l'origine des variations d'amplitude des saccades. Plusieurs études ont déjà mis à jour l'existence de ce type d'influence de la fréquence de stimulation sur l'amplitude des saccades, mais la nature exacte de la relation entre la fréquence de stimulation et la taille de la population activée dans le colliculus reste encore à déterminer.

Cependant, dans le modèle, les variations de fréquence de stimulation n'ont que très peu d'impact sur la taille de la population activée (voir figures 35 à 40 en annexe). Plus que la taille de la population activée, ce sont les fréquences de décharge des neurones qui sont modifiées. Dans la boucle de rétroaction proposée dans ce modèle, seuls les neurones OPN ne reçoivent pas de stimulation du colliculus. La stimulation du colliculus atteint les neurones EBN et la cMRF. Or l'activité de la cMRF détermine l'arrêt et la reprise des OPN, et l'activité des EBN détermine l'arrêt et la reprise des neurones de la cMRF. Ainsi, en augmentant la fréquence de stimulation, on augmente à la fois la fréquence des EBN et de la cMRF ce qui permet de conserver un équilibre entre les deux populations et peut expliquer le maintient de l'amplitude des saccades générées par le modèle. La diminution des amplitudes observée pour des stimulations à 150Hz s'explique par le fait qu'en dessous de 300Hz la taille de la population activée dans le colliculus diminue. On observe également une diminution de l'amplitude des saccades dans les cas de stimulation à 750Hz. Dans ce cas, le fait que l'amplitude de la saccade diminue peut-être du au fait qu'à 750Hz, compte tenu de

la fréquence de décharge dans le colliculus et de la période réfractaire des neurones, une partie des impulsions de stimulation sont « ignorées ». Dans ce cas, les stimulations prises en compte par les neurones du colliculus pourraient être équivalentes à une stimulation à 500Hz ou moins.

6.7.3 La boucle de rétroaction

L'équilibre de la boucle de rétroaction implémentée dans le modèle repose sur les délais et les rapports entre les influences excitatrices et inhibitrices entre les populations de neurones OPN, EBN et cMRF.

Bien que la diminution de l'intensité de la stimulation entraîne effectivement une diminution de la taille de la population activée et donc de l'amplitude, l'augmentation de l'intensité a un impact limité sur l'amplitude des saccades (voir figure 6.17).

D'un point de vue biologique, on peut associer cette limitation à une saturation des muscles de l'oeil menant à une saturation de l'amplitude des saccades. Il apparaît cependant que le type d'inhibition existant dans le CS et permettant le maintient d'une bulle d'activité pour la génération d'une saccade, pourrait être déterminant dans les variations de l'amplitude des saccades lors de la variation de l'intensité de la stimulation.

D'un autre point de vue, l'activité des neurones de la cMRF est responsable, dans le modèle, de l'arrêt des OPN. Dès l'arrêt des cMRF, les OPN reprennent et inhibent fortement les EBN. La durée de la saccade dépend donc d'une étroite relation entre l'activité des activités des OPN, EBN et cMRF. L'augmentation de l'intensité de la stimulation augmente l'activité des EBN, mais en contrepartie, il n'y a pas de variation de l'intensité de la stimulation des OPN ce qui implique que le « frein » appliqué aux EBN par les OPN est comparativement plus fort lorsque l'on diminue l'intensité, et plus faible lorsque l'intensité de stimulation augmente. dans les cas d'augmentation de la fréquence, l'inhibition des OPN n'est plus totalement efficace ce qui mène à une série de saccades « déformées » (figure 6.16) pour les intensités fortes (position 10 à $6 \times T$). Dans les enregistrements électrophysiologiques effectués par Guillaume & Pelisson (2001), l'augmentation de l'intensité de la stimulation entraîne une augmentation de l'amplitude d'autant plus forte que la position de la stimulation est caudale. Ainsi, les limitations observées dans le modèle montrent que le modèle proposé manque malgré tout d'un « contrôle » plus précis des saccades (par le biais d'un contrôle des dynamiques neuronales). La boucle de rétroaction implémentée ici semble donc être incomplète.

De nombreuses afférences et efférences au système oculomoteur du tronc cérébral ont été omises dans le modèle proposé dans ce travail. Ces omissions peuvent expliquer que la boucle de rétroaction modélisée soit incomplète. En particulier, l'arrêt de la saccade, dans notre modèle, est commandé par la reprise d'activité des neurones OPN. Pourtant, des études ont montré qu'une inactivation temporaire des neurones OPN avait pour conséquence une baisse de la vitesse de déplacement et une augmentation de la durée de la saccade (Soetedjo et al. 2002). Cependant, les saccades obtenues se terminent malgré tout correctement. Cela montre que l'action des OPN, bien qu'effectivement inhibitrice sur les EBN, n'est pas seule responsable de l'arrêt de la saccade. Dans notre modèle, une inactivation des OPN mènerait inévitablement à une impossibilité de stopper les saccades.

Par ailleurs, de nombreux éléments indiquent que la formation réticulée mésencéphalique centrale (cMRF) pourrait être la source d'un *feedback* vers le colliculus qui pourrait permettre un contrôle de la saccade. La cMRF reçoit des afférences du CS et projette également vers le CS.

6.7.4 Perspectives

À partir des résultats obtenus dans ce travail, il apparaît tout d'abord que le mode d'inhibition implémenté dans le modèle de carte motrice colliculaire ne permet que partiellement le maintient de la bulle d'activité. Une inhibition distante progressive sur la carte colliculaire pourrait permettre plus de variations de la taille de la population activée dans le CS lors des variations de l'intensité de stimulation (Lee & Hall 2006).

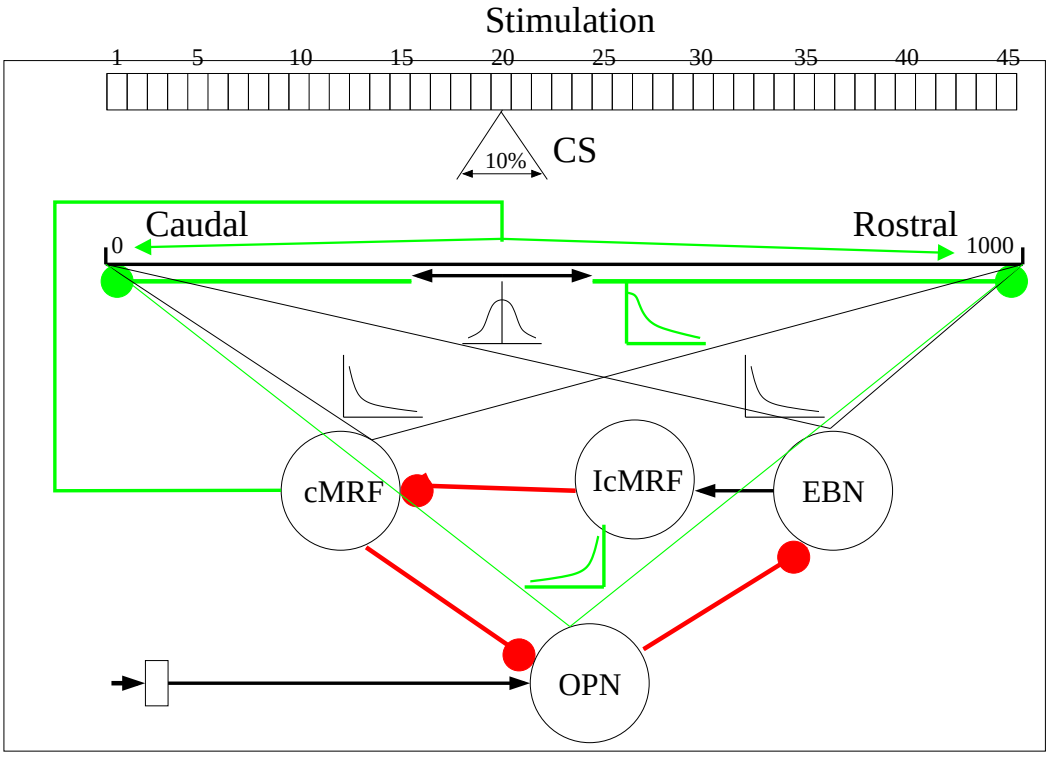

FIG. 6.21 – *La figure reprend le modèle proposé dans ce travail. Les projections supplémentaires proposées en section 6.8 sont ajoutées au modèle et sont présentées en vert.*

Une évolution du modèle pourrait prendre en compte ce type d'inhibitions dans le CS (voir figure 6.21).

Par ailleurs, une projection de la cMRF vers le CS permettrait un feedback qui pourrait avoir un rôle de régulateur de l'activité du CS en fonction de la progression de la saccade (voir figure 6.21). Cette projection pourrait être excitatrice. Celle-ci permettrait au CS de maintenir une activité pendant la durée de la saccade. À la fin de la saccade, lorsque les EBN ont suffisamment inhibé les neurones de la cMRF, l'activité de ces derniers décroît progressivement jusqu'à l'extinction (permettant la reprise de l'activité des neurones omnipause). Cette extinction serait alors synonyme d'un

baisse de la stimulation du CS qui ne serait plus en mesure de maintenir son activité. La saccade se terminerai alors dans une dynamique plus lente avec l'extinction du CS et donc des neurones EBN.

Ce type de feedback pourrait donc permettre un arrêt des saccades même en l'absence d'activité des neurones OPN (Soetedjo et al. 2002). La boucle de rétroaction proposée dans ce travail serait maintenue parallèlement. Dans un tel modèle, l'action des OPN sur les neurones EBN permettrait d'avoir un démarrage, une terminaison et une accélération plus francs dans l'exécution des saccades (Miura & Optican 2006).

Enfin, tester le modèle sur des saccades non plus induites par stimulation, mais par présentation d'une cible vers laquelle l'œil doit se déplacer. Dans ce cas, il pourrait être utile d'implémenter dans le modèle les projections existantes entre la carte motrice du colliculus supérieur et les neurones OPN. Cette projection est excitatrice pour la zone la plus rostrale du colliculus puis s'affaiblit en s'approchant du pôle caudal (Gandhi & Keller 1999).

6.8 Conclusion

Nous avons proposé un modèle simple du système saccadique, entièrement composé de neurones impulsionnels, dans lequel la boucle de rétroaction ne passe pas par le colliculus. La carte motrice du colliculus supérieur modélisée reproduit correctement l'influence de la position de la stimulation sur l'amplitude des saccades. L'étude de l'influence de l'intensité et de la fréquence de la stimulation a montré que les variations d'amplitude des saccades induites par les variations de la stimulation générées par le modèle sont comparables aux variations observées dans des études chez l'animal. Des durées et latences de saccades proches de celles observées en biologie sont générées par le modèle.

Ce travail montre que le modèle proposé est capable de générer des saccades précises malgré le maintient d'une activité continue dans le CS. La présence du CS dans la boucle de rétroaction donnant le signal de fin de la saccade n'est donc pas indispensable. Un modèle basant cette boucle sur les interactions entre des populations de neurones EBN, OPN et de neurones de la cMRF peut expliquer l'arrêt des saccades en conditions de microstimulation électrique.

Enfin, on a montré que le simulateur DAMNED est parfaitement adapté à la définition et à la simulation de modèles de réseaux de neurones précisément définis, dans le but d'étudier de réelles Hypothèses neuroscientifiques. En apportant, pour la première fois à l'aide d'un modèle entièrement composés de neurones échangeant des potentiels d'action, un éclairage sur le fonctionnement des saccades oculaires à un niveau sous-cortical, ce travail permet aux biologistes de préciser les modèles de contrôle des saccades oculaires envisageables et d'écarter certaines hypothèses inexploitables lorsqu'appliquées à des réseaux de neurones biologiques.

Conclusion

L'objectif principal de ce travail de thèse était d'apporter à la communauté scientifique des neurosciences computationnelles la possibilité de manipuler de grands réseaux de neurones impulsionnels, à l'aide d'une stratégie événementielle et d'apporter les moyens logiciels nécessaires à l'exploitation de supports matériels parallèles.

Par ailleurs, il semblait essentiel de montrer que le travail réalisé permettrait la réalisation d'études comparatives entre modélisations et données biologiques. Ce type d'études nécessite une maîtrise précise des éléments constitutifs des réseaux de neurones implémentés, de leurs paramètres ainsi qu'une grande flexibilité dans la récupération des données de simulation.

Les modèles de neurones impulsionnels et de réseaux actuellement utilisés et développés par la communauté scientifique ont été présentés au chapitre 1. Les caractéristiques et contraintes des méthodes de simulation de ces modèles par une stratégie événementielle ont été étudiées au chapitre 2. Enfin, des méthodes spécifiques aux simulations événementielles distribuées ont été introduites au chapitre 3.

Ce cadre de travail nous a permis de développer le simulateur DAMNED présenté au chapitre 4, un simulateur événementiel distribué et multithreadé qui permet l'utilisation de nombreux modèles de neurones impulsionnels de la littérature. Les modèles de synapses et les méthodes d'apprentissage locales des poids (de type plasticité synaptique) et des délais de transmission peuvent également être utilisées. Le simulateur DAMNED peut exploiter la plupart des supports parallèles actuels.

La réalisation et la simulation de réseaux de grande taille deviennent possibles grâce à ce travail comme l'a montré le chapitre 5. De plus le simulateur DAMNED accélère les temps de simulation des réseaux et les accélérations obtenues sont d'autant plus importantes que les réseaux sont étendus.

Enfin, le chapitre 6 a montré que l'utilisation DAMNED pour effectuer l'étude de réseaux d'inspiration biologique et permettre leurs comparaison à des comportements de populations neuronales connus est également possible. La proposition et l'étude d'un modèle entièrement connexionniste, composé de neurones impulsionnels, du système oculomoteur du tronc cérébral a permis de produire des saccades précises lors de stimulations d'une carte motrice du colliculus. Le modèle a également permis de formuler des hypothèses biologiquement plausibles quant à la boucle de rétroaction permettant l'arrêt des saccades lors du maintient de la stimulation.

Perspectives

De ce travail de thèse ont émergé de nombreuses pistes de recherche qui feront l'objet de futurs travaux. Les perspectives concernent plus particulièrement les fonctionnalités et performances du simulateur DAMNED, les études à venir utilisant le simula-

teur et le modèle du système saccadique proposé. Enfin, la réalisation et l'étude d'un modèle biologiquement plausible de colonne corticale sera envisagée.

Le simulateur DAMNED

Le simulateur DAMNED est prévu pour être utilisable par le plus grand nombre et le plus aisément possible. L'interface entre le simulateur et l'utilisateur doit donc être adaptée.

Interface

Une interface web est actuellement en développement. Celle-ci permet aux utilisateurs autorisés de se connecter pour effectuer des simulations en utilisant des ressources matérielles disponibles. L'interface permet également la création de fichiers de configuration de réseaux. Enfin, l'interface permet de lancer des simulations à distance et de récupérer les données enregistrées durant celle-ci. Cette interface est actuellement développée par un stagiaire de Master 2 lors d'un projet co-encadré avec le directeur de cette thèse.

Le simulateur DAMNED permet l'utilisation de nombreuses catégories de modèles de neurones. Il faudra être en mesure de définir de nouveaux modèles de neurones à partir de l'interface.

Visualiser les réseaux définis et le déroulement des simulations sera également un plus pour une interface, même si ce type de fonctionnalités a un impact fort sur les temps d'exécution et l'occupation mémoire, en particulier pour les réseaux de grande taille ($> 10^4$).

D'une façon générale, une interface graphique est l'outil idéal pour un accès aisé aux fonctionnalités du simulateur.

Occupation mémoire

L'utilisation de la mémoire est un paramètre déterminant dans les exécutions séquentielles. Lorsqu'un support parallèle est utilisable, son importance est moindre du fait que l'ajout de matériel peut pallier le manque de mémoire.

Cependant, ce critère reste important pour optimiser la taille des réseaux simulables, à ressources matérielles constantes.

Dans ce travail de thèse, nous avons constaté que l'occupation mémoire du simulateur DAMNED n'est pas optimale. Des redondances dans le stockage des connexions sont notamment dues aux tables de propriétés et aux tables de connexion des *threads* CMC du simulateur. De même, les modèles de neurones et de synapses utilisés pour ces tests n'ont pas été optimisés du point de vue de l'occupation mémoire.

Les événements transmis entre les LP sont des événements PPS contenant un couple événement PA/ neurone cible. Ceci implique qu'un neurone émettant vers plusieurs neurones hébergés par le même LP entraînera le stockage et la transmission sur le réseau d'autant d'événements PPS que de neurones cibles. Une optimisation de ce comportement consistera à transmettre des événements PA entre les LP. Ainsi dans l'exemple ci-dessus, un seul événement PA serait transmis entre deux LP, quelque soit le nombre d'événements PPS générés sur un LP par un PA.

Les accélérations observées pour les temps de création des réseaux ne sont pas optimales. L'occupation mémoire du simulateur implique une augmentation de l'espace nécessaire au stockage d'un même réseau lorsque l'on augmente le nombre de machines utilisées. De plus la parallélisation de la phase de création n'est pas optimale. Un parcours de toutes les connexions potentielles dans le réseau est effectué par chaque LP. Bien évidemment, seules les connexions utiles sont créées. Ces deux critères peuvent expliquer la faible accélération du temps de création obtenue pour des réseaux de très grande taille (10^5) sur un grand nombre de machines (> 30).

Un stagiaire de Master 1, en co-encadrement avec le directeur de cette thèse, étudie actuellement les optimisations possibles du stockage des variables de type entiers en mémoire dans le simulateur. En effet, une variable de type entier est codée sur 1 octet en mémoire. Cependant, sur les machines actuelles, l'adressage mémoire est effectué par mots de 32 bits soit 4 octets. Il est donc possible d'optimiser l'occupation mémoire due au stockage de variables entières en exploitant le fait que 4 entiers de 1 octet peuvent être stockés dans un espace mémoire de 32 bits.

Types d'événements

Le simulateur DAMNED est particulièrement adapté aux simulations de réseaux de neurones impulsionnels. Cependant, les simulateur à été développé dans un esprit de généricité lui garantissant une utilisation possible dans de nombreuses catégories de simulations événementielles.

Dans les deux cas, le version actuelle de DAMNED ne fournit aux ObjetEvenementiels (OE) implémenté qu'un seul type d'événement : des événements PPS. Dès lors, seule une discrimination par l'identité de l'OE source est possible pour les OE. Il est donc prévu de permettre aux OE de faire usage de différents types d'événements afin, par exemple, de spécialiser certains traitements. Dans le cas des neurones impulsionnels, seuls certains modèles sont exploitables à l'aide d'un seul type d'événements. En effet, nous avons vu au chapitre 2 que la stratégie événementielle impose qu'un neurone ne peut émettre que lors de la réception d'un événement. Or les modèles les plus riches, en termes de comportements biologiques reproductibles (comme le modèle d'Izhikevich ou de Hodking et Huxley), sont décrits par plusieurs équations différentielles couplées. Les stratégies mises en place pour implémenter ces modèles peuvent faire appel à des événements intermédiaires (n'indiquant pas des potentiels d'action) pour permettre le traitement.

Le simulateur DAMNED prendra donc en compte la possibilité (idéalement via l'interface) de définir et d'utiliser plusieurs types d'événements dont la gestion pourra être spécifique (e.g. priorité, localité, étiquette).

Mapping des réseaux

Les modèles de réseaux définis à l'aide du simulateur DAMNED peuvent avoir des topologies variées. La stratégie actuelle du simulateur pour répartir le réseau de neurones sur les processus logiques (LP) consiste à minimiser le nombre de neurones hébergés par chaque LP. Cette stratégie, simple d'implémentation, permet un équilibrage statique mais ne tient pas compte de la distribution synaptique, de la distribution des délais ni de la complexité des modèles de neurones.

Un mapping efficace pourra être implémenté en fonction de la définition des réseaux. Les spécifications des projections entre populations de neurones, de leurs poids respectifs ainsi que de leurs délais fournissent les informations nécessaires à une meilleure répartition des neurones.

De plus, le simulateur DAMNED permet l'utilisation de plusieurs modèles de neurones au sein d'une même simulation. La complexité des modèles utilisés pourra être utilisée pour effectuer une répartition optimisant la charge computationnelle de chaque LP. Par ailleurs la définition de modèles pourra être simplifiée par l'interface.

Scalabilité

De nouvelles simulations devront être effectuées :

1. Pour étudier plus précisément l'influence de la taille, de la connectivité et de l'activité des réseaux sur les temps d'exécution ;
2. Pour étudier les aptitudes du simulateur dans des exécutions à très grande échelle. Des réseaux de l'ordre du million de neurones, et un nombre de machines supérieur à 100 permettrait de mieux éclairer les performance du simulateur.

Dans ce but l'utilisation des plateformes de calcul parallèle de l'INRIA Sophia-Antipolis et de Nancy/Metz permettront d'établir les profils de performance sur des clusters dédiés de bonne qualité.

Études utilisant DAMNED

Le modèle du système oculomoteur dans le tronc cérébral développé au chapitre 6 est en développement dans le cadre du projet ANR MAPS. La conception et l'étude des évolutions du modèle présentées dans la discussion (section 6.7) donneront lieu à la soumission d'un article.

Un étudiant de Master 2 consacre son stage à l'étude des hyperparamètres du modèle de neurone LIF à l'aide d'algorithmes génétiques en utilisant DAMNED comme support de simulation et le modèle de contrôle oculomoteur dans des tests de production de saccades comme base d'étude.

La thèse de cet étudiant : Mr Sébastien Girier-Dufournier, débutant l'année prochaine sous réserve de financement, s'intéressera à l'utilisation du simulateur DAMNED dans des simulations événementielles distribuées non appliquées aux réseaux de neurones. Ces travaux seront consacrées à l'étude de modélisations de phénomènes tels que la circulation, les mouvements de population, la propagation d'épidémies, les phénomènes présentant des risques naturels (glissements de terrains, inondations, cyclones, etc.).

En effet DAMNED est conçu en tant qu'outil de simulation événementiel générique et son développement à été réalisé dans ce but. Cette thèse permettra au simulateur de faire la preuve de son adéquation avec les problématiques des simulation événementielles distribuées en général.

Annexes

Annexe A : Caractéristiques des simulateurs

La légende de la figure **??** du chapitre 2 présentant un comparatif de différents simulateurs disponibles dans la littérature est détaillée ci-après :

HH	Permet l'utilisation du modèle HH,
leaky IF	Permet l'utilisation du modèle Intègre et Tire à Fuite,
Izhikevich IF	Permet l'utilisation du modèle d'Izhikevich,
Cable Eqs	Permet l'utilisation de modèles à compartiments,
ST plasticity	Permet l'utilisation de plasticité à court terme,
LT plasticity	Permet l'utilisation de plasticité à long terme,
Event-based	Permet l'utilisation d'une stratégie événementielle,
exact	Simulation exacte,
Clock-based	Permet l'utilisation d'une stratégie basée sur une horloge,
interpolated	Interpolation des spikes,
G synapses	Permet l'utilisation de modèles synaptiques à conductances,
parallel	Permet l'utilisation d'un support parallèle,
graphics	Dispose d'une interface graphique,
simple analysis	L'interface permet-elle des analyses simples ?,
complex analysis	Permet des analyses plus complexes ?,
development	Simulateur en développement,
how many p.	Combien de développeurs actifs ?,
support	Y a-t-il un support utilisateur ?,
type	Type de support,
user forum	Y a-t-il un forum ou adresse mail pour les utilisateurs ?,
publ list	Liste de publications utilisant le simulateur,
codes	Codes des modèles disponibles en ligne
online manual	Manuel en ligne,
book	Livres publiés sur le simulateur,
XML import	Permet l'importation de spécifications XML,
XML export	Permet l'exportation de spécifications XML,
web site	Y a-t-il un site internet mettant à disposition les supports,
LINUX	Le simulateur fonctionne sous LINUX,
Windows	Le simulateur fonctionne sous Windows,
Mac-Os	Le simulateur fonctionne sous Mac-Os X,
Interface	Peut interfacer le simulateur avec des

	outils extérieurs ? (capteurs, caméras, neurones ...),
Save option	Permet l'interruption/sauvegarde des simulations.
B.I.	Intégré dans le simulateur,
YES	Facile à implémenter,
POSS	Possible à implémenter mais nécessite un peu de programmation,
NO	Non implémenté, code du simulateur à modifier,
**	Implémentation prévue dans de futures versions.
(*)	Interface graphique intégrable avec des outils tels que Python ou Matlab.

Question	NEURON	GENESIS	NEST	NCS	CSIM	XPP	SPLIT	Mvaspike	DAMNED
HH	B.I.	B.I.	YES	B.I.	B.I.	YES	B.I.	POSS	POSS**
leaky IF	B.I.	POSS	YES	B.I.	B.I.	YES	POSS**	B.I.	B.I.
Izhikevich IF	YES	B.I.	YES	NO	B.I.	YES	POSS**	POSS**	POSS**
Cable eqs	B.I.	B.I.	NO	NO	NO	YES	B.I.	NO	NO
ST plasticity	YES	B.I.	YES	B.I.	B.I.	YES	B.I.	YES	YES
LT Plasticity	YES	YES	YES	B.I.	B.I.	YES	NO**	YES	YES
Event-based	B.I.	NO	YES	NO	NO	YES	NO	YES	YES
exact	B.I.	-	YES	-	-	NO	-	YES	YES
Clock-based	B.I.	B.I.	YES	B.I.	YES	YES	YES	POSS**	NO
interpolated	B.I.	NO	YES	NO	NO	YES	B.I.	POSS	NO
G synapses	B.I.	B.I.	YES	B.I.	B.I.	YES	B.I.	POSS**	YES
parallel	B.I.	YES	B.I.	B.I.	NO**	NO	B.I.	NO**	B.I.
graphics	B.I.	B.I.	NO(*)	NO(*)	NO(*)	YES	NO	NO	NO**
simple analysis	B.I.	B.I.	YES	NO(*)	NO(*)	YES	NO	NO	NO
complx analysis	B.I.	YES	NO(*)	NO(*)	NO(*)	YES	NO	NO	NO
development	YES	YES	YES	YES	YES	YES	YES	YES	YES
how many p.	3	2-3	4	2-3	2	1	2	1	1
support	YES	YES	YES	YES	YES	YES	YES	YES	YES
type	e,p,c	e	e	e	e	e	e	e	e
user forum	YES	YES	YES	NO	NO	YES	YES	NO	NO**
publ list	YES	YES	YES	YES	YES	NO	NO	NO	NO
codes	YES	YES	YES	YES	YES	YES	NO	NO	NO**
online manual	YES	YES	YES	YES	YES	YES	YES	YES	NO**
book	YES	YES	NO	NO	NO	YES	NO	NO	NO
XML import	NO**	POSS	NO**	NO**	NO	YES	NO	NO**	NO
XML export	B.I.	NO**	NO**	NO**	NO	NO	NO	NO**	NO
web site	YES	YES	YES	YES	YES	YES	YES	YES	NO**
LINUX	YES	YES	YES	YES	YES	YES	YES	YES	YES
Windows	YES	YES	YES	YES	YES	YES	NO	NO	NO
Mac-Os	YES	YES	YES	NO	NO	YES	NO	NO	NO
Interface	B.I.	B.I.	POSS	B.I	YES	POSS	POSS	POSS	YES
Save option	B.I.	YES	NO**	B.I.	NO	NO	NO	NO	NO**

FIG. 22 – *Récapitulatif des fonctionalités des simulateurs existants, ainsi que du simulateur DAMNED. (Modifié de (Brette et al. 2007))*

La réalisation du simulateur DAMNED dans ce travail de thèse nous permet de mettre à jour la figure 2.1 en y ajoutant les fonctionalités de DAMNED, voir figure 22.

Annexe B : DAMNED, Évaluation

Modèle de neurone utilisé

Le modèle de neurone utilisé pour ces tests est un modèle simple d'intégrateur à fuite. Il est basé sur l'équation 1.9. Ce modèle est présenté en détail dans La solution de l'équation à la date t_0 est une forme exponentielle. Si on considère $\hat{t}$, la date du dernier PA émis par le neurone et si on est pas en période réfractaire :

$$t_0 - \hat{t} > \tau_r, \tag{1}$$

où $\tau_r = 1ms$ est la période réfractaire utilisée. Alors, si t^{pre} est la date d'impact du dernier PPS arrivé sur la membrane, la fuite du potentiel de membrane V_m^L peut s'exprimer comme :

$$V_m^L(t_0) = V_r + (V_m(t^{pre}) - V_R) \times e^{\frac{-(t_0 - t^{pre})}{\tau_m}} \tag{2}$$

où $V_R = -60 \times 10^{-3}$ est le potentiel de repos, $V_m(t^{pre})$ est la valeur du potentiel à la date du dernier PPS reçu. La constante de temps de la membrane vaut $\tau_m = 20ms$. Le potentiel de repos est aussi potentiel d'inversion de la membrane.

Il faut ensuite ajouter l'influence synaptique avec :

$$\Delta V_m^S(t_0) = p \times g_s \times (V_m^L(t_0) - V_R^S) \tag{3}$$

où p est le poids de la synapse recevant le PPS, g_s est la conductance synaptique. Elle peut être inhibitrice $g_{si} = 67 \times 10^{-3}$ ou excitatrice $g_{se} = 6 \times 10^{-3}$ selon la synapse. V_R^S est le potentiel d'inversion synaptique. Il peut, lui aussi être inhibiteur $V_R^{S_i} = -500 \times 10^{-3}$ ou excitateur $V_R^{S_e} = 0$.

Ainsi la réception d'un PPS à t_0 engendre un changement de la valeur du potentiel :

$$V_m(t_0) = V_m^L(t_0) + \Delta V_m^S(t_0) \tag{4}$$

Le seuil d'émission d'un PA est de $\theta = -50 \times 10^{-3}$.

Indicateurs temporels

Les indicateurs temporels utilisés pour l'évaluation du simulateur DAMNED sont chacun composés de deux fonctions. La première démarre le « chronomètre » spécifique de cet indicateur, la seconde le stoppe. La position exacte des appels à ces fonctions dans le code source du programme a donc une influence sur les valeurs obtenues.

Temps total d'exécution

Le temps total d'exécution T_T est enregistré depuis l'entrée dans la boucle tantque des algorithmes CPC et CMC jusqu'à sa terminaison. La valeur enregistrée est donc représentative du déroulement de la simulation dans sa totalité.

Les temps d'initialisation du réseau physique et de création du réseau de neurones ne sont pas pris en compte par cet indicateur.

Temps total d'activité / d'inactivité

L'indicateur de temps total d'activité T_A est démarré en même temps que T_T. Lorsque le *thread* considéré (CPC ou CMC) n'a plus rien à faire, il entre en pause et l'indicateur T_A est stoppé et l'indicateur de temps total d'inactivité T_I est démarré. Dès que la pause se termine (des traitements à effectuer), T_I est stoppé et T_A est de nouveau démarré.

Temps de traitement d'un événement

Le *thread* CPC, respectivement CMC, effectue un traitement sur les événements qui consiste à contrôler la date et fournir (si autorisé) l'événement à l'OE cible et effectuer le traitement de l'événement par la cible, respectivement contrôler la date et créer (si autorisé) les paires événement/cible à envoyer.

Pour le CPC, l'indicateur de temps de traitement d'un événement T_t^{CPC} est démarré à l'appel de la fonction *lanceTraitementEvenement()* et est stoppé après celle-ci. Cette fonction comprend les traitements suivants :

- la récupération du prochain événement à traiter depuis la fet
- les contrôles autorisant le traitement ou non selon le temps virtuel global
- la suppression éventuelle de l'événement de la file
- la récupération éventuelle de l'OE cible
- le traitement éventuel de l'événement par l'OE cible
- la récupération de l'événement éventuellement créé en retour par la cible
- la mise à jour éventuelle du temps virtuel local

Pour le CMC, l'indicateur de temps de traitement d'un événement T_t^{CMC} est démarré à l'appel de la fonction *traiteProchainEvenementAEmettre()* et est stoppé après celle-ci. Cette fonction comprend les traitements suivants :

- la récupération de l'événement depuis la file
- les contrôles autorisant l'émission ou non selon le temps virtuel global
- la destruction éventuelle de l'événement en file
- la mise en paquets éventuelle des paires événement/cible générées par l'émission de l'événement vers les OE cibles.
- la mise à jour éventuelle du temps virtuel local

Temps d'ajout d'un événement en file

L'indicateur de temps d'ajout d'un événement en file T_a est démarré juste avant l'insertion d'un nouvel événement en file et stoppé juste après. Le CPC insère des événements dans fee et CMC insère des paires événement/cible dans la fet. Les coûts sont donc différents entre CPC et CMC.

Temps de récupération d'un événement

Les indicateurs de temps de récupération d'un événement T_r, sur CPC et CMC sont démarrés lors la récupération d'un événement (pour CMC) ou d'une paire événement/cible (pour CPC) et sont stoppés immédiatement après.

Temps de mise en paquets

Lors de l'émission d'un événement, une fois l'événement extrait, l'indicateur de temps de mise en paquets T_p (uniquement pour le CMC) est démarré avant la création des paires événement/cible et leur ajout dans les paquets spécifiques de chaque LP de la simulation. L'indicateur est stoppé dès les ajouts terminés.

Temps d'émission de paquets

L'indicateur de temps d'émission de paquets T_e est démarré juste avant l'étape d'émission de messages (dans l'algorithme du CMC), et stoppé juste après.

Cette étape contient la mise à jour des requêtes d'envoi et des buffers d'envoi associés à chaque LP. Pour les LP dont les précédentes requêtes d'envoi sont terminées, les paquets en attente sont envoyés (transférés dans les buffers d'envoi, ou dans la file locale) et les événements émis sont détruits.

Annexe C : Modèle de Saccades

Paramètres du simulateur DAMNED

Outre les paramètres propres à la simulation, le simulateur DAMNED est utilisé avec un jeu de paramètres internes. La précision de la simulation effectuée est de $10^{-1}ms$. Le délai minimum dans une simulation est de $d_{min} = 1ms$. Le délai maximum est fixé à $d_{max} = 100ms$. Le délai de base d'une connexion est de $d_{base} = 4ms$. Les délais d_{ij} sont ensuite déterminés, à la création, avec une variabilité telle que :

$$d_{min} < d_{base} - \Delta d_{min} < d_{ij} < d_{base} + \Delta d_{max} < d_{max} \quad (5)$$

Δd_{min} et Δd_{max} sont les variations maximum et leur valeur est fixée à $\Delta d_{min} = \Delta d_{max} = 1ms$

De la même manière, les valeurs des poids peuvent varier au moment de leur création selon :

$$0 < poids_{base} - \Delta p_{min} < poids_{ij} < poids_{base} + \Delta p_{max} \quad (6)$$

La possibilité d'utiliser des *threads* pour chaque traitement neuronal n'est pas exploitée pour cette série de simulations.

Les simulations ont été réalisées sur une machine bi-processeurs, à l'aide de 4 noeuds MPICH. Chaque simulation a une durée de l'ordre d'une trentaine de secondes.

Les files de priorité considèrent les événement inhibiteurs prioritaires sur les excitateurs, pour une même date.

Paramètres du modèle de neurone LIF utilisé

Le modèle de neurone utilisé est le même que celui décrit pour les tests préliminaires du simulateur (voir section 6.8).

Choix des valeurs des paramètres

Les paramètres ont été obtenus empiriquement par simulations successives du modèle. Les influences des paramètres sont inter-dépendantes et il est donc difficile d'en décrire les influences spécifiques. Cependant, les valeur des potentiels d'inversion ont une influence sur les particulière sur l'impact potentiel de la variation de l'intensité de la stimulation. En effet si le poids de la stimulation, et donc l'amplitude du PPS induit, est plus importante que la différence entre le potentiel de membrane et le potentiel d'inversion au moment de l'impact, alors une influence excitatrice devient inhibitrice et inversement. Ainsi l'augmentation de l'intensité de la stimulation, est limitée par ce paramètre.

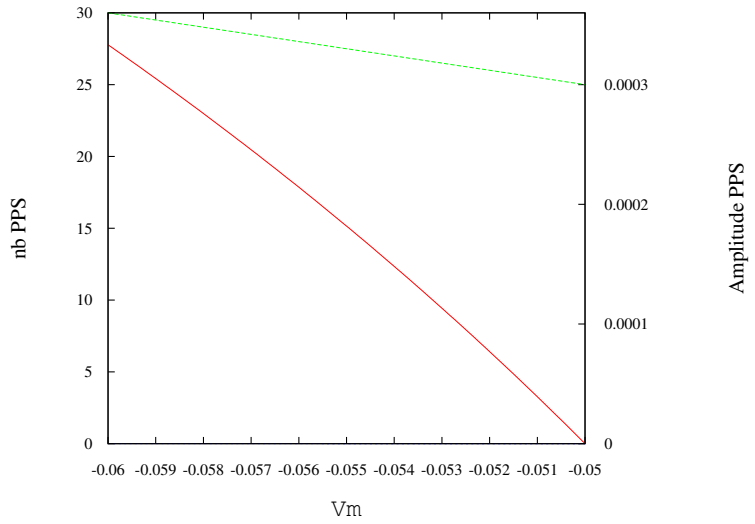

FIG. 23 – *Nombre de PPS excitateurs (axe de droite) nécessaires pour franchir le seuil et amplitude de l'impact d'un PPS (axe de gauche), en fonction du potentiel de membrane V_m.*

Par exemple, si $Vm = Vrest$ et $p = 1$ alors l'impact excitateur vaut $\Delta V_m^e = 36 \times 10^{-5}$ (l'impact inhibiteur vaut $\Delta V_m^i = -16 \times 10^{-3}$). Or $V_r - \theta = 10 \times 10^{-3}$ et le potentiel d'inversion excitateur $V_R^e = 0$. Il faut donc $n_{pps} = \frac{10 \times 10^{-3}}{36 \times 10^{-5}} = 28$ PPS excitateurs (15 PPS inhibiteurs pour arriver au potentiel d'inversion V_R^e) pour provoquer l'émission d'un PA. Cependant la différence $V_m - \theta$ évolue. La figure montre l'évolution du nombre de PPS n_{pps} selon l'équation (7), pour un potentiel de membrane compris entre le potentiel de repos $V_r = -60 \times 10^3$ et le seuil $\theta = -50 \times 10^{-3}$. L'amplitude de l'impact des pps reçus est également indiquée (courbe verte). On remarque qu'entre la valeur de repos et la valeur seuil, l'influence de la variation de l'amplitude des PPS est négligeable. Cependant cette variation est plus importante dans les valeurs de potentiel de membrane proches des potentiels d'inversion.

$$n_{pps}(V_m) = \frac{(\theta + V_m)}{g_{se} \times (V_m - V_R^e)} \tag{7}$$

L'influence de la variation des poids des connexions entre les populations de neurones indiqués ci-après n'a pas été étudiée. En particulier, les relations entre les projections de EBN, cMRF et OPN permettant de maintenir une boucle saccadique devraient être étudiées dans le détail pour en apprendre plus sur le fonctionnement de la boucle de rétroaction dans ce modèle du système oculomoteur au sein du tronc cérébral.

Paramètres des projections du CS

Les projections excitatrices internes au CS sont de type champs récepteurs. Une gaussienne excitatrice, centrée sur la cellule n_i, répartit les poids excitateurs p_{ij} internes au CS.

$$p_{ij} = e^{\frac{-((d_{ij}-\mu)\times(d_{ij}-\mu))}{2\times\sigma^2}} \tag{8}$$

d_{ij} est la distance entre les indices du neurone pré-synaptique n_i et des n_j neurones post-synaptiques. Le tableau d'indices dans l'ordre croissant fourni une carte unidimensionnelle pour le CS.

Une gaussienne excitatrice est également utilisée pour la répartition des poids entre les cellules d'entrées stimulatrices et les neurones du CS.

Stimulations du colliculus

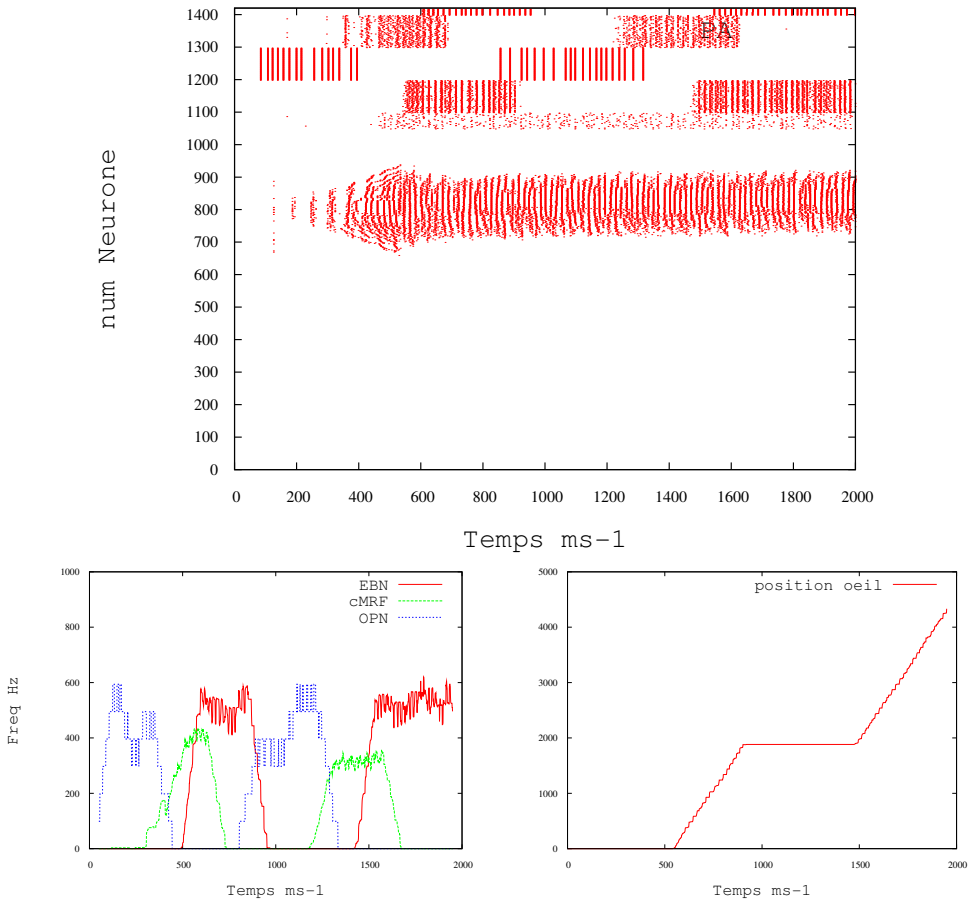

FIG. 24 – *Stimulation du Modèle colliculaire. Position 35. (Rostral - Min). Raster , fréquences des EBN, de la cMRF et des OPN (à gauche), et position de l'oeil (à droite).*

Le modèle est stimulé en 7 positions. Les figures qui suivent correspondent chacune à la réponse du modèle à une stimulation dans les 7 positions suivantes :

1. Position 5, centrée sur le neurone numéro 135 (sur les rasters).
2. Position 10, centrée sur le neurone numéro 245.
3. Position 15, centrée sur le neurone numéro 355.
4. Position 20, centrée sur le neurone numéro 465.
5. Position 25, centrée sur le neurone numéro 575.
6. Position 30, centrée sur le neurone numéro 685.

7. Position 35, centrée sur le neurone numéro 795.

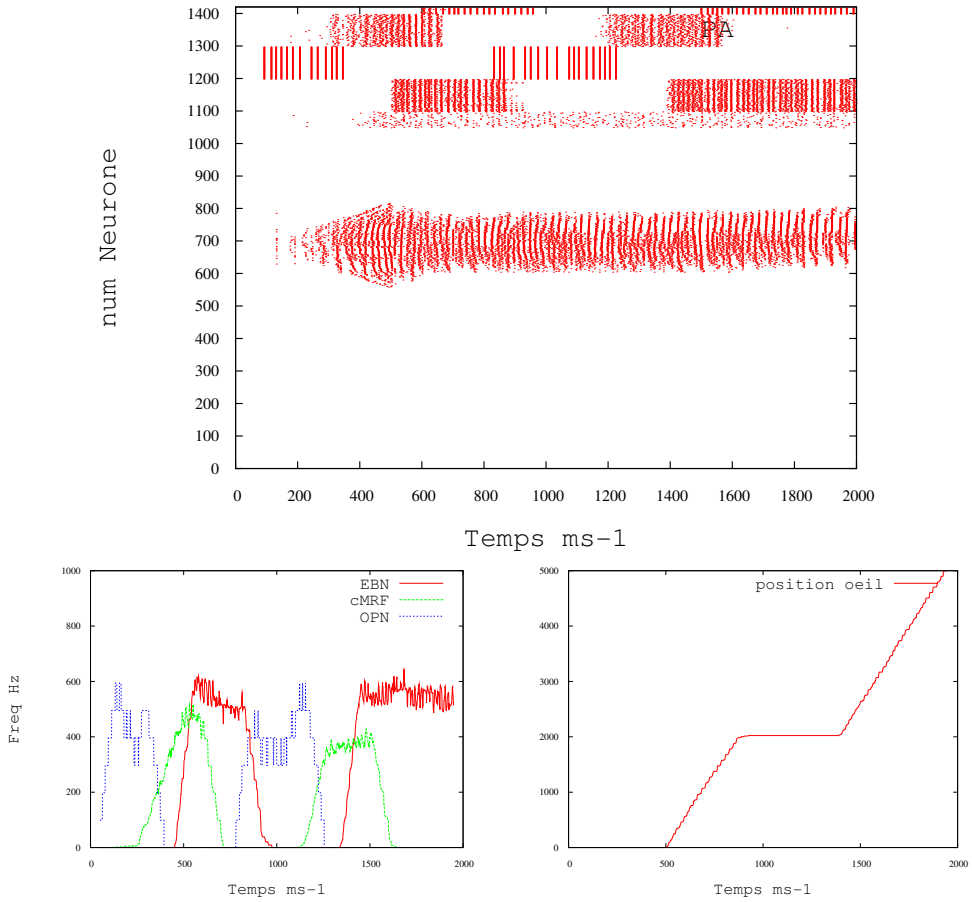

FIG. 25 – *Stimulation du Modèle colliculaire. Position 30. (Rostral - faible). Raster , fréquences des EBN, de la cMRF et des OPN (à gauche), et position de l'oeil (à droite).*

Sur chaque figure, on trouve le raster complet. De bas en haut, les émissions du colliculus ; jusqu'à la ligne 1047, les émissions des inhibiteurs du Colliculus ; jusqu'à la ligne 1097, les émissions des EBN ; jusqu'à la ligne 1197, les émission des OPN ; jusqu'à la ligne 1297, les émissions de la cMRF ; jusqu'à la ligne 1397, et enfin les émissions des inhibiteurs de la cMRF ; jusqu'à la ligne 1417.

En bas à gauche des figures se trouve le tracé de la fréquence de décharge des neurones EBN (en rouge), cMRF (en vert) et OPN (en bleu). La fréquence de décharge est calculée à l'aide fenêtre glissante de 10ms.

La troisième courbe représente la position de l'oeil. Elle est obtenue par intégration de tous les spikes émis par les neurones EBN.

Variations de l'intensité

Pour les stimulations concernant la variation de l'intensité des stimulations, les figures qui suivent contiennent les réponses du modèle pour à deux positions de stimulation (10,30). Chacune des positions est stimulée dans les deux conditions $1,3 \times T$

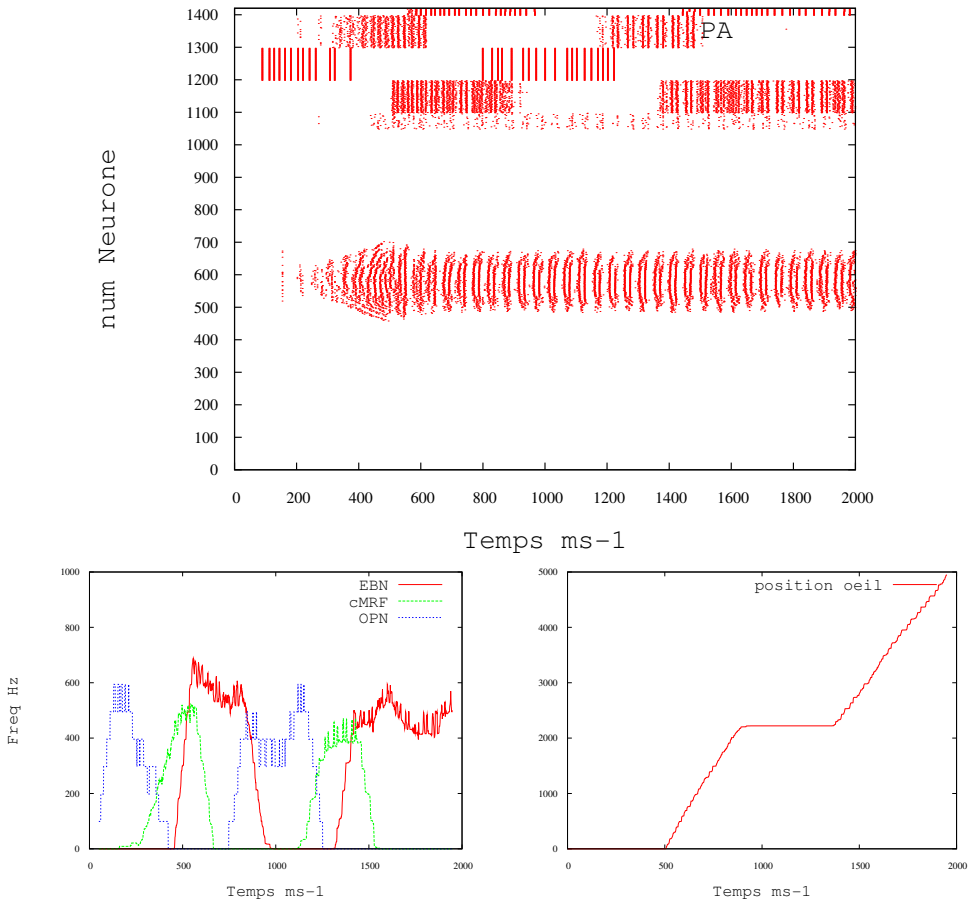

FIG. 26 – *Stimulation du Modèle colliculaire. Position 25. (Médian - moyen). Raster , fréquences des EBN, de la cMRF et des OPN, et position de l'oeil.*

et $6 \times T$. Pour la condition $2 \times T$, on conserve les réponses obtenues lors de la première partie de l'expérience.

Les figures 31 à 34 concernent les réponses du modèles lorsque l'on fait varier la taille de la population stimulée dans le CS.

Variations de la fréquence de stimulation

Les figures **??** à 40 contiennent les réponses du modèle aux variations de fréquences de la stimulation. Le raster, les fréquences EBN, cMRF et OPN ainsi que la position de l'oeil sont indiquées. Les positions de stimulation 10 et 30 sont utilisées pour appliquer ces variations.

Les fréquences 150, 300, 500, et 750 ont été utilisées pour la stimulation. L'intensité de la stimulation est maintenue à $2 \times T$.

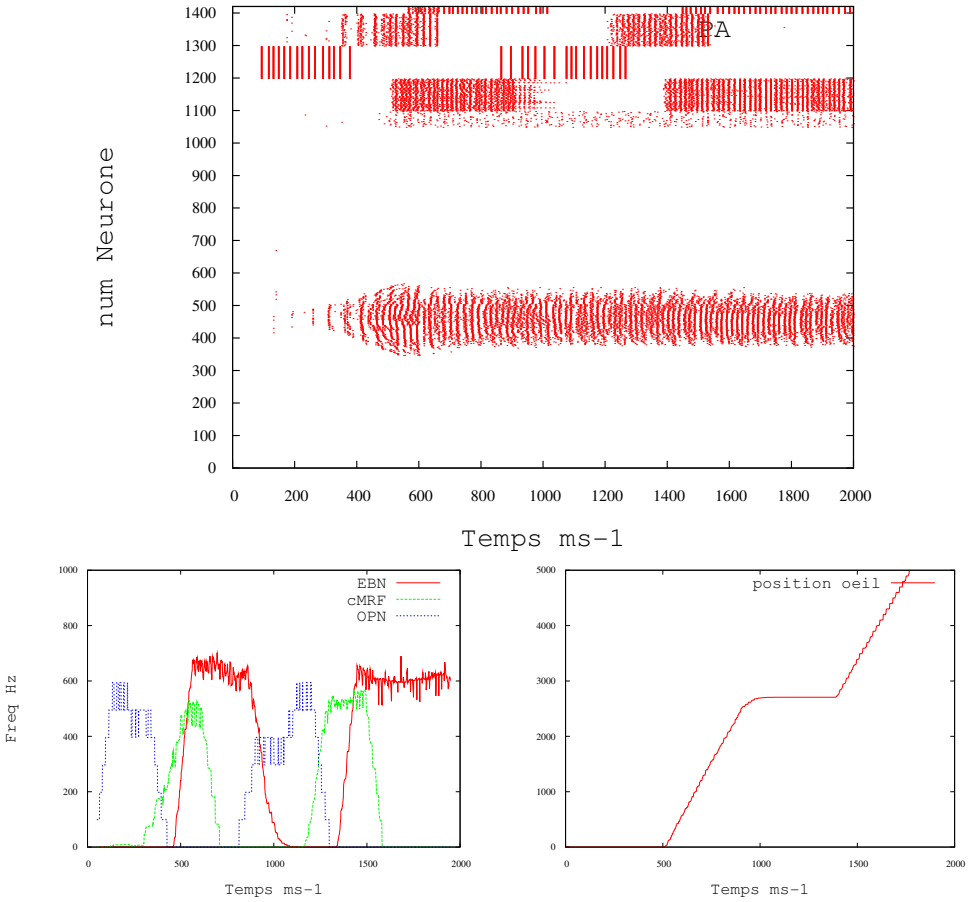

FIG. 27 – *Stimulation du Modèle colliculaire. Position 20. (Médian - moyen). Raster , fréquences des EBN, de la cMRF et des OPN, et position de l'oeil.*

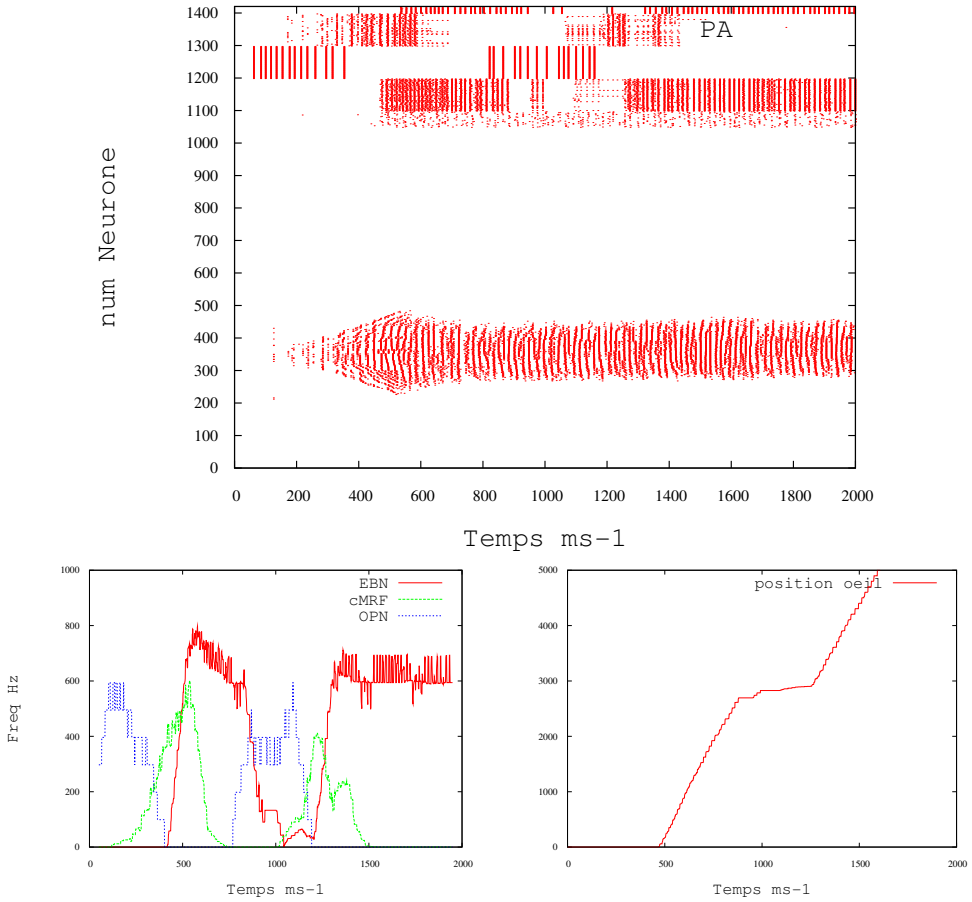

FIG. 28 – *Stimulation du Modèle colliculaire. Position 15. (Caudal - fort). Raster , fréquences des EBN, de la cMRF et des OPN, et position de l'oeil.*

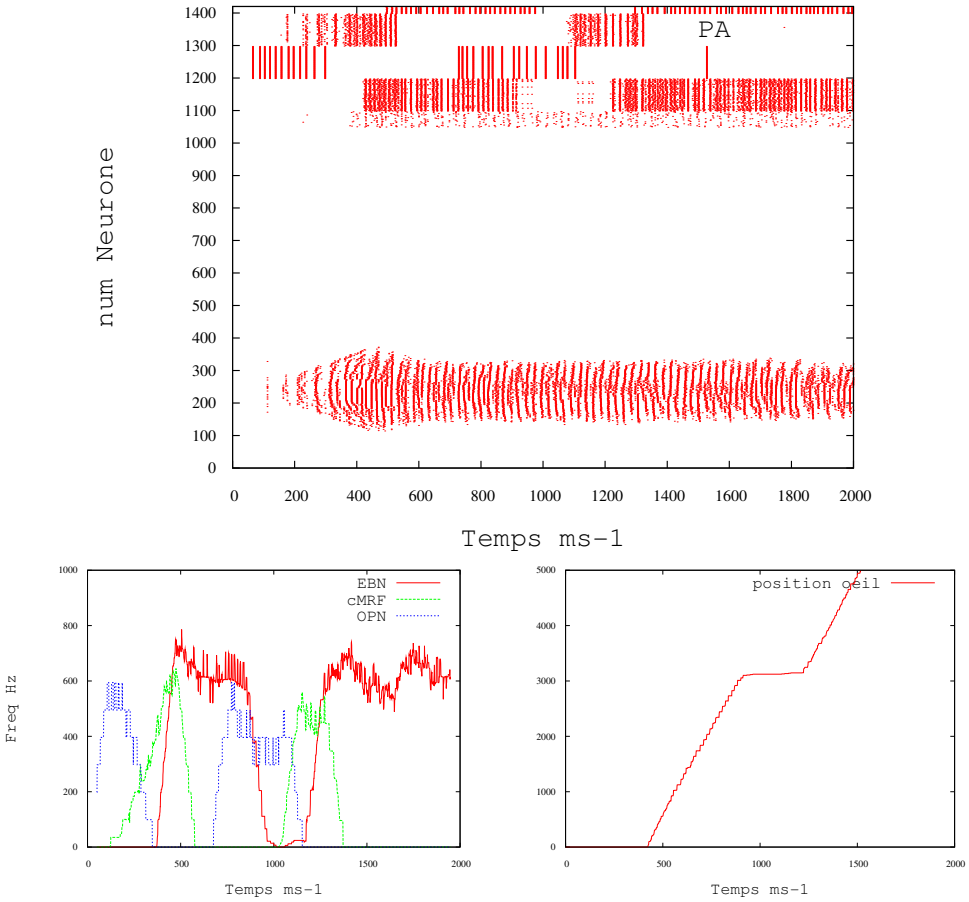

FIG. 29 – *Stimulation du Modèle colliculaire. Position 10. (Caudal - fort). Raster , fréquences des EBN, de la cMRF et des OPN, et position de l'oeil.*

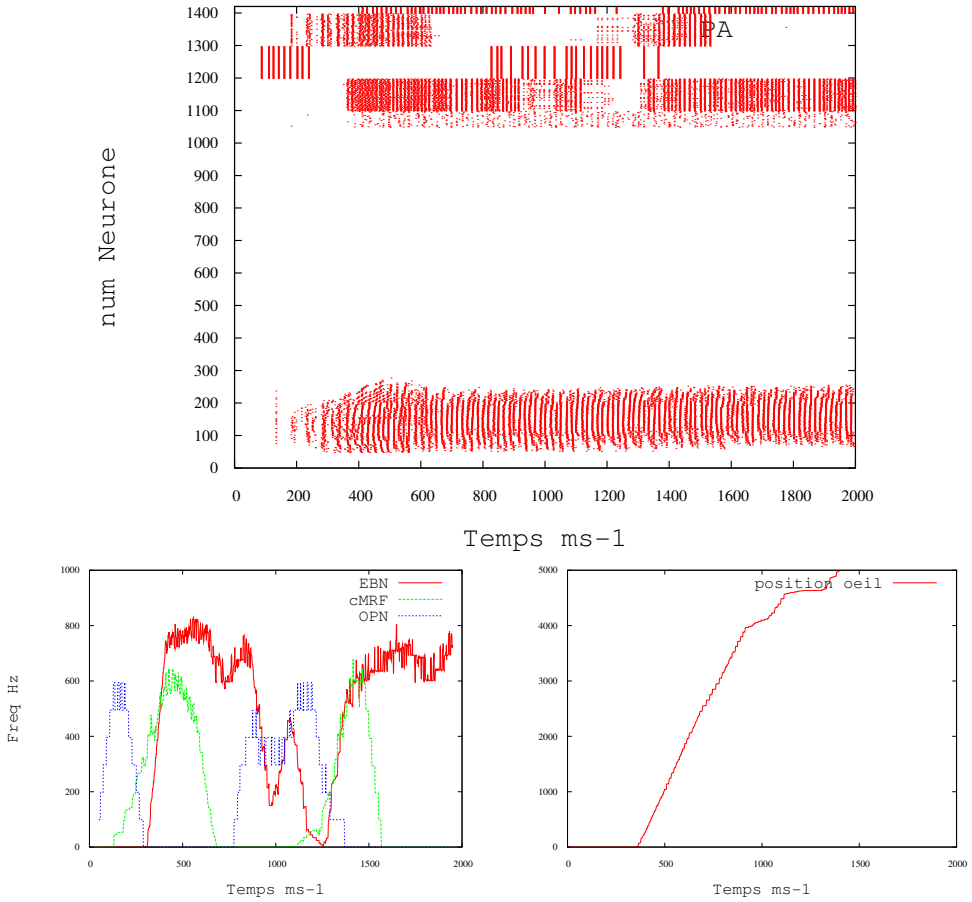

FIG. 30 – *Stimulation du Modèle colliculaire. Position 5. (Caudal - max). Raster , fréquences des EBN, de la cMRF et des OPN, et position de l'oeil.*

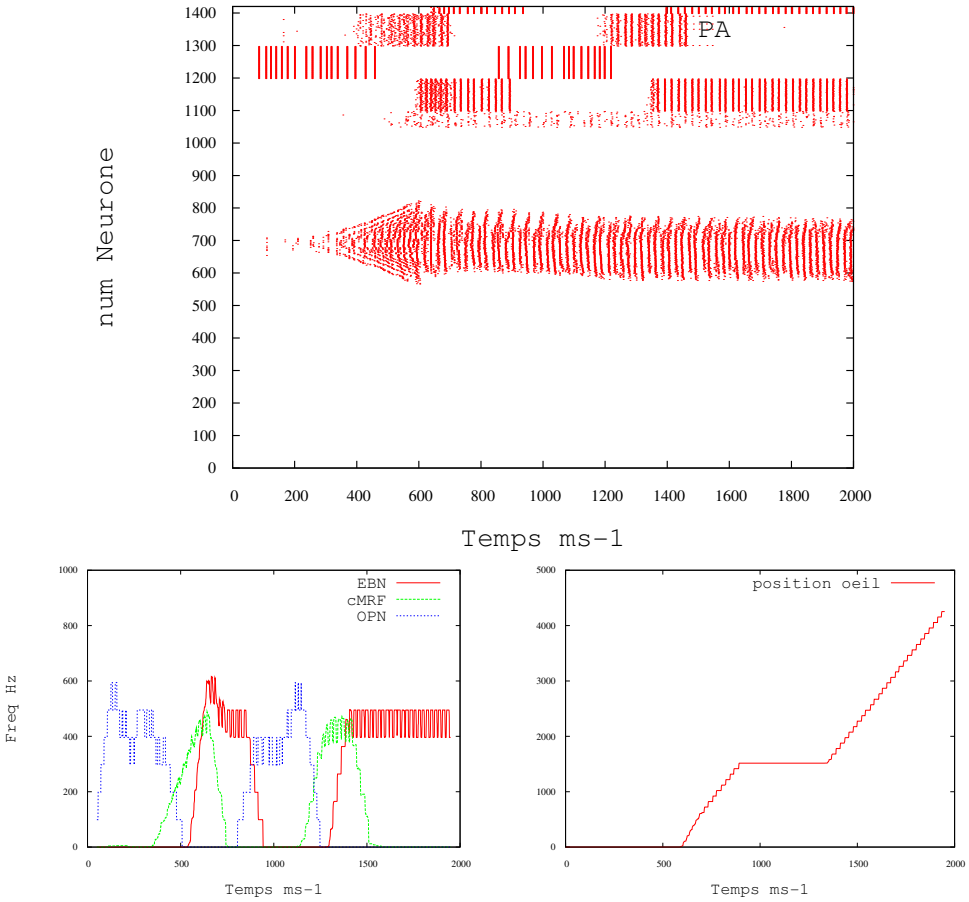

FIG. 31 – *Stimulation du Modèle colliculaire. La stimulation est appliquée en position 30. L'intensité de la stimulation est de* $1,3 \times T$.

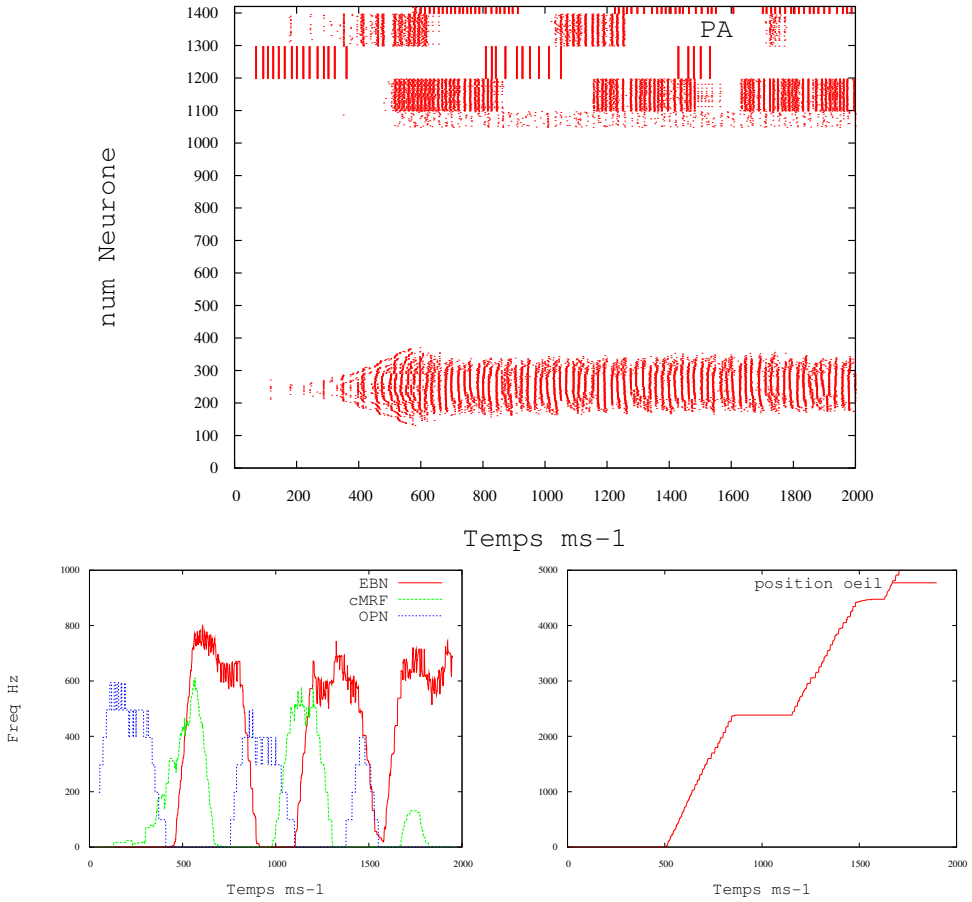

FIG. 32 – *Stimulation du Modèle colliculaire. La stimulation est appliquée en position 10. L'intensité est de* $1,3 \times T$.

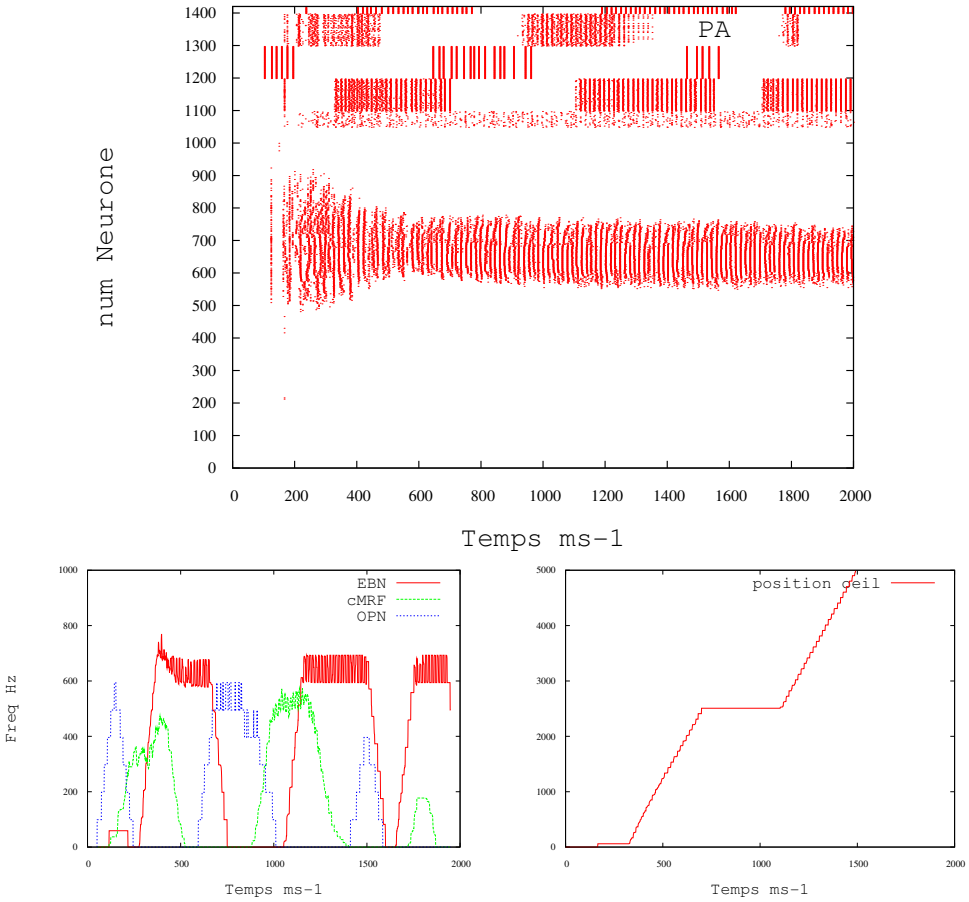

FIG. 33 – *Stimulation du Modèle colliculaire. La stimulation est appliquée en position 30. L'intensité est de* $6 \times T$.

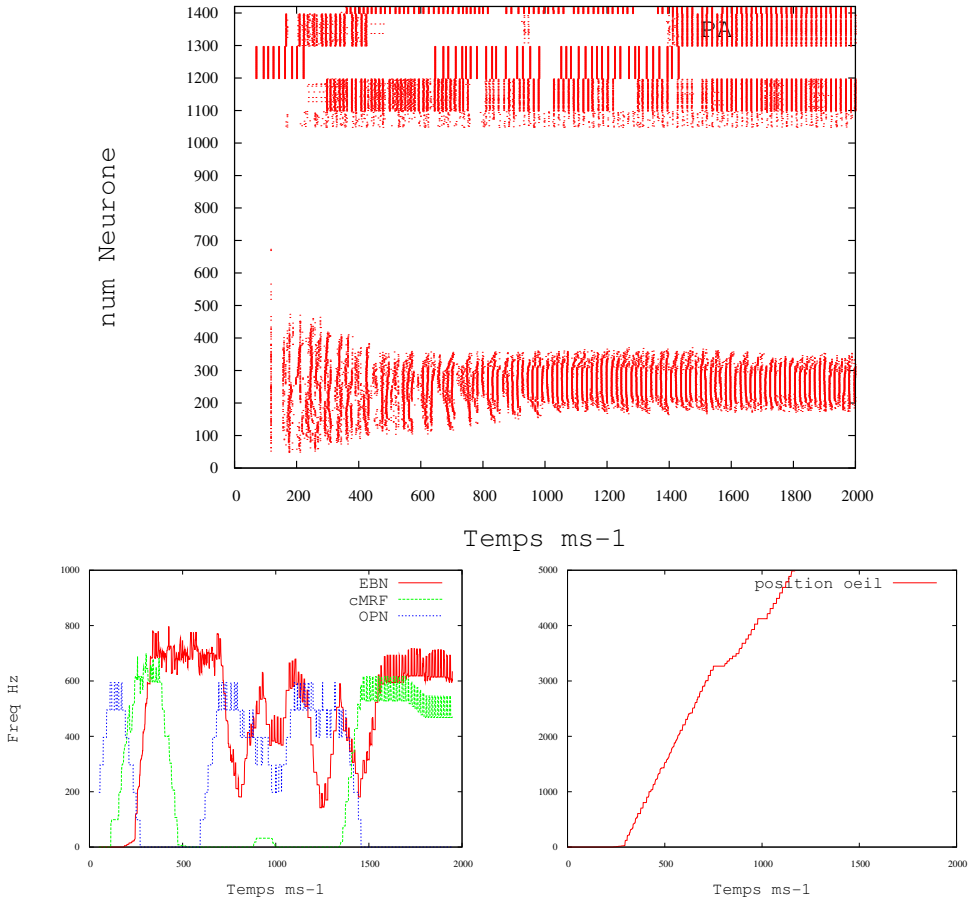

FIG. 34 – *Stimulation du Modèle colliculaire. La stimulation est appliquée en position 10. L'intensité est de* $6 \times T$.

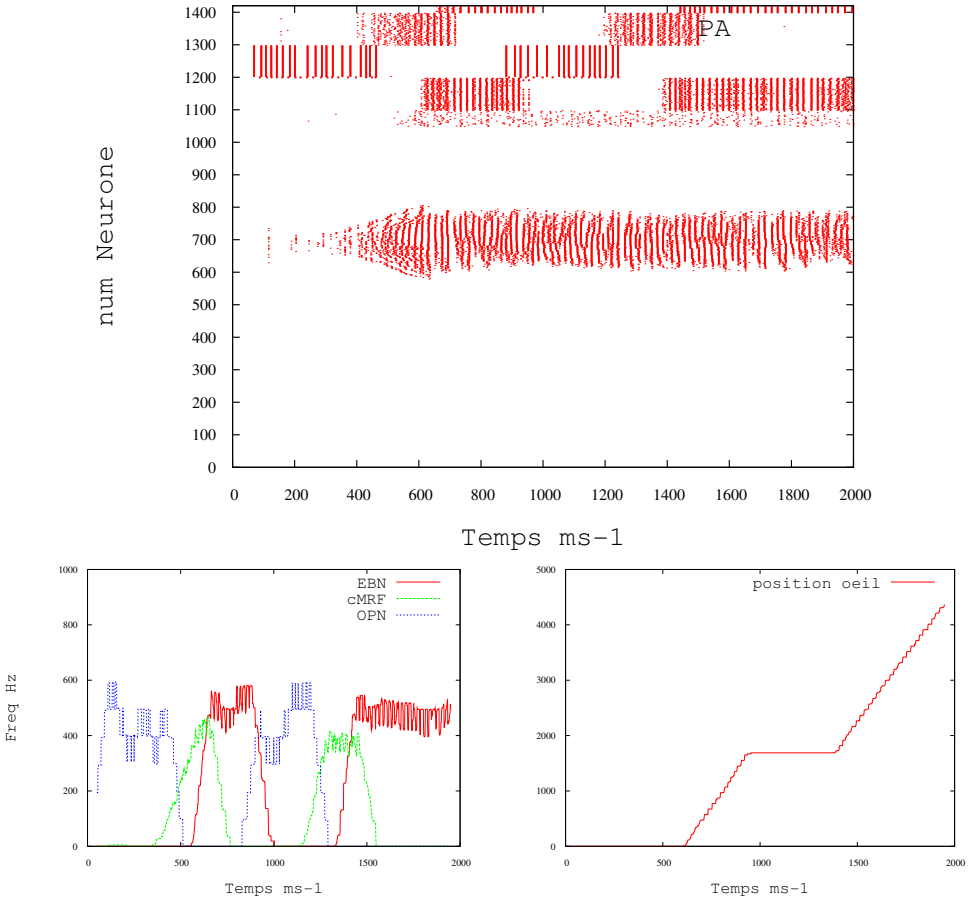

FIG. 35 – *Stimulation colliculaire en position 30. Fréquence de stimulation* 150*Hz*

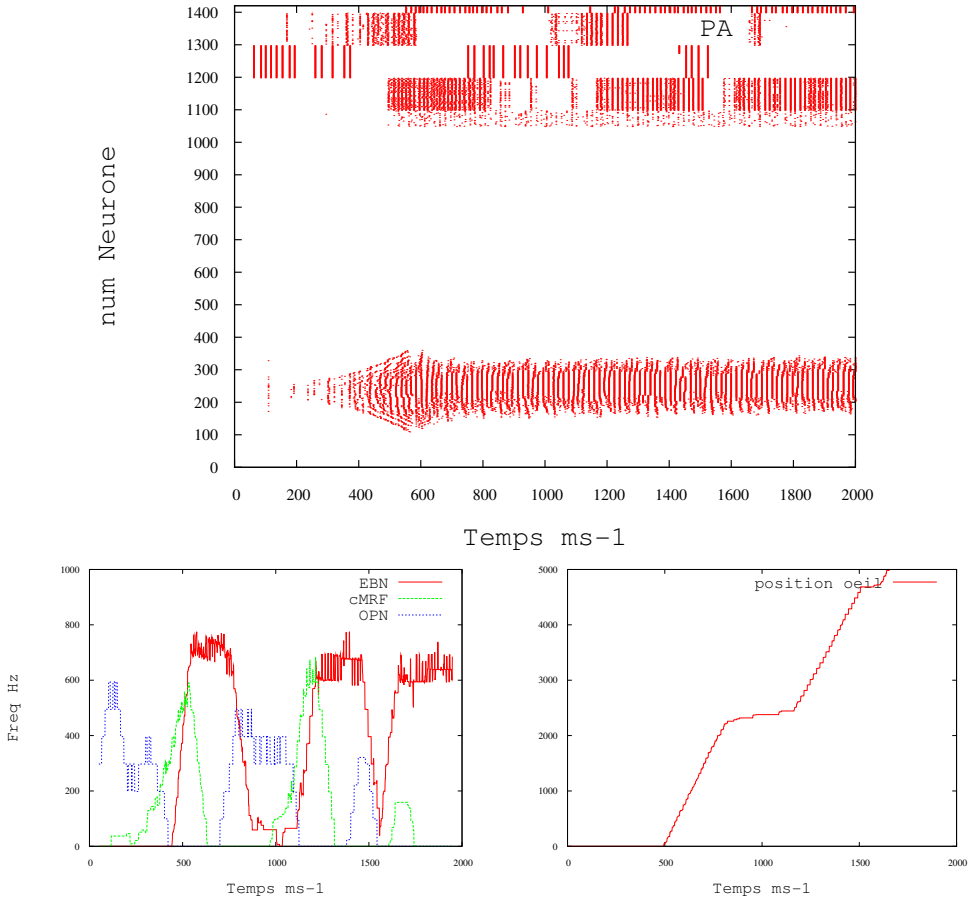

FIG. 36 – *Stimulation colliculaire en position 10. Fréquence de stimulation* 150*Hz*

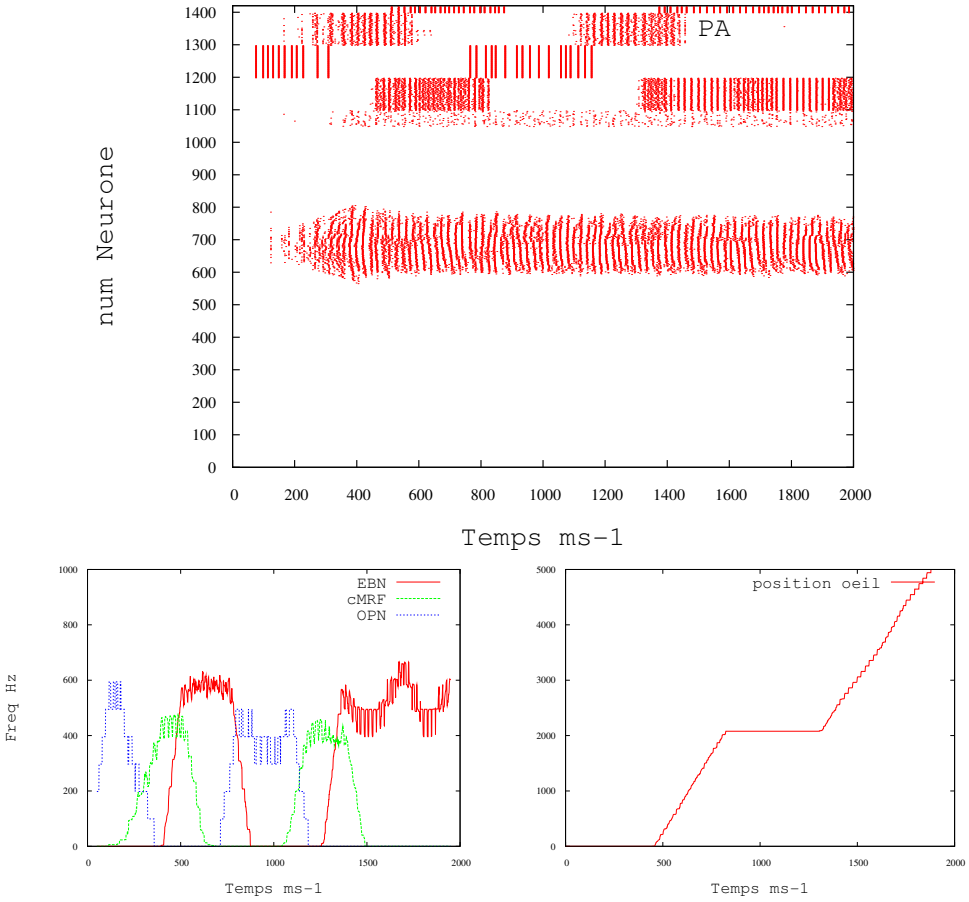

FIG. 37 – *Stimulation colliculaire en position 30. Fréquence de stimulation* 500*Hz*

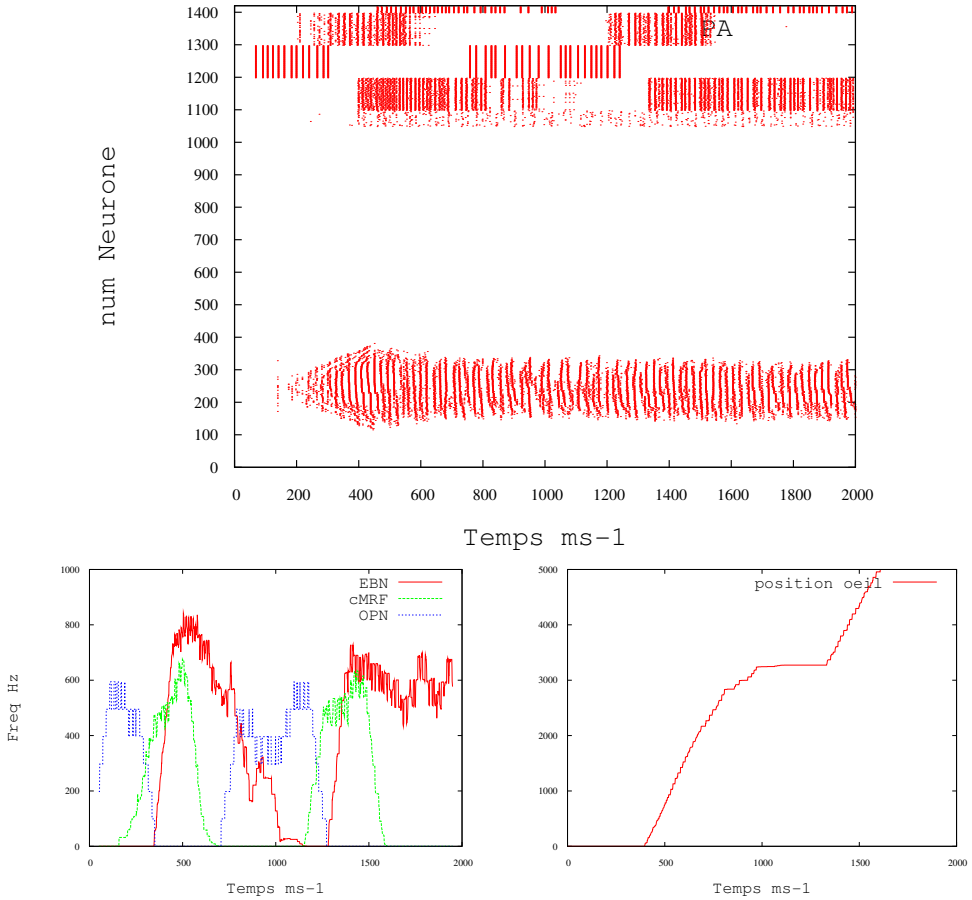

FIG. 38 – *Stimulation colliculaire en position 10. Fréquence de stimulation* 500*Hz*

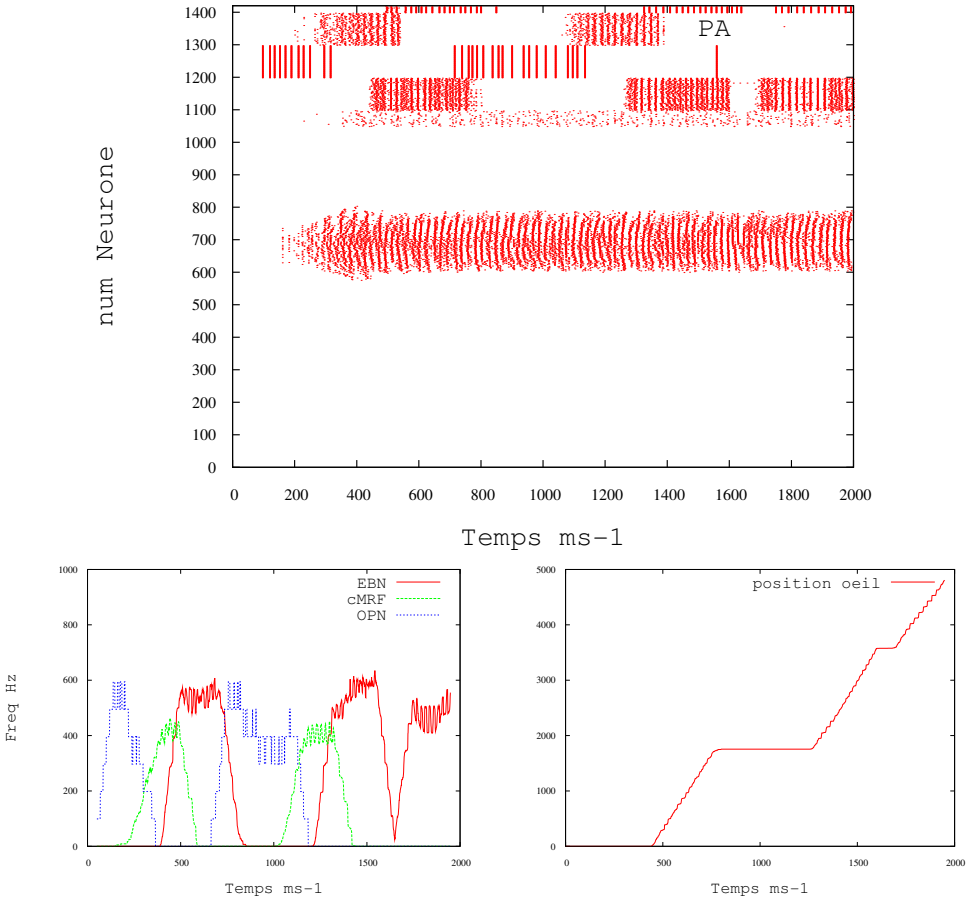

FIG. 39 – *Stimulation colliculaire en position 30. Fréquence de stimulation 750Hz*

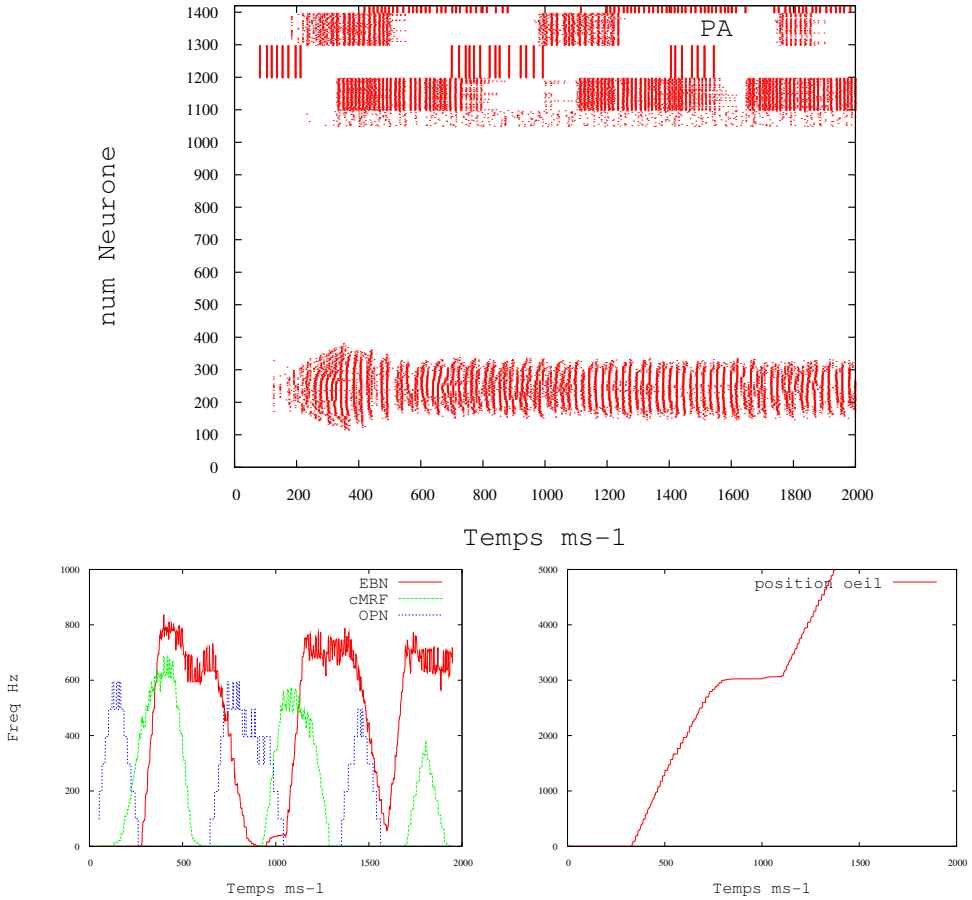

FIG. 40 – *Stimulation colliculaire en position 10. Fréquence de stimulation* 750*Hz*

Bibliographie

Abbott, L. F. (1999). Lapicque's introduction of the integrate-and-fire neuron model. *Brain Research Bulletin*, *50*(5-6), 303–304. (Cité page 10.)

Abbott, L. F., & Nelson, S. B. (2000). Synaptic plasticity : taming the beast. *Nature Neuroscience*, *3*, 1178–1183. (Cité page 26.)

Abeles, M. (1982). *Local Cortical Circuits : An Electrophysiological study*. Berlin : Springer. (Cité page 23.)

Abeles, M. (1991). *Corticotronics : Neural circuits of the Cerebelar Cotex*. Cambridge : Cambridge University Press. (Cité page 23.)

Alexandre, F. (2009). Cortical basis of communication : local computation, coordination, attention. *Neural Networks*, *21*. (to appear). (Cité page 30.)

Arai, K., & Keller, E. L. (2005). A model of the saccade-generating system that accounts for trajectory variations produced by competing visual stimuli. *Biological Cybernetics*, *92*, 21–37. (Cité page 119.)

Badler, B., Jeremy, & Keller, E. L. (2002). Decoding of a motor command vector from distributed activity in superior colliculus. *Biological Cybernetics*, *86*, 179–189. (Cité page 119.)

Bakkum, D. J., Chao, Z. C., & Potter, S. M. (2008). Long-term activity-dependent plasticity of action potential propagation delay and amplitude in cortical networks. *PLoSONE*, *3*(5), e2088. (Cité page 24.)

Bannister, P. A. (2005). Inter- and intra-laminar connections of pyramidal cells in the neocortex. *Neuroscience Research*, *53*, 95–103. (Cité page 30.)

Barton, E. J., Nelson, J. S., Gandhi, N. J., & Sparks, D. L. (2003). Effects of partial lidocaine inactivation of the paramedian pontine reticular formation on saccades of macaques. *Journal of Neurophysiology*, *90*, 372–386. (Cité page 115.)

Bi, G.-q., & Poo, M.-m. (1998). Synaptic modifications in cultured hippocampal neurons : Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, *18*(24), 10464–10472. (Cité pages 26 et 27.)

Bi, G.-q., & Poo, M.-m. (2001). Synaptic modification of correlated activity : Hebb's postulate revisited. *Annual Review of Neuroscience*, *24*, 139–166. (Cité page 26.)

Bi, G.-q., & Rubin, J. (2005). Timing in synaptic plasticity : from detection to integration. *Trends in Neuroscience*, *28*(5), 222–228. (Cité page 27.)

Bohte, S. M. (2004). The evidence for neural information processing with precise spike-times : A survey. *Natural Computing*, *3*(4), 195–206. (Cité page 5.)

Boniface, Y., Alexandre, F., & Vialle, S. (1999). A bridge between two paradigms for parallelism : neural networks andgeneral purpose mimd computers. In *IJCNN '99. International Joint Conference on Neural Networks*, (pp. 2441–2446). (Cité page 31.)

Bower, J. M., & Beeman, D. (1998). *The book of GENESIS : exploring realistic neural models with the GEneral NEural SImulation System*. vol. 2nd. New York : Springer. (Cité pages 32 et 40.)

Brette, R. (2006). Exact simulation of integrate-and-fire models with synaptic conductances. *Neural Computation*, *18*(8), 2004–2027. (Cité pages 20, 31 et 46.)

Brette, R. (2007). Exact simulation of integrate-and-fire models with exponential currents. *Neural Computation*, *19*(10), 2604–2609. (Cité pages 20, 31 et 46.)

Brette, R., & Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, *94*(7), 3637–3642. (Cité page 31.)

Brette, R., Rudolph, M., Carnevale, N. T., Hines, M. L., Beeman, D., Bower, J. M., Diesmann, M., Morrison, A., Goodman, P., Friederick, H. J., Zirpe, M., Natschläger, T., Pecevski, D., Ermentrout, G. B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davidson, A., El Boustani, S., & Destexhe, A. (2007). Simulation of networks of spiking neurons : A review of tools and strategies. *Journal of Computational Neuroscience*. (Cité pages 32, 33, 39 et 146.)

Brown, R. (1988). Calendar queues : a fast o(1) priority queue implementation for the simulation event set problem. *Commun. ACM*, *31*(10), 1220–1227. (Cité page 44.)

Brunel, N., Hakim, V., & Richardson, M. J. E. (2003). Firing-rate resonance in a generalized integrate-and-fire neuron with subthreshold resonance. *Phys. Rev. E*, *67*(5), 051916. (Cité page 13.)

Bryant, R. E. (1984). A switch-level model and simulator for mos digital systems. *IEEE Transactions on Computers*, *33*(2), 160–177. (Cité page 61.)

Bumble, M., & Coraor, L. (1998). Implementing parallelism in random discrete event-driven simulation. (Cité page 54.)

Büttner, U., & Büttner-Ennever, J. A. (2006). Present concepts of oculomotor organization. In J. B& (Ed.) *Neuroanatomy of the Oculomotor System*, vol. 151 of *Progress in Brain Research*, chap. 1, (pp. 1–42). Elsevier. (Cité pages 112 et 133.)

Buxhoeveden, D. P., & Casanova, M. F. (2002). The minicolumn hypothesis in neuroscience. *Brain*, *125*, 935–951. (Cité page 30.)

Carrillo, R. R., Ros, E., Brabour, B., Boucheny, C., & Cœnen, O. (2007). Event-driven simulation of neural population synchronization facilitated by electrical coupling. *Biosystems*, *87*(2-3), 275–280. (Cité pages 43 et 46.)

Cessac, B., Doyon, B., Quoy, M., & Samuelides, M. (1994). Mean-field equations, bifurcation map and route to chaos in discrete time neural networks. *Physica D*, *74*, 24–44. (Cité page 29.)

Chandy, K. M., & Misra, J. (1979). Distributed simulation : A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, *SE-5*(5), 440–452. (Cité pages 56 et 61.)

Chevallier, S., & Tarroux, P. (2008). Covert attention with a spiking neural network. In A. Gasteratos, M. Vincze, & J. K. Tsotsos (Eds.) *ICVS*, vol. 5008 of *Lecture Notes in Computer Science*, (pp. 56–65). Springer. (Cité page 7.)

Claverol, E. T., Brown, A. D., & Chad, J. E. (2002). Discrete simulation of large aggregates of neurons. *Neurocomputing*, *47*(1-4), 277–297. (Cité pages 44, 45, 46 et 51.)

Claverol, E. T., Cannon, R. C., Chad, J. E., & Brown, A. D. (1999). Event based neuron models for biological simulation. a model of the locomotion circuitry of the nematode c.elegans. *Computational Intelligence and Applications*, (pp. 217–222). (Cité page 45.)

Crick, F., & Koch, C. (2003). A framework for consciousness. *Nature Neuroscience*, *6*(2), 119–126. (Cité page 7.)

Cromer, J. A., & Waitzman, D. M. (2007). Comparison of saccade-associated neuronal activity in the primate central mesencephalic and paramedian pontine reticular formations. *Journal of Neurophysiology*, *98*, 835–850. (Cité page 115.)

Delorme, A., Gautrais, J., VanRullen, R., & Thorpe, S. J. (1999). Spikenet : A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, *26-27*, 989–996. (Cité pages 32 et 39.)

Destexhe, A. (1997). Conductance-based integrate-and-fire models. *Neural Computation*, *9*(3), 503–514. (Cité page 20.)

Destexhe, A., Mainen, Z. F., & Sejnowski, T. J. (1994). An efficient method for computing synaptic conductances based on a kinetic model of receptor binding. *Neural Computation*, *6*(1), 14–18. (Cité page 20.)

Diesmann, M., & Gewaltig, M.-O. (2002). Nest : An environment for neural systems simulations. In *Forschung und wisschenschaftliches Rechnen, Beitrage zum Heinz-Billing-Preis 2001*, vol. 58 of GWDG-Bericht, (pp. 43–70). Gottingen : Ges. fur Wiss. Datenverarbeitung. (Cité pages 32 et 46.)

Eckhorn, R., Reitbœck, H. J., Arndt, M., & Dicke, P. (1989). Feature linking via stimulus - evoked oscillations : Experimental results for cat vi- sual cortex and functional implications from a network model. In *IJCNN'89*, (pp. 723–730). (Cité page 7.)

Engel, A. K., Konig, P., Kreiter, A. K., & Singer, W. (1991). Interhemispheric synchronisation of oscillatory neuronal responses in cat visual cortex. *Science*, *252*, 1177–1179. (Cité page 7.)

Eppler, J. M., Morrison, A., Diesmann, M., Plesser, H. E., & Gewaltig, M.-O. (2006). Parallel and distributed simulation of large biological neural networks with nest. In *Computational Neurscience Meeting CNS'06*, (p. S48). Edingburgh, UK. (Cité page 54.)

Ermentrout, G. B. (2002). *Simulating, Analyzing, and Animating Dynamical Systems : A Guide to XPPAUT for Researchers and Students*. SIAM. (Cité page 32.)

Ermentrout, G. B. (2005). *Neural Oscillators*, vol. 1860/2005 of *Lecture Notes in Mathematics*, chap. 3, (pp. 69–106). Springer Berlin / Heidelberg. (Cité page 14.)

Ermentrout, G. B., & Kopell, N. (1986). Parabolic bursting in an excitable system coupled with a slow oscillation. *SIAM Journal on Applied Mathematics*, *46*(2), 233–253. (Cité page 13.)

Felderman, R. E., & Kleinrock, L. (1990). An upper bound on the improvement of asynchronous versus synchronous distributed processing. In D. Nicol (Ed.) *Proceedings of the SCS Multiconference on Distributed Simulations*, (pp. 131–136). (Cité page 61.)

Ferscha, A. (1996). Parallel and distributed simulation of discrete event systems. In *Parallel and Distributed Computing Handbook*, (pp. 1003 – 1041). McGraw-Hill. (Cité pages 44 et 57.)

Ferster, D., & Lindström, S. (1983). An intracellular analysis of geniculo-cortical connectivity in area 17 of the cat. *Journal of Physiology*, *342*, 181–215. (Cité page 23.)

Findlay, J., & Walker, R. (1999). A model of saccade generation based on parallel processing and competitive inhibition. *Behavioral and Brain Sciences*, *22*, 661–674. (Cité page 112.)

FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, *1*, 445–466. (Cité page 15.)

Flynn, M. J., & Rudd, K. W. (1996). Parallel architectures. *ACM Computation Surveys*, *28*(1), 67–70. (Cité page 54.)

Fourcaud-Trocmé, N., Hansel, D., Van Vreeswijk, C., & Brunel, N. (2003). How spike generation mechanisms determine the neuronal response to fluctuating inputs. *Journal of Neuroscience*, *23*(37), 11628–11640. (Cité page 14.)

Fries, P., Roelfsema, P. R., Engel, A. K., Konig, P., & Singer, W. (1997). Synchronization of oscillatory responses in visual cortex correlates with perception in interocular rivalry. *Proceedings of the National Academy of Sciences USA*, *94*, 12699–12704. (Cité page 7.)

Gancarz, G., & Grossberg, S. (1998). A neural model of the saccade generator in the reticular formation. *Neural Networks*, *11*, 1159–1174. (Cité page 119.)

Gandhi, N. J., & Keller, E. L. (1999). Activity of the brain stem omnipause neurons during saccades perturbed by stimulation of the primate superior colliculus. *Journal of Neurophysiology*, *82*, 3254–3267. (Cité page 137.)

Gerstner, W., Kempter, R., van Hemmen, J. L., & Wagner, H. (1999). Hebbian learning of pulse timing in the barn owl auditory system. (pp. 353–377). (Cité pages 7 et 23.)

Gerstner, W., & Kistler, W. M. (2002). *Spiking Neuron Models : An Introduction*. New York, NY, USA : Cambridge University Press. (Cité pages 15, 17 et 29.)

Gerstner, W., & Van Hemmen, J. L. (1992). Associative memory in a network of 'spiking' neurons. *Network*, *3*, 139–164. (Cité pages x et 15.)

Gerstner, W., Van Hemmen, J. L., & Cowan, J. D. (1996). What matters in neuronal locking. *Neural Computation*, *8*, 1653–1676. (Cité page 18.)

Girard, B., & Berthoz, A. (2005). From brainstem to cortex : Computational models of saccade generation circuitry. *Progress in Neurobiology*, *77*(4), 215–251. (Cité pages 112, 114 et 115.)

Goodman, D. F. M., & Brette, R. (2008). Brian : a simulator for spiking neural networks in python. *Frontiers in Neuroinformatics*. (Cité page 33.)

Goodman, P. (2005). NCS (neocortical simulator project) homepage. Tech. rep. (Cité page 33.)

Grassmann, C., & Anlauf, J. K. (1998). Distributed, event-driven simulation of spiking neural networks. In *NC'98, International ICSC/IFAC Symposium on Neural Computation*, (pp. 100–105). ICSC Academic Press. (Cité pages 43, 54 et 58.)

Gray, C. M., & Singer, W. (1989). Stimulus-specific neuronal oscillations in orientation columns of cat visual cortex. *Proceedings of the National Academy of Sciences USA*, *86*, 1698–1702. (Cité page 5.)

Grossberg, S. (1999). The link between brain, learning, attention and consciousness. *Consciousness and cognition*, *8*, 1–44. (Cité page 7.)

Grossberg, S. (2001). Linking the laminar circuits of visual cortex to visual perception : Development, grouping and attention. *Neuroscience and Biobehavioral Reviews*, *25*, 513–526. (Cité page 7.)

Grossberg, S., Govindarajan, K. K., Wyse, L. L., & Cohen, M. A. (2004). Artstream : a neural network model of auditory scene analysis and source segregation. *Neural Networks*, *17*, 511–536. (Cité page 31.)

Guillaume, A., & Pelisson, D. (2001). Gaze shifts evoked by electrical stimulation of the superior colliculus in the head-unrestrained cat. i. effect of the locus and of the parameters of stimulation. *European Journal of Neuroscience*, *14*, 1331–1344. (Cité pages 117, 118, 126, 127, 130, 131, 133, 134 et 135.)

Hammarlund, P., & Ekeberg, O. (1998). Large neural network simulations on multiple hardware platforms. *Journal of Computational Neuroscience*, *5*, 443–459. (Cité page 32.)

Hanes, D. P., & Wurtz, R. H. (2001). Interaction of the frontal eye field and superior colliculus for saccade generation. *Journal of Neurophysiology*, *85*, 804–815. (Cité page 117.)

Hansel, D., Mato, G., Meunier, C., & Neltner, L. (1998). On numerical simulations of integrate-and-fire neural networks. *Neural Computation*, *10*(2), 467–483. (Cité page 38.)

Hebb, D. O. (1949). *The organization of behavior*. Willey, New York. (Cité page 25.)

Hellmich, H. H., & Klar, H. (2004). SEE : a concept for an FPGA based emulation engine for spiking neurons with adaptive weights. In *5th WSEAS International Conference on Neural Networks and Applications (NNA '04)*, vol. 25-27, (pp. 930–935). Udine ; Italy. (Cité page 54.)

Hines, M. L., & Carnevale, N. T. (2001). Neuron : a tool for neuroscientists. *The Neuroscientist*, *7*, 123–135. (Cité pages 32 et 40.)

Hines, M. L., Morse, T., Migliore, M., Carnevale, N. T., & Shepherd, G. M. (2004). Modeldb : A database to support computational neuroscience. *Journal of Computational Neuroscience*, *17*(1), 7–11. (Cité page 35.)

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, *117*(4), 500–544. (Cité pages 7, 9 et 13.)

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, vol. 79, (pp. 2554–2558). Washington : The Academy. (Cité pages 25 et 28.)

Hopfinger, J., Buonocore, M., & Mangun, G. (2000). The neural mechanisms of top-down attentional control. *Nature neuroscience*, *13*(3), 284–291. (Cité page 29.)

Horton, J. C., & Adams, D. L. (2005). The cortical column : a structure without a function. *Philosophical Transactions of the Royal Society B : Biological Sciences*, *360*(1456), 837–862. (Cité page 29.)

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interactions and functional architecture in the cat's visual cortex. *Journal of Physiology*, *160*, 106–154. (Cité page 4.)

Hugues, E., & Martinez, D. (2005). Encoding in a network of sparsely connected spiking neurons : application to locust olfaction. *Neurocomputing*, *65-66*, 537–542. (Cité pages 7 et 14.)

Hüning, H., Glünder, H., & Palm, G. (1998). Synaptic delay learning in pulse-coupled neurons. *Neural Computation*, *10*(3), 555–565. (Cité page 24.)

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569–1572. (Cité pages 15 et 47.)

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons ? *IEEE Transactions on Neural Networks*, *15*(5), 1063–1070. (Cité page 15.)

Izhikevich, E. M. (2006). Polychronisation : Computation with spikes. *Neural Computation*, *18*, 245–282. (Cité pages 12 et 23.)

Izhikevich, E. M., & Edelman, G. M. (2008). Large-scale model of mammalian thalamocortical systems. *Proceedings of the National Academy of Sciences USA*, *105*(9), 3593–3598. (Cité page 15.)

Izhikevich, E. M., Gally, J. A., & Edelman, G. M. (2004). Spike-timing dynamics of neuronal groups. *Cerebral cortex*, *14*, 933–944. (Cité page 23.)

Jaeger, H. (2001). The "echo state" approach to analysins and training recurrent neural networks. Tech. Rep. TR-GMD-148, German National Research Center for Information Technology. (Cité page 29.)

Jahnke, A., Schœnauer, T., Roth, U., Mohraz, K., & Klar, H. (1997). Simulation of spiking neural networks on different hardware platforms. In W. Gerstner, A. Germond, M. Hasler, & J.-D. Nicoud (Eds.) *ICANN*, vol. 1327 of *Lecture Notes in Computer Science*, (pp. 1187–1192). Springer. (Cité page 57.)

Jurgens, R., Becker, W., & Kornhuber, H. (1981). Natural and drug-induced variation of velocity and duration of human saccadic eye movements : evidence for control of the neural pulse generator by local feedbac. *Biological Cybernetics*, *39*, 87–96. (Cité page 115.)

Kastner, S., & Ungerleider, G. (2000). Mechanisms of visual attention in the human cortex. *Annual Review of Neuroscience*, *23*, 315–341. (Cité page 29.)

Kato, R., Grantyn, A., Dalezios, Y., & Moschovakis, A. K. (2006). The local loop of the saccadic system closes downstream of the superior colliculus. *Neuroscience*, *143*(1), 319–337. (Cité page 115.)

Kempter, R., Gerstner, W., & Van Hemmen, J. L. (1999). Hebbian learning and spiking neurons. *Physical Review*, *59*(4), 4498–4514. (Cité page 27.)

Kirkland, K. L., & Gerstein, G. L. (1999). A feedback model of attention and context dependence in visual cortical networks. *Journal of Computational Neuroscience*, *7*, 255–267. (Cité page 31.)

Kistler, W. M., Gerstner, W., & Van Hemmen, J. L. (1997). Reduction of the Hodgkin-Huxley equations to a single-variable threshold model. *Neural Computation*, *9*(5), 1015–1045. (Cité pages 17 et 18.)

Knight, B. W. (1972). Dynamics of encoding in a population of neurons. *Journal of General Physiology*, *59*, 734–766. (Cité page 10.)

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*, 59–69. (Cité page x.)

Kopell, N., & Ermentrout, G. B. (1986). Subcellular oscillations and bursting. *Mathematical Biosciences An International Journal*, *78*(2), 265–291. (Cité page 13.)

Korniss, G., Nowotny, M., Guclu, H., Torockai, Z., & Rikvold, P. (2003). Suppressing roughness of virtual times in parallel discrete-event simulations. *Science*, *299*(5607), 677–679. (Cité pages 62 et 64.)

Lapicque, L. (1907). Recherches quantitatives sur l'excitation electrique des nerfs traitée comme une polarisation. *Journal de Physiologie et Pathologie General*, *9*, 620–635. (Cité page 10.)

Larkum, M., Zhu, J., & Sakmann, B. (1999). A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, *398*, 338–341. (Cité page 29.)

Larkum, M., Zhu, J., & Sakmann, B. (2001). Dendritic mechanisms underlying the coupling of the dendritic with the axonal action potential initiation zone of adult rat layer 5 pyramidal neurons. *Journal of physiology*, *533*(2), 447–466. (Cité page 29.)

Lee, C., Rohrer, W. H., & Sparks, D. L. (1988). Population coding of saccadic eye movements by neurons in the superior colliculus. *Nature*, *332*, 357–360. (Cité page 117.)

Lee, G., & Nabil H. Farhat, N. H. (2001). The double queue method : a numerical method for integrate-and-fire neuron networks. *Neural Networks*, *14*(6-7), 921–932. (Cité page 38.)

Lee, P., & Hall, W. C. (2006). An in vitro study of horizontal connections in the intermediate layer of the superior colliculus. *Journal of Neuroscience*, *26*(18), 4763–4768. (Cité page 136.)

Legenstein, R., Markram, H., & Maass, W. (2003). Input prediction and autonomous movement analysis in recurrent circuits of spiking neurons. *Reviews in the Neurosciences (Special Issue on Neuroinformatics of Neural and Artificial Computation)*, *14*, 5–19. (Cité page 31.)

Lobb, C. J., Chao, Z. C., Fujimoto, R. M., & Potter, S. M. (2005). Parallel event-driven neural network simulations using the hodgkin-huxley model. In *Proceedings of the Workshop on Principles of Advanced and Distributed Simulations*, (pp. 16–25). PADS'05. (Cité pages 47 et 54.)

Lytton, W. W., Destexhe, A., & Sejnowski, T. J. (1996). Control of slow oscillations in the thalamocortical neuron : A computer model. *Neuroscience*, *70*(3), 673–684. (Cité page 31.)

Maass, W. (1994). Lower bounds for the computational power of networks of spiking neurons. *Electronic Colloquium on Computational Complexity (ECCC)*, *1*(19). (Cité page 19.)

Maass, W. (1996). Networks of spiking neurons : The third generation of neural network models. In P. Bartlett, A. Burkitt, & R. Williamson (Eds.) *Australian Conference on Neural Networks*, (pp. 1–10). Australian National University. (Cité page 12.)

Maass, W., & Markram, H. (2004). On the computational power of circuits of spiking neurons. *Journal of computer and system sciences*, *69*(4), 593–616. (Cité page 19.)

Maass, W., & Natschläger, T. (1998). Emulation of hopfield networks with spiking neurons in temporal coding. In *CNS '97 : Proceedings of the sixth annual conference on Computational neuroscience : trends in research, 1998*, (pp. 221–226). New York, NY, USA : Plenum Press. (Cité page 7.)

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states : A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560. (Cité page 29.)

Maass, W., & Schmitt, M. (1999). On the complexity of learning for spiking neurons with temporal coding. *Information and Computation*, *153*, 26–46. (Cité page 24.)

Makino, T. (2003). A discrete-event neural network simulator for general neuron models. *Neural Computing and Applications*, *11*(3-4), 210–223. (Cité pages 47 et 48.)

Marian, I., & Reilly, R. G. (2001). SpikeNNS - a simulator for spike processing networks. In B. Katalinic (Ed.) *Proceedings of 12th DAAAM International Symposium on Intelligent Manufacturing & Automation : Focus on Precision Engineering*. Jena, Germany : DAAAM International. (Cité page 32.)

Marian, I., Reilly, R. G., & Mackey, D. (2002). Efficient event-driven simulation of spiking neural networks. In *Proceedings of the 3rdWSEAS International Conference on Neural Networks and Application*. (Cité page 46.)

Marin, M. (1998). *Discrete-Event Simulation On The Bulk-Synchronous Parallel Model*. Ph.D. thesis, University of Oxford, Faculty of Mathematical Sciences, Wolfson Building, Parks Road, Oxford OX1 3QD, UK. (Cité pages 58 et 61.)

Marin, M. (2000a). Comparative analysis of a parallel discrete-event simulator. In *SCCC*, (pp. 172–177). (Cité page 44.)

Marin, M. (2000b). An evaluation of conservative protocols for bulk-synchronous parallel discrete-event simulation. In *ESM*, (pp. 83–90). (Cité page 62.)

Markram, H. (2006). The blue brain project. *Nature Reviews Neuroscience*, *7*, 153–160. (Cité page 53.)

Markram, H., Lubke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsp. *Science*, *275*, 213–215. (Cité page 27.)

Mattia, M., & Del Giudice, P. (2000). Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Computation*, *12*, 2305–2329. (Cité pages 45 et 50.)

Mayrhofer, R., Affenzeller, M., Prähofer, H., Hfer, G., & Fried, A. (2002). Devs simulation of spiking neural networks. In *Proceedings of Cybernetics and Systems (EMCSR)*, vol. 2, (pp. 573–578). Austrian Society for Cybernetic Studies. (Cité page 19.)

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115–133. (Cité page 4.)

McIlwain, J. T. (1976). Large receptive fields and spatial transformations in the visual system. *International review of physiology*, *10*, 223–248. (Cité page 117.)

Migliore, M., Cannia, C., Lytton, W. W., Markram, H., & Hines, M. L. (2006). Parallel networks simulations with neuron. *Journal of Computational Neuroscience*, (21), 119–129. (Cité page 54.)

Misra, J. (1986). Distributed discrete event simulation. *ACM Computing Surveys*, *18*(1), 39–65. (Cité pages 58 et 61.)

Miura, K., & Optican, L. M. (2006). Membrane channel properties of premotor excitatory burst neurons may underlie saccade slowing after lesions of omnipause neurons. *Journal of Computational Neuroscience*, *20*, 25–41. (Cité page 137.)

Mohraz, K., Schott, U., & Pauly, M. (1997). Parallel simulation of pulse-coded neural networks. In *IMACS*. Berlin, Germany. (Cité page 57.)

Morris, C., & Lecar, H. (1981). Voltage oscillations in the barnacle giant muscle fiber. *Biophysical Journal*, *35*(1), 193–213. (Cité page 13.)

Morrison, A., Diesmann, M., & Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, *98*, 459–478. (Cité page 27.)

Morrison, A., Mehring, C., Geisel, T., Aertsen, A., & Diesmann, M. (2005). Advancing the boundaries of high-connectivity network simulation with distributed computing. *Neural Computation*, *17*, 1776–1801. (Cité pages 43 et 54.)

Moschovakis, A. K., Kitama, T., Dalezios, Y., Petit, J., Brandi, A. M., & Grantyn, A. A. (1998). An anatomical substrate for the spatiotemporal transformation. *Journal of Neuroscience*, *18*(23), 10219–10229. (Cité pages 114 et 115.)

Mountcastle, V. B. (1978). *An Organizing Principle for Cerebral Function : The Unit Model and the Distributed System*. Cambridge MA : MIT Press. (Cité page 29.)

Mountcastle, V. B. (1997). The columnar organization of the neocortex. *Brain*, *120*, 701–722. (Cité page 29.)

Mountcastle, V. B. (2003). Introduction. computation in cortical columns. *Cerebral cortex*, *13*(1), 2–4. (Cité page 29.)

Mouraud, A., & Paugam-Moisy, H. (2006). Learning and discrimination through stdp in a top-down modulated associative memory. In ESANN'06 (Ed.) *European Symposium on Artificial Neural Networks*, (pp. 611–616). (Cité page 29.)

Mouraud, A., Paugam-Moisy, H., & Puzenat, D. (2006). A distributed and multithreaded neural event driven simulation framework. In T. Fahringer (Ed.) *Parallel and Distributed Computing and Networks*, (pp. 212–217). IASTED/ACTA Press. (Cité page 92.)

Muresan, R. C., & Ignat, I. (2004). Principles of design for large-scale neural simulators. In *International Conference on Automation, Quality and Testing, Robotics*. (Cité pages 32 et 46.)

Mysore, S., & Quartz, S. (2005). Modeling structural plasticity in the barn owl auditory localization system with a spike-time dependant hebbian learning rule. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, (pp. 2766–2771). IEEE-INNS. (Cité page 23.)

Nagumo, J., Arimoto, S., & Yoshizawa, S. (1962). An active pulse transmission line simulating nerve axon. *Proc IRE*, *50*, 2061–2070. (Cité page 15.)

Natschläger, T., Markram, H., & Maass, W. (2003). *Computer models and analysis tools for neural microcircuits*, chap. 9, (pp. 123–138). Boston, MA : Kluwer Academic Publishers. (Cité page 32.)

Nethercote, N., & Seward, J. (2007). Valgrind : A framework for heavyweight dynamic binary instrumentation. In *Proceedings of ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation (PLDI 2007)*, (pp. 353–362). San Diego, California, USA. (Cité page 93.)

Nowotny, T., Zhigulin, V. P., & Selverston, A. I. (2003). Enhancement of synchronization in a hibrid neural circuit by spike-time-dependent plasticity. *Journal of Neuroscience*, *23*(30), 9776–9785. (Cité page 27.)

Ong, H., & Farell, P. A. (2000). Performance comparison of lam/mpi, mpich, and mvich on a linux cluster connected by a gigabit ethernet network. In *Proceedings of the 4th Annual Linux Showcase and Conference*, (pp. 353–362). Atlanta, Georgia. (Cité page 55.)

O'Regan, K., & Noe, A. (2001). A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences*, *24*, 939–1031. (Cité page 7.)

Paugam-Moisy, H. (2006). Spiking neuron networks a survey. Idiap-rr-11-2006, IDIAP. (Cité page 12.)

Paugam-Moisy, H., Martinez, R., & Bengio, S. (2008). Delay learning and polychronization for reservoir computing. *Neurocomputing*, *71*, 1143–1158. (Cité page 23.)

Plesser, H. E., Eppler, J. M., Morrison, A., Diesmann, M., & Gewaltig, M.-O. (2007). Efficient parallel simulation of large-scale neuronal networks on clusters of multiprocessor computers. In Springer (Ed.) *Euro-Par 2007 Parallel Processing*, vol. 4641/2007 of *Lecture Notes in Computer Science*, (pp. 672–681). Berlin / Heidelberg. (Cité page 54.)

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C : the art of scientific computing*, (2nd ed.). Cambridge ; New York ; New Rochelle ; Melbourne : Cambridge University Press. (Cité page 37.)

Qi, W., & Crook, s. (2004). Tools for neuroinformatic data exchange : An xml application for neuronal morphology data. *Neurocomputing*, *58-60C*, 1091–1095. (Cité page 35.)

Rao, R., & Sejnowski, T. J. (2001). Spike-timing-dependent plasticity as temporal difference learning. *Neural computation*, *13*, 2221–2237. (Cité page 27.)

Raynal, M., & Singhal, M. (1996). Logical time : Capturing causality in distributed systems. *IEEE Computer*, *29*(2), 49–56. (Cité page 60.)

Reutimann, J., Giugliano, M., & Fusi, S. (2003). Event-driven simulation of spiking neurons with stochastic dynamics. *Neural Computation*, *15*(4), 811–830. (Cité page 43.)

Reynolds, J., & Desimone, R. (1999). The role of neural mechanisms of attention in solving the binding problem. *Neuron*, *24*, 19–29. (Cité page 7.)

Robinson, D. A. (1972). Eye movements evoked by collicular stimulation in the alert monkey. *Vision Research*, *12*, 1795–1808. (Cité page 119.)

Robinson, D. A. (1975). *Basic Mechanisms of Ocular Motility and their Clinical Implications*. Oxford, UK : Pergamon. (Cité page 115.)

Rochel, O., & Martinez, D. (2003). An event-driven framework for the simulation of networks of spiking neurons. In *ESANN'03, European Symposium on Artificial Neural Network*, (pp. 295–300). (Cité pages 32 et 46.)

Ros, E., Carrillo, R. R., & Ortigosa, E. M. (2006). Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics. *Neural Computation*, *18*, 2959–2993. (Cité pages 43 et 46.)

Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. (Cité page 25.)

Rudolph, M., & Destexhe, A. (2006). Analytical integrate-and-fire neuron models with conductance-based dynamics for event-driven simulation strategies. *Neural Computation*, *18*(9), 2146–2210. (Cité pages 20, 46 et 48.)

Rumelhart, D., Hinton, G., & Williams, R. (1986). *Parallel Distributed Processing : Explorations in the microstructure of cognition*, vol. I, chap. Learning internal representations by error propagation, (pp. 318–362). MIT Press. (Cité page 25.)

Schaefer, M., Schœnauer, T., Wolff, C., Hartmann, G., Klar, H., & Ruckert, U. (2002). Simulation of spiking neural networks - architectures and implementations. *Neurocomputing*, *48*(1), 647–679. (Cité page 57.)

Seiffert, U. (2004). Artificial neural networks on massively parallel computer hardware. *Neurocosmputing*, *57*, 135–150. (Cité page 58.)

Senn, W., Markram, H., & Tsodyks, M. (2001). An algorithm for modifying neurotransmitter release probability based on pre- and post-synaptic spike timing. *Neural Computation*, *13*(1), 35–68. (Cité page 27.)

Senn, W., Schneider, M., & Ruf, B. (2002). Activity-dependent development of axonal and dendritic delays, or, why synaptic transmission should be unreliable. *Neural Computation*, *14*(3), 583–619. (Cité page 24.)

Shelley, M. J., & Tao, L. (2001). Efficient and accurate time-stepping schemes for integrate-and-fire neuronal network. *Journal of Computational Neuroscience*, *11*(2), 111–119. (Cité page 38.)

Siegel, M., Körding, K. P., & König, P. (2000). Integrating top-down and bottom-up sensory processing by somato-dendritic interactions. *Journal of Computational Neuroscience*, *8*, 161–173. (Cité pages 7 et 29.)

Soetedjo, R., Kaneko, C. R. S., & Fuchs, A. F. (2002). Evidence that the superior colliculus participates in the feedback control of saccadic eye movements. *Journal of Neurophysiology*, *87*, 679–695. (Cité pages 135 et 137.)

Song, S., & Abbott, L. F. (2001). Cortical development and remapping through spike timing-dependent plasticity. *Neuron*, *32*, 339–350. (Cité page 27.)

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, *3*, 919–926. (Cité page 27.)

Sparks, D. L., Holland, R., & Guthrie, B. L. (1976). Size and distribution of movement fields in the monkey superior colliculus. *Brain Research*, *113*, 21–34. (Cité page 117.)

Steinhaus, H. (1956). Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, *1*, 801–804. (Cité pages 25 et 99.)

Stuart, G. J., & Sakmann, B. (1994). Active propagation of somatic action potentials into neocortical pyramidal cell dendrites. *Nature*, *367*(1), 69–72. (Cité page 5.)

Su, W.-K., & Seitz, C. L. (1988). Variants of the chandy-misra-bryant distributed discrete-event simulation algorithm. Tech. Rep. cs-tr-88-22, California Institute of Technology. (Cité page 62.)

Swadlow, H. A. (1985). Physiological properties of individual cerebral axons studied in vivo for as long as one year. *Journal of Neurophysiology*, *54*(5), 1346–1362. (Cité page 23.)

Swadlow, H. A. (1988). Efferent neurons and suspected interneurons in binocular visual cortex of the awake rabbit : Receptive fileds and binocular properties. *Journal of Neurophysiology*, *59*(4), 1162–1187. (Cité page 23.)

Swadlow, H. A., Gusev, A. G., & Bezdudnaya, T. (2002). Activation of a cortical column by a thalamocortical impulse. *Journal of Neuroscience*, *22*(17), 7766–7773. (Cité page 29.)

Thorpe, S. J., & Gautrais, J. (1997). Rapid visual processing using spike asynchrony. In M. Mozer, M. L. Jordan, & T. Petsche (Eds.) *Advances in Neural Information Processing Systems*, vol. 9, (pp. 901–907). MIT Press. (Cité page 7.)

Thorpe, S. J., & Imbert, M. (1989). Biological constraints on connectionist modelling. In F. F.-S. . L. S. R. Pfeifer, Z. Schreter (Ed.) *Connectionism in perspective*, (pp. 63–93). Amsterdam ; Netherlands : Elsevier. (Cité page 5.)

Tonnelier, A., Belmabrouk, H., & Martinez, D. (2007). Event-driven simulations of nonlinear integrate-and-fire neurons. *Neural Computation*, *19*(12), 3226–3238. (Cité page 47.)

Turing, A. (1936). On computable numbers, with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Society*, vol. 42 of *2*, (pp. 230–âĂŞ265). (Cité page 4.)

Van der Vlist, E. (2002). *XML Schema : The W3C's Object-Oriented Descriptions for XML*. O'Reilly. (Cité page 35.)

Van Gisbergen, J. A. M., Robinson, D. A., & Gielen, S. (1981). A quantitative analysis of generation of saccadic eye movements by burst neurons. *Journal of Neurophysiology*, *45*(3). (Cité page 113.)

Vapnik, V., & Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Apl.*, *16*, 264–280. (Cité page 25.)

Verìssimo, P. (1996). Causal delivery protocols in real-time systems : a generic model. *Journal of Real-Time Systems*, *10*(1), 45–73. (Cité page 60.)

Von Der Malsburg, C. (1981). The correlation theory of brain function. Internal Report 81-2, Dept. of Neurobiology, Max-Planck-Institute for Biophysical Chemistry, Göttingen, Germany. (Cité page 7.)

Waitzman, D. M., Silakov, V. L., & Cohen, B. (1996). Central mesencephalic reticular formation (cMRF) neurons discharging before and during eye movements. *Journal of Neurophysiology*, *75*(4), 1546–1572. (Cité page 116.)

Wang, X., & Buzsaki, G. (1996). Gamma oscillation by synaptic inhibition in a hippocampal interneuronal network model. *Journal of Neuroscience*, *16*(20), 6402–6413. (Cité pages 10 et 14.)

Watts, L. (1994). Event-driven simulation of networks of spiking neurons. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.) *Advances in Neural Information Processing System*, vol. 6, (pp. 927–934). MIT Press. (Cité page 41.)

Widrow, B., & Hoff, M. E. J. (1960). Adaptive switching circuits. *IRE WESCON Convention Record*, *4*, 96–104. (Cité page 25.)

Zee, D. S., Optican, L. M., Cook, J. D., Robinson, D. A., & Engel, W. K. (1976). Slow saccades in spinocerebellar degeneration. *Archives of Neurology*, *33*(4), 243–251. (Cité page 115.)

Cette thèse, réalisée par Anthony Mouraud sous la co-direction d'Hélène Paugam-Moisy et Didier Puzenat, est financée par l'Université Antilles Guyane et a été réalisée dans les laboratoires : ISC ;Lyon, LIRIS ;Lyon et GRIMAAG ;Pointe-à-Pitre.

Titre Approche distribuée pour la simulation événementielle de réseaux de neurones impulsionnels.
Application au contrôle des saccades oculaires.

Résumé Pour les réseaux de neurones impulsionnels, effectuer les simulations en mode événementiel (*event-driven*) permet de réduire le temps d'exécution séquentiel par rapport aux méthodes *clock-driven*. D'autre part, l'utilisation d'un support parallèle permet de profiter d'un plus grand nombre de ressources matérielles pour optimiser les performances des simulations. Ce travail de thèse propose un simulateur événementiel, multithreadé et distribué pour la simulation de réseaux de neurones impulsionnels de grande taille. Il est dénommé par le sigle DAMNED, qui signifie *Distributed And Multithreaded Neural Event-Driven simulation framework*. Répartissant le réseau de neurones sur les ressources matérielles synchronisées par une méthode décentralisée de gestion du temps virtuel, DAMNED introduit également un fonctionnement multithread. DAMNED permet d'accélérer les calculs et de simuler des réseaux de plus grande taille qu'en séquentiel. DAMNED offre la possibilité d'exploiter de nombreux modèles de réseaux et de neurones impulsionnels et la plupart des supports matériels sont exploitables. Nous présentons l'utilisation de DAMNED sur un modèle simple de réseau pour différentes valeurs de tailles, connectivités et dynamiques. Ensuite, nous proposons une application directe de DAMNED dans une modélisation du contrôle des saccades oculaires. Ce modèle, entièrement basé sur des neurones impulsionnels, étudie les circuits neuronaux du système saccadique restreints au tronc cérébral. On montre, à l'aide de ce modèle, que l'hypothèse selon laquelle une somme vectorielle (« *vector summation* ») des activités de la carte motrice du colliculus supérieur coderait pour l'amplitude de la saccade correspond davantage aux données obtenues pour le modèle exécuté sur DAMNED qu'à l'hypothèse d'un moyennage de vecteurs (« *vector average* »). Même si des évolutions et optimisations sont envisagées, l'originalité de ce travail, parmi les premiers simulateurs distribués de réseaux de neurones impulsionnels, réside dans le couplage d'une stratégie événementielle, d'un multithreading interne aux processus logiques et une architecture physique distribuée. Le simulateur DAMNED constitue donc une avancée dans le domaine des réseaux de neurones impulsionnels, et particulièrement dans celui des réseaux de grande taille. Les expériences réalisées sur le modèle connexionniste du contrôle des saccades oculaires, et les résultats qu'elles apportent à la communauté neuroscientifique confirment les perspectives d'utilisation de ce travail de thèse.

Mots-clés Réseaux de neurones impulsionnels. Simulations événementielles. Simulation événementielles distribuées (DDES). Mutithreading. Neurosciences computationnelles. Système saccadique.

Title DAMNED, A Distributed And Multithreaded Event-Driven Spiking Neural Network Simulation Framework.
Application To The Control Of Saccadic Eye Movements.

Abstract Simulating spiking neuron networks with a sequential event-driven approach consumes less computation time than clock-driven methods. On the other hand, a parallel computing support provides a larger amount of material ressources for optimizing simulation performance. This PhD dissertation proposes an event-driven, multithreaded and distributed framework for simulating large size spiking neuron networks. The name of the simulator is the acronym DAMNED, for Distributed And Multithreaded Neural Event-Driven simulation framework. DAMNED distributes the neurons and connections of the network on the material ressources synchronized through a decentralized global virtual time. DAMNED also couples local multithreaded processing to the distributed hardware. DAMNED allows to speed up the simulation and to manage wider neural networks than sequential processing. DAMNED is suited to run many models of spiking neurons and networks, and most material supports are workable. Using DAMNED is presented first on toy networks and on a benchmark. Next, DAMNED is applied to modelling the control of saccadic eye movements. Completely based on spiking neurons, the model studies interactions between the neural circuits of the saccadic system located in the brainstem. The model helps validating the hypothesis that the saccade amplitude would be encoded by a vector summation of the activities in the superior colliculus motor map rather than a vector average, from comparison with data obtained in the simulation. Even if further developments and improvements may be forecasted, the originality of the work is to couple event-driven and distributed programming. Moreover, among the parallel simulators for spiking neuron networks, DAMNED is the first one taking advantage of an event-driven strategy internal multithreading of the logic processes and a distributed architecture of physical processes. Hence DAMNED is an advance in the area of simulating spiking neuron networks, mainly for wide size networks. Experiments on simulating the control of saccadic eye movements by a spiking neural network model and their contributions for the neuroscience community confirm the perspectives for further uses of the present work.

Keywords Spiking Neural Networks, Event-driven simulations, Distributed Discrete Event Simulations, Multithreading, Computational neuroscience, Saccadic system