Motion anticipation in a Virtual Retina

Selma Souihel

Ecole des Mines de Saint-Etienne

Institut National de Recherche en Informatique et en Automatique

Stage de fin d'étude
1^{er} Avril - 30 Septembre

Motion anticipation in a virtual Retina

Auteur :
Selma Souihel
Promotion 2016

Tuteur Industriel :
M. Bruno Cessac

Tuteur Académique :
M. Pierre Uny

28 Septembre 2016

Table des matières

Remerciements

Je souhaite tout d'abord remercier l'ensemble de l'équipe projet Biovision pour leur accueil chaleureux. En particulier, je remercie mon maître de stage Bruno Cessac de m'avoir offert cette opportunité et encadré au cours de ce stage, pour sa disponibilité, sa pédagogie et sa bienveillance. Merci également à Daniela Pamplona et Sélim Kraria pour leur aide technique et à Dora Karvouniari pour son partage d'expérience et ses précieux conseils. Enfin, je remercie Marie-Cécile Lafont pour s'être occupée de la partie administrative et d'avoir facilité mon arrivée et mon installation à Sophia Antipolis.

De chaleureux remerciements à mes enseignants et mes collègues du cycle ISMIN de l'école des Mines de Saint-Etienne, notamment Pierre Uny, mon tuteur académique, pour sa contribution à mon parcours scolaire et pour avoir toujours répondu positivement à mes requêtes. Je tiens également à remercier les professeurs et les étudiants de l'université nationale de Taiwan de m'avoir permis de vivre une expérience incroyable.

Plus généralement, merci à toutes les personnes qui ont rendu ce stage possible.

Introduction

Le système visuel fait partie intégrante du système nerveux central. Il permet de percevoir et d'interpréter une scène complexe et d'en extraire un très grand nombre d'informations.

La rétine, objet de notre étude, réalise l'acquisition et un premier traitement de l'image, dont le résultat sera transmis au cortex visuel via le nerf optique et le thalamus. La structure et l'agencement des neurones impliqués dans la perception visuelle permettent d'atteindre de grandes performances en terme de rapidité et de faible consommation énergétique.

Dans le cadre de ma dernière année à l'Ecole des Mines de Saint-Etienne, en cycle spécialisé en micro-électronique, informatique et nouvelles technologies, option biomédicale, j'ai été amenée à effectuer un stage de fin d'étude en lien avec ma spécialisation. Mon choix s'est porté sur l'étude de la rétine car la neurobiologie est un domaine qui m'a toujours fascinée. Le système visuel revêt en particulier une grande importance car ses mécanismes ne sont pas encore entièrement élucidés.

Le but du stage est d'utiliser et de mettre à jour le logiciel VirtualRetina `http://www-sop.inria.fr/neuromathcomp/public/software/virtualretina/`, un simulateur implémentant un modèle rétinien développé initialement par Adrien Wohrer et Pierre Kornprobst et intégré par la suite au logiciel Enas développé par l'équipe biovision, afin de reproduire l'activité de la rétine en réponse au stimulus d'une barre en mouvement, observé par M. Berry & al.[1] Une forme d'anticipation du mouvement a, en effet, été mise en évidence expérimentalement par ses auteurs dans les rétines de salamandre, de lapin et de poisson rouge. Cette anticipation peut être expliquée, dans le cas d'une trajectoire simple, par le méchanisme de contrôle de gain propre aux cellules ganglionnaires, méchanisme intégré dans VirtualRetina. Mais, pour des trajectoires plus complexes, ce mécanisme pourrait ne plus être suffisant, et la connectivité latérale de la rétine, i.e. le fait que les cellules rétiniennes soient connectées leur permettant de se transmettre des informations sur le mouvement de la barre, pourrait jouer un rôle majeur. Il s'agit donc, dans un premier temps, de vérifier si le simulateur VirtualRetina, implémentant le méchanisme de contrôle de gain, permet de reproduire les expériences de M. Berry pour une trajectoire simple. Il s'agit aussi de tester et de calibrer le logiciel pour reproduire au mieux les courbes expérimentales. On étudiera ensuite la réponse à des trajectoires plus complexes d'abord en l'absence de connectivité latérale. Enfin, si le temps le permet, nous introduirons la connectivité latérale afin d'en étudier l'effet sur l'anticipation du

mouvement. Nous utiliserons également Enas, un logiciel de simulation des réseaux neuronaux et d'analyse des trains de spikes, afin d'étudier le lien entre les correlations spatio-temporelles et la connectivité latérale.

Ce stage s'inscrit dans le cadre du projet "Trajectory" financé par l'Agence Nationale de la Recherche (ANR). Le but du projet est de modéliser les mécanismes d'anticipation dans la rétine et dans le cortex V1 qui jouent des rôles clés dans le traitement visuel préliminaire du mouvement. Ce travail de modélisation sera fait en partenariat avec l'Institut de la Vision (IDV) `http://www.institut-vision.org/fr/` et l'Institut des Neurosciences de la Timone (INT) `http://www.int.univ-amu.fr/`, qui nous fourniront les enregistrements micro et mésoscopiques. J'ai également eu la chance de me voir proposer une thèse doctorale dans le cadre de ce projet par mon tuteur, proposition que j'ai acceptée.

1 Présentation et mise en situation

1.1 Présentation du contexte général

1.1.1 INRIA, INRIA Sophia Antipolis

L'Institut National de Recherche en Informatique et en Automatique (INRIA) est un institut de recherche français en mathématiques et informatique. Créé en 1967 à Rocquencourt, INRIA s'est depuis étendu et dispose à présent de 8 sites en France : Rocquencourt, Rennes, Sophia Antipolis, Nancy, Grenoble, Bordeaux, Lille et Saclay. L'INRIA emploie des collaborateurs venus du monde entier, organisés en « équipes-projets », des équipes autonomes qui rassemblent des compétences complémentaires autour d'un projet scientifique focalisé. Ces projets peuvent appartenir à différents domaines tels que la santé, l'énergie, la communication, la sécurité ... L'INRIA développe de nombreux partenariats avec le monde industriel et favorise le transfert scientifique et la création d'entreprises

Le centre de Sophia Antipolis est présent sur la technopole depuis 1981. Il emploie plus de 700 personnes dont environ 500 scientifiques et 200 personnes dédiées au support et soutien à la recherche. 50 nationalités y sont représentées. Les équipes de recherche sont réparties entre le site de Sophia Antipolis et celui de Montpellier. Le centre possède également de nombreux partenaires, principalement en formation doctorale et enseignement, à l'exemple de l'Université Nice Sophia Antipolis. Enfin, le centre est fortement impliqué dans la création d'un nouveau campus universitaire à Sophia Antipolis, et l'élaboration de son contenu scientifique.

1.1.2 L'équipe Biovision

L'équipe Biovision a été créée en Janvier 2016. Elle émane de l'équipe-projet NEUROMATHCOMP dont les travaux portaient sur l'exploration du fonctionnement du cerveau grâce aux mathématiques et l'informatique. Tout en restant dans le domaine des neurosciences, Biovision a choisi la vision comme axe de recherche. Le cheminement scientifique pensé par Biovision aura donc pour but de mieux comprendre les mécanismes impliqués dans la vision humaine grâce à des études expérimentales. En cela elle compte de nombreux biologistes parmi ses partenaires.Une meilleure compréhension de ces méchanismes permettra de modéliser la vision biologique en tenant compte de ses particularités, dans l'optique de développer des technologies d'aide pour les patients malvoyants ou aveugles.

L'équipe se compose de deux chercheurs permanents Bruno Cessac et Pierre Kornprobst, une assistante d'équipe Marie-Cécile Lafont, deux doctorants Dora Karvouniari et Kartheek Medathati et deux chercheurs postdoctoraux Audric Drogoul et Daniela Pamplona.

1.2 Présentation du contexte scientifique

1.2.1 Neurosciences computaionnelles

La neuroinformatique ou les neurosciences computationnelles sont un domaine de recherche dont le but est de découvrir les principes régissant les fonctions cérébrales et l'activité neuronale en utilisant des modèles implémentés par des algorithmes. Le cerveau humain représente la structure la plus complexe connue dans la nature : cent milliards de neurones connectés par un réseau d'une grande complexité capables de traiter des informations complexes en un temps record. Le but donc des neurosciences computationnelles est de comprendre les mécanismes cérébraux par des moyens théoriques et informatiques en combinant l'expérimentation et la simulation.

1.2.2 Biologie de la rétine

La rétine est une mince membrane couvrant environ 75% de la surface interne du globe occulaire. Des études en neurophysiologie ont montré que les traitements de haut niveau réalisés par le système visuel humain ne sont possibles que grâce à un nombre important de pré-traitements effectués dans la rétine et permettant d'extraire des paramètres multidimensionnels de l'environnement.

La rétine (Figure 1) comporte plusieurs couches de cellules. La plus externe est celle des photorécepteurs, les bâtonnets et les cônes, qui constituent la partie photosensible de la rétine. Ces cellules assurent la transformation de l'énergie lumineuses en signal biochimique. Les cellules bipolaires, les cellules horizontales et amacrines forment quant à elles la couche des grains qui a pour rôle la transmission de l'influx nerveux aux cellules ganglionnaires, dont les axones forment les fibres du nerf optique. L'organisation complexe de la rétine et la nature des connexions entre les différentes catégories de cellules et des neurotransmetteurs libérés au niveau des synapses ne sont pas encore entièrement compris et font l'objet d'études très poussées.

La rétine constitue donc une première étape dans le processus de perception visuelle réduisant considérablement la quantité de données à transmettre

au cortex grâce à des prétraitements locaux et globaux. Notre étude servira à établir le degré d'implication de ces prétraitements rétiniens dans le phénomène de l'anticipation du mouvement, à la lueur de l'article de M. Berry & al.[1].

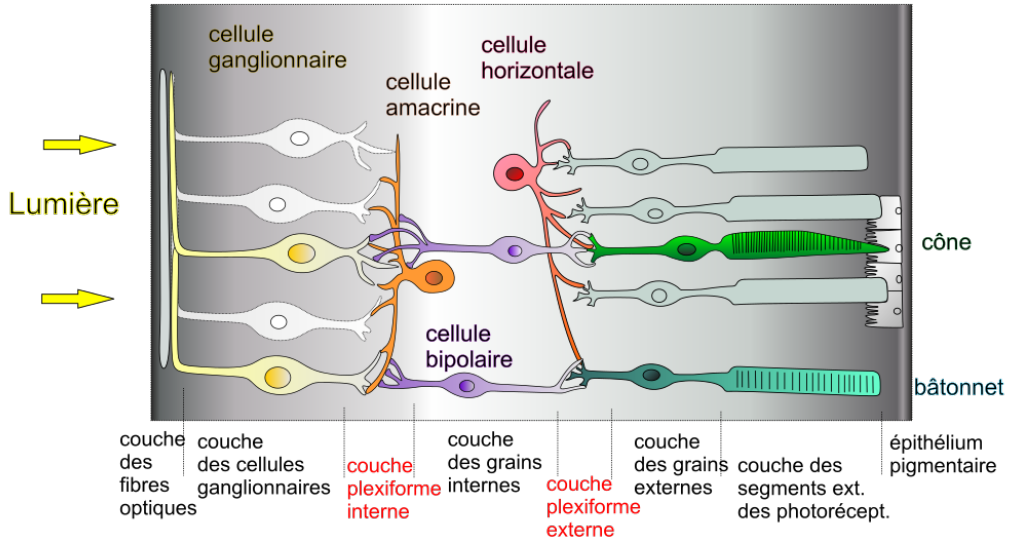

FIGURE 1 – Structure de la rétine. La couche des photorécepteurs reçoit les photons et les transforme en signal nerveux. Les cellules horizontales assurent la connectivité des photorécepteurs. Les bipolaires les relient aux cellules ganglionnaires, qui se prolongent pour former les fibres du nerf optique. (source wikipedia)

1.2.3 Article de M. Berry & al.

La conversion des stimuli en signal électrique ainsi que leurs processus d'élaboration, au niveau de la rétine, imposent un délai dans la transmission des informations visuelles. A titre d'exemple, un flash lumineux induit une activité neuronale dans le cerveau avec un retard de 30 à 100 ms.

Berry et al. (1999)[1] exposent dans leur article les résultats d'une recherche sur le temps de réponse de la rétine chez la salamandre et le lapin, dans l'optique de comprendre s'il existe des mécanismes au niveau local capables de compenser les délais de réponse du système visuel, lorsqu'il faut réagir à un danger par exemple. L'hypothèse avancée est que le système visuel

serait capable d'extrapoler la position courante d'un objet en mouvement à l'aide de sa position passée et de sa vitesse.

L'expérience est donc la suivante : une rétine isolée est connectée à une grille multi-électrodes d'enregistement. Elle a été stimulée d'abord par une barre statique (flash), ensuite par une barre en translation rectiligne uniforme avec une vitesse comprise entre 0,11 mm/s et 1,76 mm/s. L'acquisition des signaux par les électrodes a permis de mesurer la réponse spatio-temporelle des cellules ganglionnaires. La présence d'une activation anticipée de certaines cellules ganglionnaires a été mise en évidence dans le cas du stimulus de la barre en mouvement. Cette activation se manifeste dans le sens du mouvement et dépend de la vitesse de la barre. Cette dépendance se traduit par une pré-activation des cellules ganglionaires plus rapide lorsque la vitesse de la barre augmente, chez la salamandre et le lapin, pour une vitesse allant de 0,11 mm/s à 1,76 mm/s. (Figure 2)

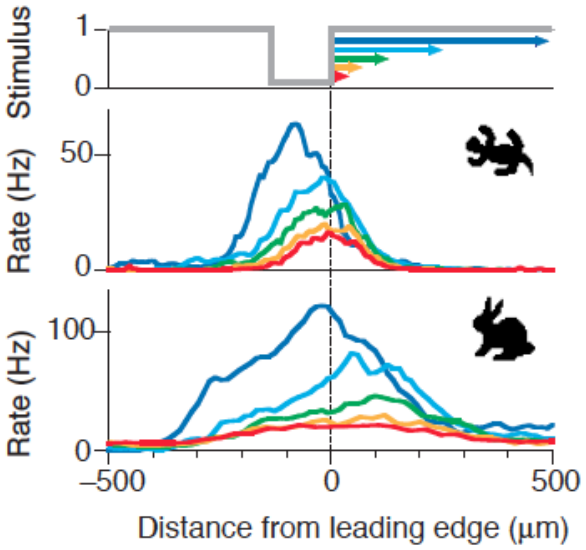

FIGURE 2 – Relation entre la pré-activation des cellules ganglionnaires et la vitesse de la barre [1] Stimulation avec des barres sombres (90% de contraste, 133 mm de largeur) en faisant varier la vitesse : 0.11, 0.22, 0.44, 0.88 et 1.76 $mm.s^{-1}$. Les fréquences de décharges correspondent à des cellules ganglionnaires de salamandre de type fast OFF et des cellules ganglionnaires de lapin de type OFF. 3

Spatialement, le pic d'activité d'une population de cellules ganglionnaires se situe à droite du centre de la barre, si le mouvement se fait de gauche à droite, et réciproquement, tandis que ce même pic se superpose au centre de

la barre lorsque celle-ci est flashée. (Figure 3)

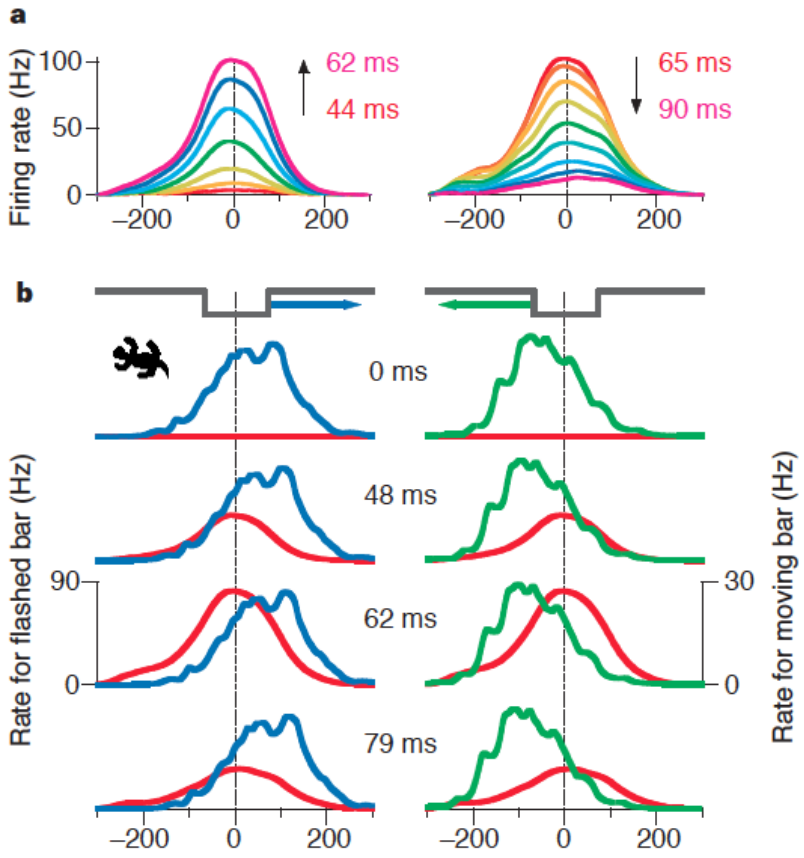

FIGURE 3 – Réponse des populations de cellules ganglionnaires dans le cas de la barre flashée (a) et de celle en mouvement (b) [1] Les graphiques a) représentent le profil spatial de l'activité d'une population de cellules ganglionnaires de salamandre de type fast OFF en réponse à une barre flashée (90% de contraste, 133 mm de largeur) à des différents temps après le flash. La réponse est maximale entre 62 et 65 ms après le flash. Les graphiques b) représentent l'activité spatiale de la population dans le cas de la barre en mouvement (courbes bleues et vertes) et de la barre flashée (courbes rouges).

2 Outils et notions fondamentales

2.1 Neurosciences théoriques

2.1.1 Potentiels d'action et fréquence de décharge

L'activité neuronale est mesurée par le nombre de potentiels d'action, appelés spikes, générés en réponse à un stimulus. Le potentiel d'action est un évènement court durant lequel le potentiel électrique d'un neurone augmente puis chute rapidement. Il dure entre 1 et 2 millisecondes.
Au repos, le potentiel de la membrane est d'environ -70mV. Survient alors une dépolarisation transitoire d'une amplitude de 100mV, le potentiel de la membrane passant de -70 à 30mV. Cette dépolarisation est suivie par une repolarisation puis une hyperpolarisation au cours de laquelle le potentiel diminue plus qu'à l'état basal (-80mV). Le potentiel retourne à son état de repos pendant une durée qui, jointe à la durée de l'hyperpolarisation, représente la période réfractaire pendant laquelle le neurone ne peut pas induire d'autre potentiels d'actions. (Figure 4)

Il est à noter que la production de potentiels d'action est un mécanisme électrique actif, qui doit rester compatible avec les contraintes d'un système vivant : en particulier avec la limitation de l'élévation de température et de la consommation d'énergie. En contraste avec nos ordinateurs, ce mécanisme est ionique et le transfert d'ions se fait grâce aux gradients de densité. La nature a ainsi trouvé une façon extrêmement économique de produite les potentiels d'action. Le corollaire est d'avoir des temps caractéristiques très lents ($\tilde{1}$ ms) en comparaison des temps "électroniques atteints dans nos ordinateurs ($<$ 1ns).

Le niveau d'excitation d'un neurone donné est essentiellement codé par sa fréquence de décharge, à savoir le nombre de potentiels d'action générés par unité de temps. Généralement, il existe une gamme de fréquences de décharge possibles pour une cellule donnée. Un neurone du système moteur, par exemple, aura une fréquence de décharge d'environ 8Hz une fois que son seuil d'excitation est atteint. La fréquence augmente à mesure que l'excitation augmente jusqu'à ce que le neurone sature à une fréquence de décharge maximale due à la période réfractaire.

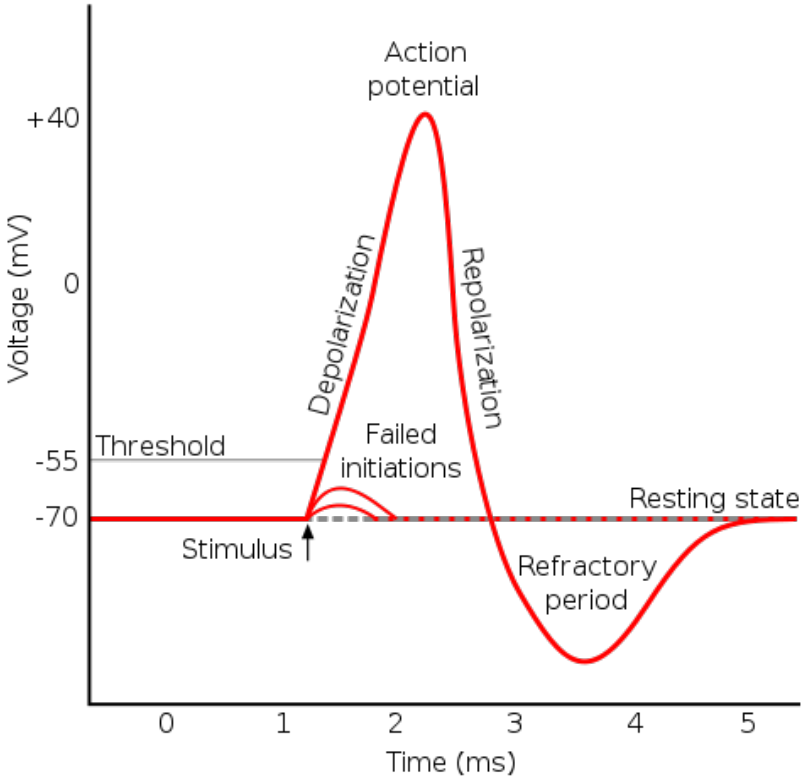

FIGURE 4 – Les différentes phases du potentiel d'action (Source : wikipedia)

Les potentiels d'action sont le plus souvent initiées par des potentiels postsynaptiques excitateurs d'un neurone présynaptique. Les neurotransmetteurs sont libérés par le neurone présynaptique puis se lient à des récepteurs sur la cellule postsynaptique. Cette liaison ouvre différents types de canaux ioniques. Cette ouverture a pour effet la modification de la perméabilité locale de la membrane cellulaire et, par conséquent, le potentiel de membrane. (Figure 5)
La neurotransmission peut également se produire par des synapses électriques. Ce type de synapses étant minoritaire, la neurotransmission électrique est plus rare que celle chimique.

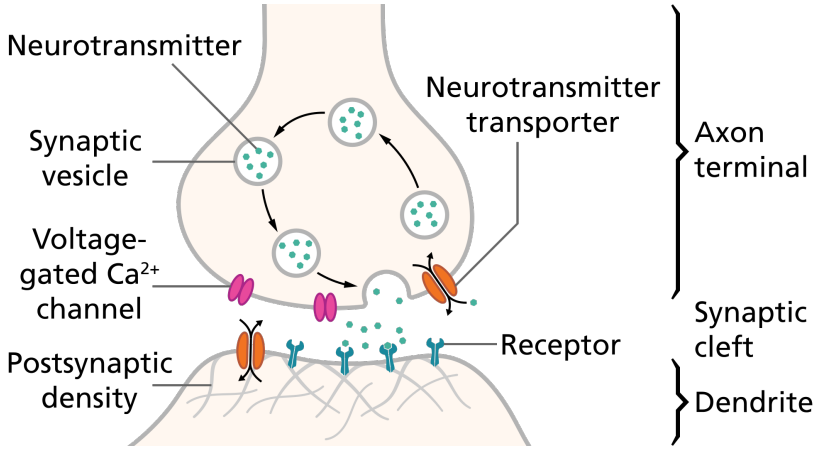

FIGURE 5 – Neurotransmission synaptique : le neurone présynaptique (haut) libère des neurotransmetteurs qui activent les récepteurs de la cellule post-synaptique (bas). (Source : wikipedia)

2.1.2 Modèles impulsionnels des réseaux de neurones

Il existe plusieurs théories formalisant l'activité des neurones. Les modélisations biophysiques traduisent en équations les mécanismes physiques et chimiques se produisant au niveau du neurone, la plus célèbre étant l'équation de Hodgkin et Huxley, à la base de la plupart des modélisations actuelles[4]. L'activité électrique des neurones et la production des spikes se fait grâce à des canaux ioniques. Dans la description faite par Alan Hodgkin et Andrew Huxley en 1952, les canaux ioniques sont constitués de portes indépendantes de trois types, qui s'ouvrent et se ferment en fonction de la valeur du potentiel de membrane. La probabilité d'ouverture de ces portes est décrite par des variables m, n et h. L'ouverture et la fermeture des portes m et h contrôlent le passage des ions sodium tandis que celles des portes n contrôlent le passage des ions potassium. Le modèle élaboré est un système d'équations qui décrit le potentiel de membrane en fonction de ces probabilités :

$$\frac{dV}{dt} = \left[I_{inj} - \bar{g}_{Na} m^3 h (V - V_{Na}) - \bar{g}_K n^4 (V - V_K) - g_L (V - V_L)\right] / C \quad (1)$$

$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V) n \quad (2)$$

$$\frac{dm}{dt} = \alpha_m(V)(1-m) - \beta_m(V) m \quad (3)$$

$$\frac{dh}{dt} = \alpha_h(V)(1-h) - \beta_h(V) h \quad (4)$$

où α et β sont des constantes qui dépendent uniquement du potentiel, et g_i les valeurs maximales de conductance.

D'autres modèles sont antérieurs à celui de Hodgkin et Huxley à l'exemple de Integrate and fire, un modèle simplifié introduisant la capacité de la membrane formulé par Louis Lapicque, le Leaky Integrate and Fire qui introduit un terme correctif à l'équation de Lapicque, ou encore le modèle exponentiel [4].

Les équations de Hodgkin Huxley peuvent être rassemblées en une équation différentielle avec quatre variable d'état V(t), m(t), n(t) et h(t). L'étude de ce système non linéaire est difficile et sa résolution ne peut être faite analytiquement. Certaines méthodes numériques permettent d'étudier le comportement général du système, mais dans les simulations où la production des trains de spikes ne représente qu'une partie d'une modélisation encore plus complexe, le modèle Leaky Integrate and Fire est préféré.

Le circuit de base d'un modèle Integrate and fire se compose d'un condensateur C en parallèle avec une résistance R entraîné par un courant I (t). (Figure 4). Le courant d'entraînement peut être divisé en deux composantes, I (t) = $I_R + I_C$. La première composante est le courant IR résistif, et la seconde IC est le courant qui charge le condensateur C.

$$\mathrm{I(t)} = \frac{u(t)}{R} + \mathrm{C}\,\frac{\mathrm{d}u}{\mathrm{d}t}$$

En 2005, G. Belson, O. Mazet et H. Soula ont proposé une version discrétisée du LIF. B. Cessac en a fait l'analyse mathématique complète.Dans toute la suite du rapport, nous parlerons du modèle BMS. L'équation du modèle BMS est la suivante :

$$V_i(t+dt) = V_i(t)(1-\tfrac{dt}{\tau}) + \tfrac{I_i(t)}{C}dt$$

$\tau = RC$ avec R la résistance de la membrane et C sa capacité.

En posant $dt = 1$ et $\gamma = 1 - \frac{dt}{\tau}$ On obtient :

$$V_i(t+dt) = \gamma V_i(t) + \tfrac{I_i(t)}{C}$$

avec $0 \leq \gamma < 1$. En effet, il faut que l'échelle de temps dt soit petite devant le temps caractéristique τ. Notons que poser dt=1 revient à choisir une nouvelle unité de temps, typiquement la ms.

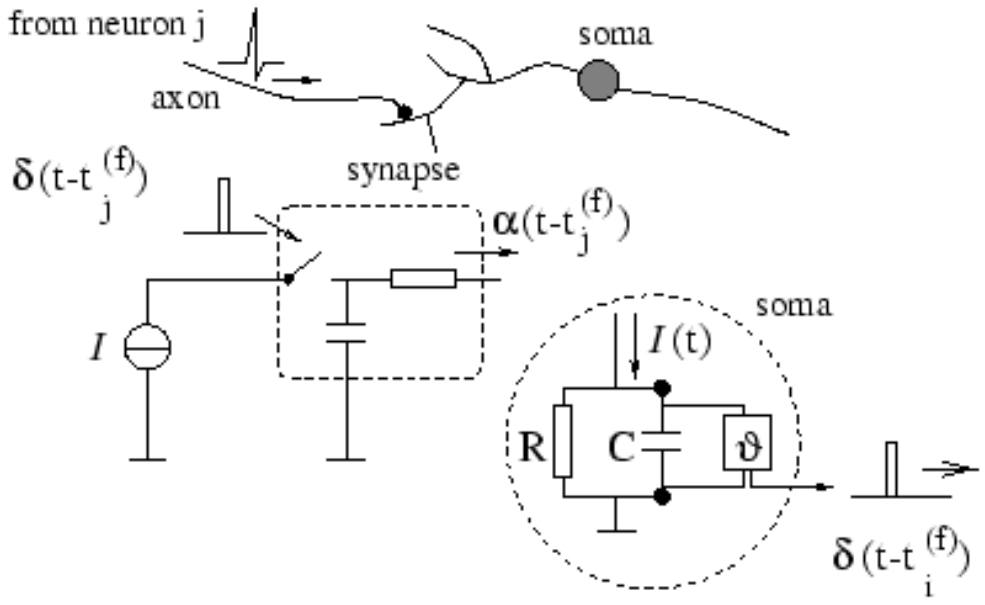

FIGURE 6 – Schéma du modèle Integrate and Fire. Le circuit de base est le module à l'intérieur du cercle en pointillés. La tension $U(t)$ aux bornes de la capacité est comparée à un seuil ϑ. Si $u(t) \geq \vartheta$ à l'instant $t_i(f)$ une impulsion de sortie $\delta(t - t_i(f))$ est générée. Le spike présynaptique $\delta(t - t_j(f))$ est filtré par un filtre passe-bas au niveau de la synapse et génère un potentiel post synaptique $\alpha(t - tj(f))$.[8]

Cette discrétisation donne une bonne approximation de l'activité des neurones. L'intêret étant de pouvoir caractériser la dynamique collective des neurones y compris en présence de bruit.

2.1.3 Peri stimulus time histogramm

En neurophysiologie, le peristimulus time histogramm PSTH est un histogramme des temps où les neurones envoient des impulsions. Ces histogrammes sont utilisés pour visualiser la fréquence des décharges neuronales par rapport à un stimulus externe ou un événement. Afin de construire un PSTH, un train de spikes enregistré à partir d'un seul neurone est aligné avec le début de la simulation, ou un spike de phase fixe, d'un stimulus identique présenté à plusieurs reprises. Les séquences alignées sont superposées dans le temps, et ensuite utilisé pour construire un histogramme.

La méthode de construction est la suivante :
1-Aligner les séquences de spikes avec le début d'un stimulus qui se répète n fois.
2-Diviser la période du stimulus ou la période d'observation T en N fenêtres de taille Δ.
3-Compter le nombre de spikes K_i de toutes les n séquences coïncidant avec la fenêtre i.
4-Dessinez un histogramme. La hauteur de barre qui correspond à la fenêtre i est donnée par $\frac{K_i}{n\Delta}$.

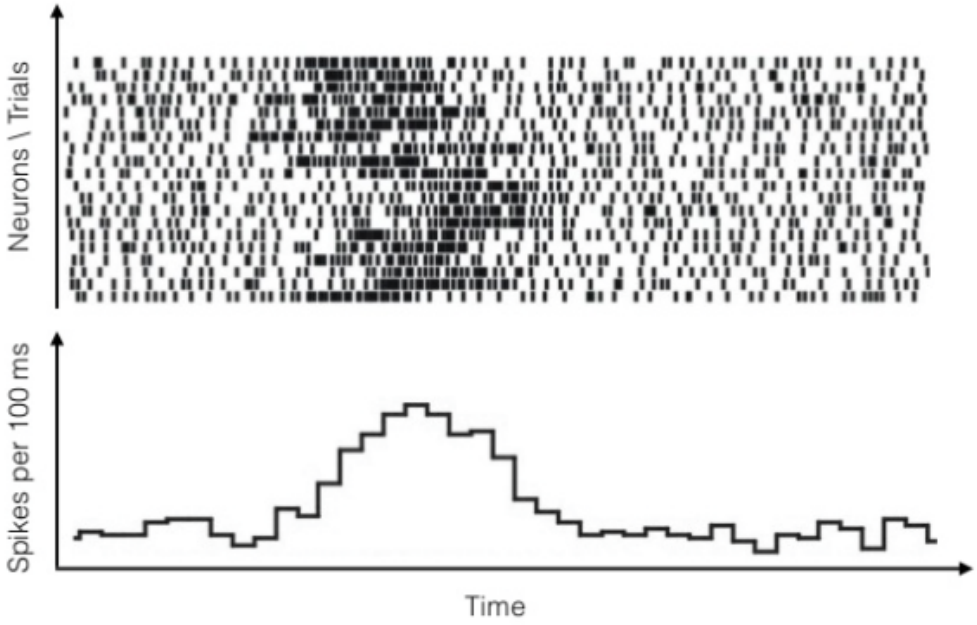

FIGURE 7 – Exemple de construction d'un PSTH à partir d'un train de spikes (Source : Machine learning on neuroimaging data by Ilya Kuzovkin)

2.1.4 Modélisation des champs récepteurs

De par sa structure, la rétine est capable d'extraire des informations locales d'une scène visuelle et de les convertir en trains de spikes. Plus précisément, on constate que chaque cellule ganglionnaire répond à un stimulus localisé en espace et en temps. Certaines cellules spécialisées vont par exemple répondre préférentiellement à une orientation déterminée ou une direction. En général, la réponse d'une cellule ganglionnaire à un stimulus se décrit par un noyau spatio-temporel K, convolué au stimulus et suivi par une non linéarité. K est appelé champ récepteur. Dans le cas le plus simple, les champs récepteurs sont des structures elliptiques de type centre-périphérie délimitant deux parties antagonistes, souvent identifiés comme étant la région de la rétine dans laquelle l'action de la lumière modifie l'activité d'un neurone. Les champs récepteurs peuvent être déterminé pour tous les niveaux du système visuel : les cellules ganglionnaires, les cellules latérales géniculées et les cellules du cortex visuel.

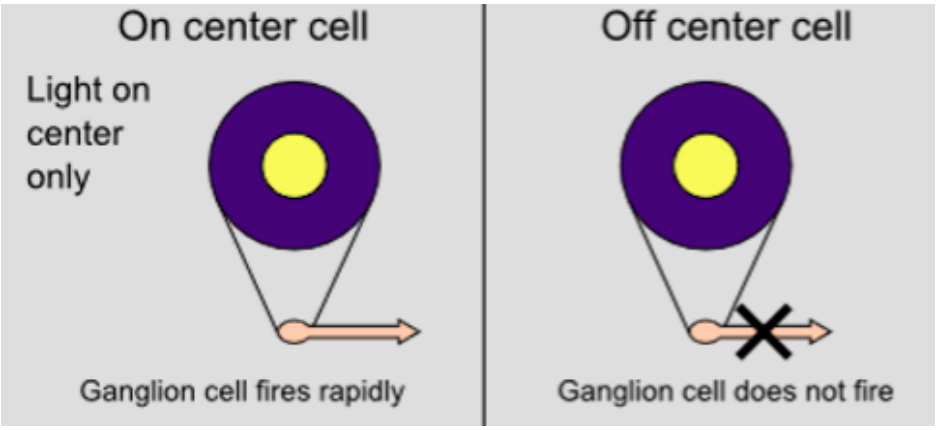

FIGURE 8 – Exemple de champs récepteurs des cellules ganglionaires de type On. Inversement, la cellule OFF produit un potentiel d'action lorsqu'elle stimulée en son contour, et inhibée lorsqu'elle est stimulée en son centre. (Source : wikipedia)

Historiquement, les champs récepteurs comportaient uniquement deux dimensions spatiales. Des recherches récentes ont cependant élargi cette notion en y intégrant une dimension temporelle. Le champ récepteur spatio-temporel décrit la relation entre la région spatiale du champ visuel où les réponses neuronales sont évoquées et le cours temporel de cette réponse. Un modèle simple de filtrage linéaire, basé sur une différence de gaussiennes, peut rendre compte de l'opposition centre-contour dans les champs récepteur. Ce modèle peut être utilisé pour les deux projections spatiale et temporelle. Certaines hypothèses simplificatrices de symétrie notamment sont parfois utilisées pour des représentations plus cohérentes.

2.1.5 Mécanisme de contrôle de gain

En automatique, le mécanisme de contrôle de gain fait référence à une boucle de rétroaction fermée dont le but est de réguler l'amplitude d'un signal en sortie d'un système, qui reçoit en entrée un signal plus ou moins variable. La valeur moyenne de la sortie est utilisée pour ajuster dynamiquement le gain du système, la réponse est atténuée lorsque le signal est fort et amplifiée lorsque celui-ci est faible. En neurosciences, les réponses d'un système neuronal suivent le même principe. En effet, elle est ajustée dynamiquement afin de couvrir la gamme des stimuli entrants. En d'autres termes, le neurone peur devenir plus ou moins sensible suivant la variabilité du stimulus. Cela a pour principale conséquence d'atténuer l'effet du bruit et d'améliorer l'extraction de l'information. Dans le système visuel, le mécanisme de contrôle de gain joue un rôle dans la détection de contours.

2.2 Outils statistiques

2.2.1 Modèles de Poisson de génération de spikes

Dans le cortex, les potentiels d'action successifs sont très irréguliers en temps. L'interprétation de cette irrégularité a conduit à deux points de vue divergents de l'organisation corticale. D'une part, l'irrégularité pourrait provenir de forces stochastiques. L'intervalle irrégulier entre les spikes reflète un processus aléatoire et implique qu'une estimation instantanée de la fréquence de décharge peut être obtenue en faisant la moyenne des réponses de nombreux neurones individuels mis en communs. En accord avec cette théorie, le temps précis des spikes individuels transporte peu d'information. Cependant, il apparaît que les coincidences précises de spikes jouent un rôle essentiel dans le codage neuronal. Dans ce cas, l'activité des neurones est

corrélée à travers leur connectivité et la présence d'un stimulus. Il en résulte une irrégularité due à l'interaction d'un grand nombre de neurones (ce qui n'exclut pas la présence de bruit). Il est possible que le système visuel utilise les deux modalités (neurones indépendants VS neurones corrélés). La modélisation la plus simple de l'activité neuronale dans le cas indépendant consiste à utiliser un modèle de Poisson qu'il est possible de généraliser pour y ajouter les caractéristiques plus réalistes comme la présence d'une période réfractaire ou le fait que l'activité d'un neurone dépend de son histoire et de l'histoire du réseau (on ne parle plus alors de processus de Poisson mais de processus de Hawkes).

2.2.2 Théorème de la limite centrale

Le théorème de la limite centrale établit la convergence en loi de la somme d'une suite de variables aléatoires vers la loi normale. Intuitivement, ce résultat affirme que toute somme de variables aléatoires indépendantes et identiquement distribuées tend au sens faible vers une variable aléatoire gaussienne.

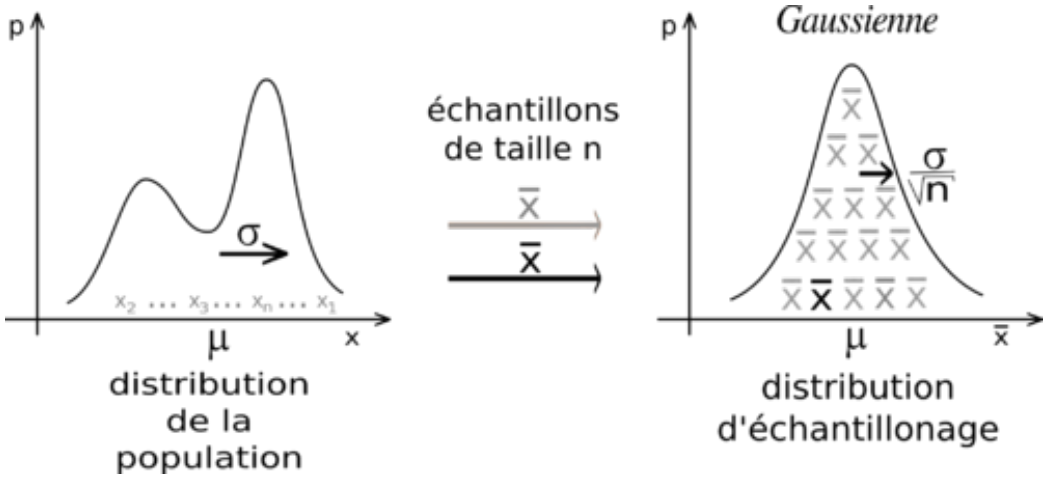

FIGURE 9 – Quelle que soit la forme de la distribution de la population, la distribution d'échantillonnage est gaussienne (Source : wikipedia)

Dans la pratique, si n est le nombre de tirage et σ l'écart-type de la distribution, $\sigma\sqrt{n}$ est constant ce qui revient à dire que $\log(\sigma)$ décroit linéairement en fonction de $\log(\sqrt{n})$ avec un coefficient directeur de -0.5. Nous aurons besoin de ce résultat par la suite afin de tester notre algorithme de calcul de fréquence de décharge.

3 Travail effectué

Le but du stage était d'utiliser la version du logiciel VirtualRetina intégrée au logiciel Enas afin de reproduire l'anticipation du mouvement décrite dans l'article de M. Berry & al., et de déceler le rôle du mécanisme de contrôle de gain dans cette anticipation.

3.1 Environnement logiciel

3.1.1 Prise en main de VirtualRetina

VirtualRetina est un logiciel qui peut simuler jusqu'à environ 100 000 cellules ganglionnaires en 100 fois le temps réel. Au delà de la simulation rétinienne, qui peut être utilisée pour mieux comprendre le fonctionnement de la rétine en conditions normales ou pathologiques, VR fournit aussi une "sortie" réaliste qui peut être ensuite utilisée comme entrée de modèles du LGN [1] ou du cortex visuel primaire. VirtualRetina peut se décomposer structurellement en 4 grands blocs. Le premier contient des gestionnaires de données. Le second comprend les différents filtres utilisés dans le programme ainsi que les fonctions d'échelonnage. Le troisième a pour principale fonction de générer les spikes en utilisant différents objets conçus sur la base de la librairie MvaSpike [2]. Et pour finir, on trouve la librairie xmlParamaters qui sert à charger et enregistrer les paramètres de la rétine et permet notamment de définir les paramètres des trois niveaux du modèle.

Le modèle implémenté par VirtualRetina possède trois composantes successives : un filtre linéaire qui modélise la couche plexiforme externe (OPL) [3] de la rétine, un mécanisme de contrôle de gain non-linéaire modélisant une interaction entre les cellules bipolaires et amacrines, et un processus d'émission de potentiels d'action au-niveau des cellules ganglionnaires. Le modèle impulsionnel retenu pour simuler ce processus est le : Leaky Integrate-and-Fire. (Cf 2.1.1)

1. Corps géniculé latéral, une partie du cerveau qui traite l'information visuelle provenant de la rétine

2. Outil polyvalent destiné à la modélisation et la simulation des réseaux complexes de neurones. Il est basé des événements qui modélisent les impulsions neuronales

3. Réseau dense de synapses entre les dendrites des cellules horizontales et les cellules photoréceptrices de la rétine

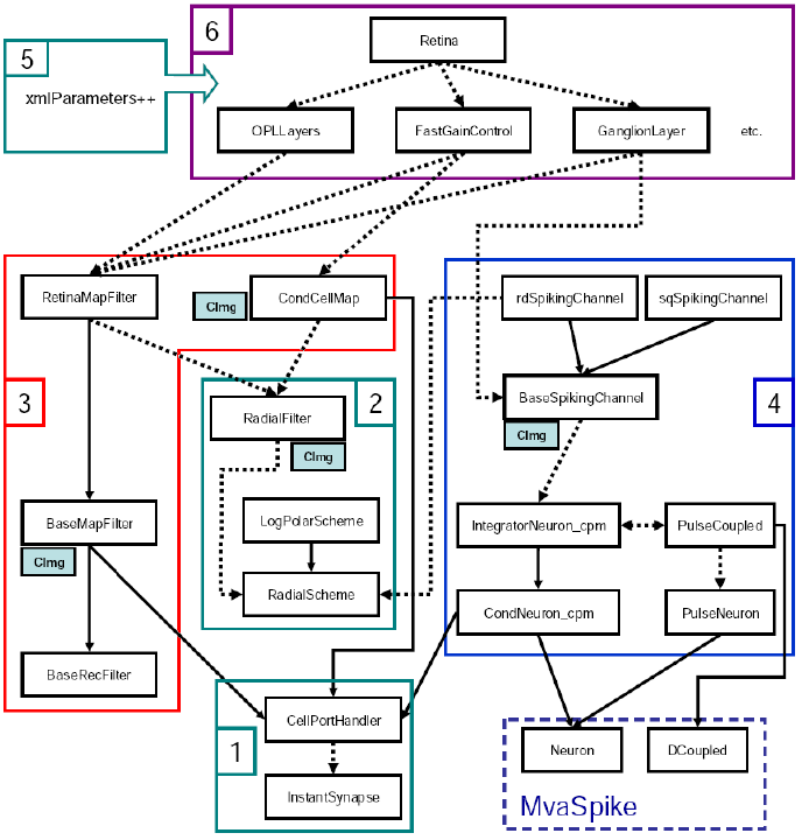

FIGURE 10 – Organisation des principaux objets de Virtual Retina. Les flèches indiquent un héritage et les flèches pointillées représentent une dépendance fonctionnelle.[14]

Simulation préliminaire

Depuis sa version originale, VirtualRetina a connu plusieurs modifications, une version simplifiée a notamment été intégrée à Enas, dans laquelle le Leaky Integrate and Fire est remplacé par le modèle à temps discret (*Cf* 2.1.1). Seul le Web service conserve la version originale. Nous avons donc décidé dans un premier temps d'utiliser le Web Service pour mieux appréhender le fonctionnement du modèle de base. N'ayant pas encore accès à la vidéo de l'expérience de Berry, nous avons décidé de reproduire l'animation d'une barre en translation uniforme à l'aide de java (*Cf* Annexe). Nous avons d'abord créé un gif d'une barre rouge en translation uniforme, mais les résultats n'étaient pas concluants (VirtualRetina n'est pas adapté à des gifs en couleur). La barre fut donc paramétrée en couleur noire. Celle-ci a une hauteur de 100 pixels

et une largeur de 30 pixels. Elle avance d'un pixel par unité de traitement. Nous lançons ensuite la fonction de simulation de VirtualRetina qui prend en entrée un Gif et un fichier xml de configuration de la rétine, puis la fonction de reconstruction qui produit le Gif de la réponse de la rétine simulée.

Configuration du fichier XML Les paramètres de la rétine dans VirtualRetina sont regroupés dans un fichier XML ordonné hiérarchiquement, dont l'élément père est simplement appelé : <retina/>. Comme dans n'importe quel arbre XML, un élément possède des attributs enfants sous la forme : param-name="value" et des éléments enfants. Les attributs correspondent à des valeurs numériques du modèle. Les élements enfants permettent de définir des structures partielles qui possèdent elles-même des paramètres sous forme d'attributs. L'architecture générale du fichier permet une organisation claire des paramètres de la rétine suivant l'ordre du modèle, mais également d'avoir une structure modulable où de nouvelles parties peuvent être ajoutées. (Figure 11)

```
<retina-description-file>

    <retina temporal-step__sec ="0.005"
        input-luminosity-range="255"
        pixels-per-degree="20.0">

        <log-polar-scheme fovea-radius__deg="1.0"
                  scaling-factor-outside-fovea__inv-deg="1.0"/>

        <outer-plexiform-layer>
            <linear-version
            center-sigma__deg="0.03"
            surround-sigma__deg="0.1"
            center-tau__sec="0.01"
            center-n__uint="2"
            surround-tau__sec="0.01"
            opl-amplification="10"
            opl-relative-weight="1" />
        </outer-plexiform-layer>

        <ganglion-layer
            sign = "1"
            transient-tau__sec="0.03"
            transient-relative-weight="0.7"
            bipolar-linear-threshold="0"
            value-at-linear-threshold__Hz="80"
            bipolar-amplification__Hz="100">
            <spiking-channel>
                <circular-spiking-channel
                fovea-density__inv-deg="20.0"
                g-leak__Hz="50" sigma-V="0" refr-mean__sec="0.003"
                refr-stdev__sec="0" random-init="1"/>
            </spiking-channel>
        </ganglion-layer>

    </retina>
</retina-description-file>
```

FIGURE 11 – Exemple de configuration du fichier XML de description de la rétine Primate_Parvo utilisé dans la simulation préliminaire

Résultats

Nous avons pu constater sur le Gif réponse généré par la fonction de reconstruction de la rétine une accélération de la barre au centre du champs visuel, phénomène qui pourrait rendre compte d'une anticipation du mouvement. En effet, la distance entre la barre du stimulus et celle de la réponse diminue à mesure que celle-ci translate.(Figure 12)

Afin de vérifier si VirtualRetina reproduit bien l'anticipation du mouvement comme décrite dans l'article de M. berry, nous avons besoin de reproduire le même stimulus, de caractériser les champs récepteurs des neurones utilisés dans l'expérience de Berry afin de pouvoir configurer VirtualRetina, de calculer la fréquence de décharge des neurones après la simulation, et de confronter nos résultats à ceux obtenus par M. Berry.

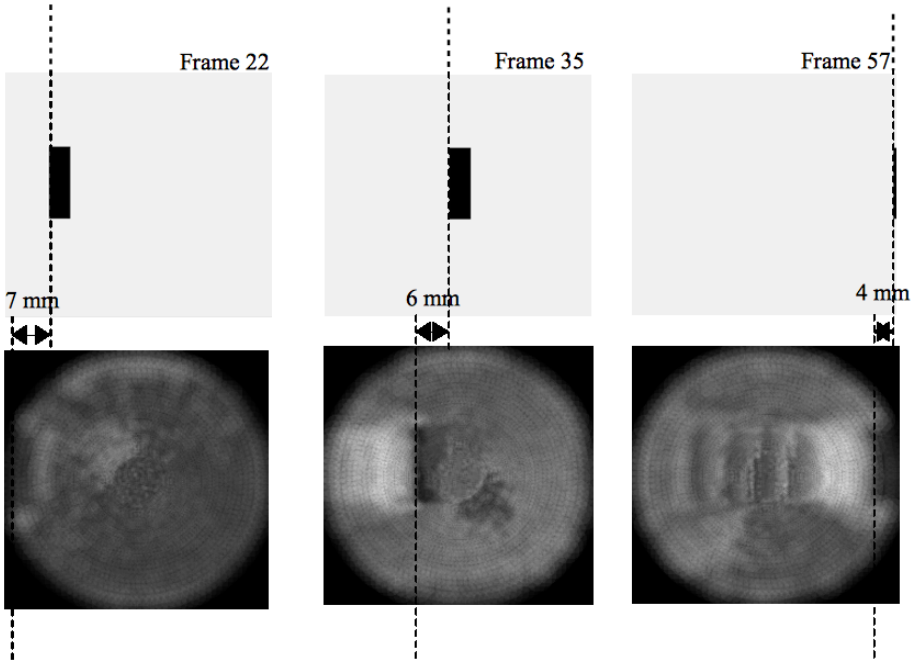

FIGURE 12 – Evolution de la distance entre la barre du stimulus et la barre de la réponse rétinienne reconstruite par VirtualRetina

3.1.2 Prise en main d'Enas

Enas est un logiciel d'analyse des trains de spikes [4] issus d'expériences biologiques ou de simulateurs neuronaux. Le logiciel utilise des outils stastiques appliqués aux neurosciences afin d'analyser le comportement de différents réseaux de neurones. Le logiciel permet notamment des calculs de corrélation, la production du Peri-Stimulus Time Histogramm, l'estimation des champs récepteurs, et l'analyse des interactions spatio-temporelles dans des populations de neurones.(Cf 2.1)

Afin d'importer le code source Enas de gforge (plateforme INRIA de développement participatif), il faut configurer ssh [5] puis utiliser svn [6] pour un premier import. SVN permet de faire des mises à jour régulières d'Enas pour prendre en compte les modifications apportées au code source. La correction des bugs d'Enas passe par l'utilisation de Trello, un outil de gestion de projet en ligne. Cet outil permet à l'équipe de développement de définir les bugs à corriger avec les différents statuts : à faire, en cours et fait, comme dans une to-do-list classique.

Au début du stage, O. Marre (IDV) nous a envoyé les données expérimentales relatives à l'article de M. Berry et un script Matlab pour les charger afin de les comparer aux résultats de simulation. Le code Matlab reçu comporte également une fonction DoPSTH qui génère le PSTH à partir de plusieurs variables telles que SpikeTimes et Amplitudes. Le PSTH obtenu montre une concentration d'activité entre 100 et 150 ms puis aux alentours de 300 ms.(Figure 13)

Afin de pouvoir utiliser les données du code Matlab dans Enas, un script Matlab a été écrit pour générer un fichier texte de format Time Unit [7] à partir de la variable SpikeTimes de format Cell. Le PSTH que nous obtenons avec Enas (Figure 14) prend en compte tous les évenements tandis que la fonction Matlab calcule une moyenne de la fréquence de décharge. On peut notamment remarquer que la valeur maximale calculée par Enas (Figure 14) est supérieure au double de celle calculée par la fonction Matlab (Figure 13), et que les distributions ne sont pas équivalentes. Après avoir échangé avec

4. Séries de signaux électriques temporelles enregistrés à partir des neurones. Ils représentent essentiellement les potentiels d'action (influx nerveux) générés par les neurones utilisés pour communiquer entre eux.

5. Protocole de communication sécurisé

6. Logiciel de gestion de versions, distribué sous licence Apache et BSD. Il s'appuie sur le principe du dépôt centralisé et unique.

7. La première colonne correspond aux identifiants des neurones qui ont tiré et la deuxième aux temps des spikes.

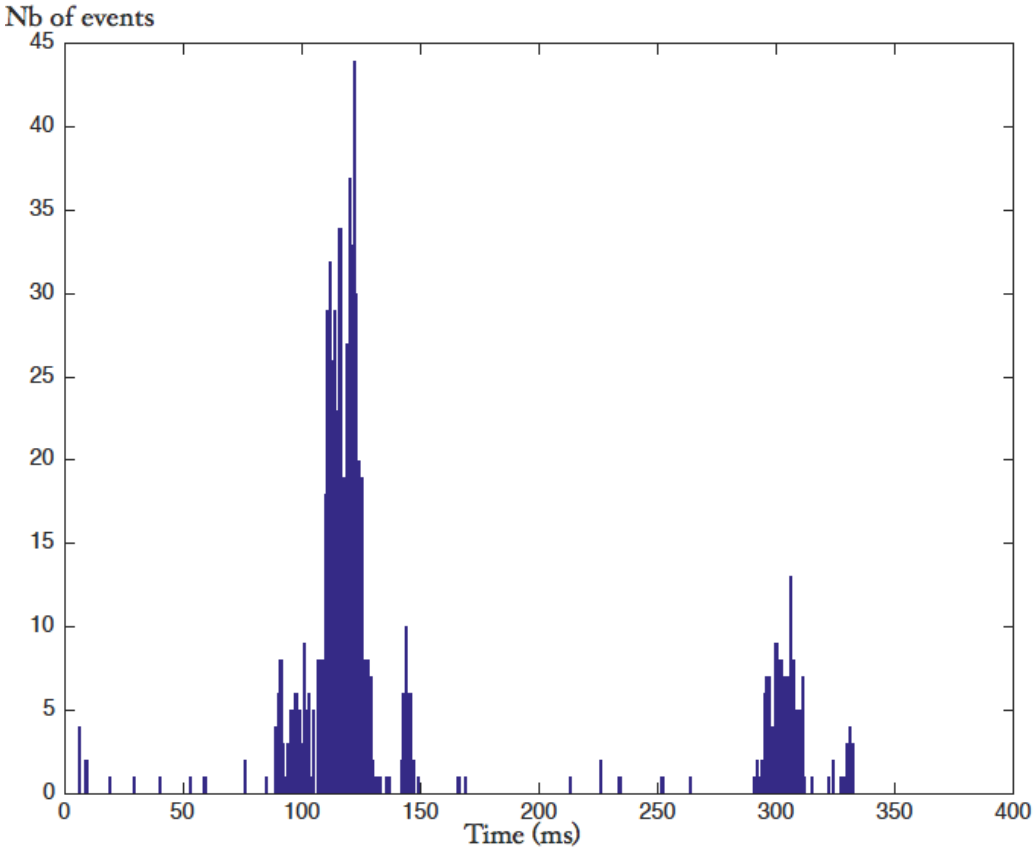

FIGURE 13 – PSTH calculé par Matlab

Daniela Pamplona, chercheuse posdoctorale au sein de mon équipe, et après concertation avec B. Cessac, le code d'Enas a été modifié de manière à diviser l'histogramme par une taille de fenêtre puis par le nombre de neurones afin d'obtenir une moyenne et d'avoir le même rendu que la fonction Matlab.

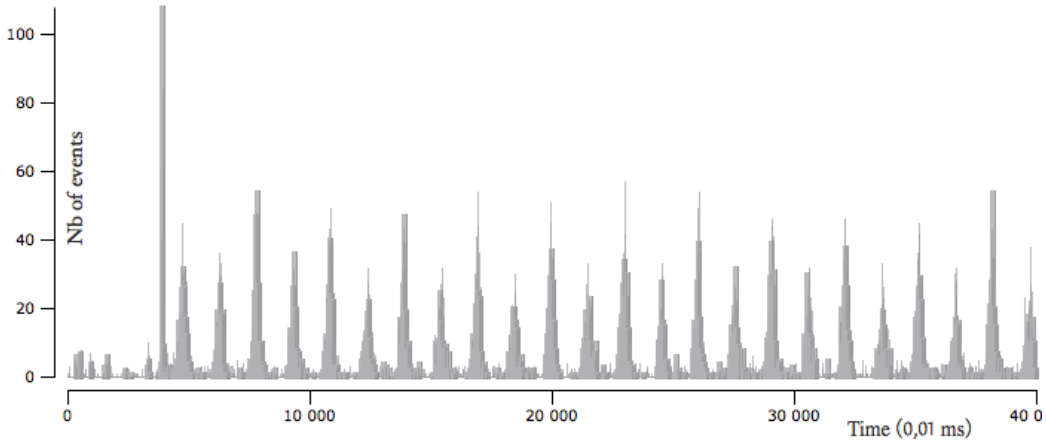

FIGURE 14 – PSTH calculé par Enas

3.2 Développement

3.2.1 Reproduction du stimulus de la barre en mouvement

Stéphane Deny (IDV) nous a envoyé les fichiers Matlab nécessaires à la reproduction du stimulus de la barre en mouvement utilisé dans l'article de M. Berry. Le stimulus consiste en une fenêtre au fond blanc de 60*60 pixels avec une barre noire centrée verticalement d'une longueur de 15 pixels et d'une largeur de 5 pixels, en translation uniforme. La barre se déplace d'un pixel toutes les deux frames à partir de la frame 31. Le rafraîchissement se fait à une fréquence de 60 Hz. Nous avons 180 frames ce qui correspond à 3 s. Toutes ces informations peuvent être retrouvées grâce à un fichier Matlab qui contient le nombre de frames, leurs temps d'affichage et une matrice binaire de représentation spatiale (le blanc étant codé par 1 et le noir par 0). La séquence d'images est ensuite créée grâce au code java écrit précédemment, auquel nous avons intégré une méthode ScreenImage pour les captures de fenêtre (Cf Annexes). Le fichier gif est ensuite construit grâce à Gimp [8] avec un pas temporel de 16 ms.

3.2.2 Algorithme de calcul de la fréquence de décharge

Nous avons d'abord développé un algorithme simple non récursif pour calculer la fréquence de décharge le long d'une fenêtre temporelle glissante.

8. Outil d'édition et de retouche d'image.

La méthode consiste à prendre une fenêtre d'une largeur fixe, de compter le nombre de neurones inclus dans la fenêtre et de diviser par sa largeur. Nous faisons glisser la fenêtre et nous réitérons le procédé. Nous avons ensuite développé un algorithme dynamique qui permet de calculer la fréquence de décharge de manière récursive. Nous commençons d'abord par stocker les temps de spikes du fichier texte dans une file qui sera ensuite parcourue lors du traitement. Au début, nous plaçons le curseur une largeur de fenêtre avant le temps du premier spike. La fréquence de décharge sera donnée par le nombre de spikes comptés divisé par la largeur de la fenêtre. Le curseur avançant d'un pas, la fréquence de décharge est calculée de manière récursive en testant la position du curseur par rapport aux éléments de la file. Si le curseur est décalé d'un pas à droite de la référence, la fréquence de décharge est diminuée d'un spike, la référence n'étant plus comprise dans la fenêtre, et augmenté du nombre de spikes rencontrés entre la fin de la fenêtre antérieure et la fin de la fenêtre actuelle. Sinon, seule l'addition est effectuée. (Cf Annexes) Les résultats de cet algorithme ont été comparés à ceux de l'algorithme simple non récursif, puis validé à l'aide un processus de Poisson.

Génération des spikes avec le processus Poissonien

Nous allons commencer par supposer que la fréquence de décharge instantanée f est temporellement constante, autrement dit un processus de Poisson homogène. La probabilité qu'un spike ait lieu sur un intervalle δt est égale à $f\delta t$. Il faut donc commencer par générer des nombres aléatoires r uniformément distribués entre 0 et 1. Nous utiliserons pour cela le générateur de nombres pseudo-aléatoires rand() et la fonction srand() pour modifier la graine à partir de laquelle le programme initialise le nombre aléatoire. Pour chaque intervalle δt , si $r < \delta t$ on génère un spike. Cette procédure est adaptée lorsque δt est suffisamment petit. (Cf Annexes)

La deuxième étape était de générer un processus de Poisson inhomogène, en remplaçant la fonction constante par une fonction qui varie au cours du temps $f(t)$. Nous choisirons ici la fonction sinus. Dans les deux cas, nous n'avons pas tenu compte de la période réfractaire.

Afin de vérifier la justesse de notre algorithme de génération du processus de Poisson, nous avons tracé la variation du logarithme de l'erreur de la variable aléatoire "nombre de spikes par tirage" en fonction du logartithme du nombre de tirages. Nous utilisons ici l'estimateur de la variance non biaisé donné par : $\frac{1}{n-1}\sum_{i=1}^{n}(X_i - m)^2$. Nous insérons ensuite une courbe de tendance linéaire et nous obtenons un coefficient directeur de -0.5 : le théorème de la limite centrale est donc vérifié. (Figure 15)

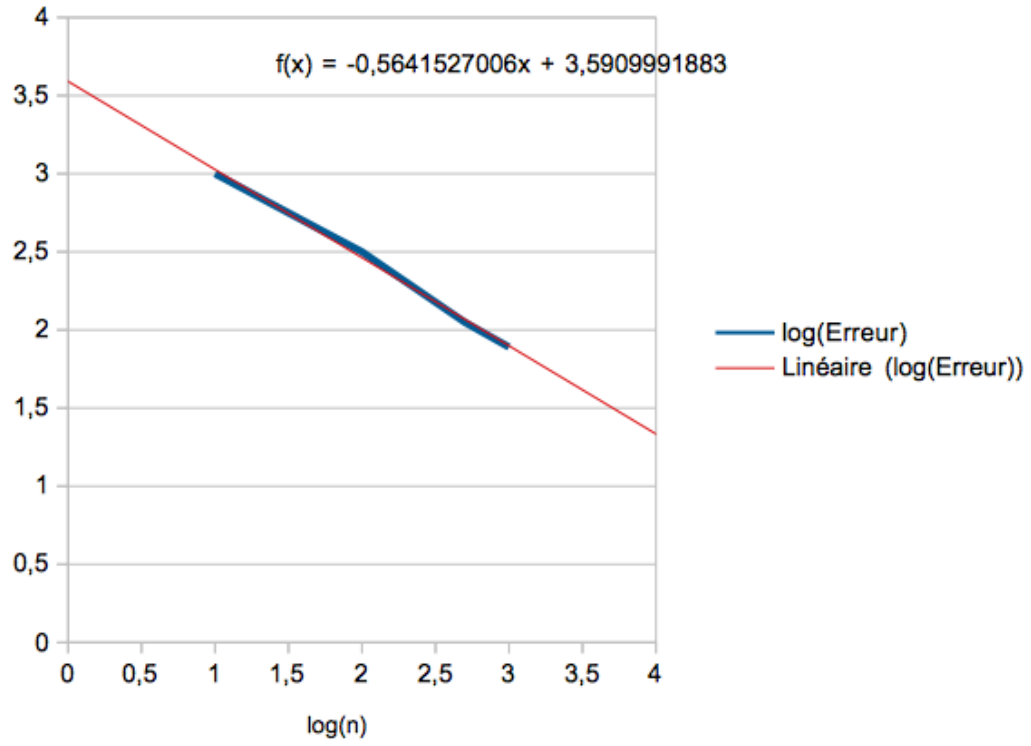

FIGURE 15 – Variation de l'écart type de la moyenne en fonction du nombre de tirages

Vérification de l'algorithme de calcul des fréquences de décharge avec le processus de Poisson Afin de valider notre algorithme, nous l'utilisons d'abord pour calculer les fréquences de décharge de 10 tirages de notre processus de Poisson homogène (chaque tirage génère un fichier texte des temps de spikes organisés en colonne). Nous calculons ensuite une moyenne empirique des fréquences de décharge des 10 tirages correspondant à chaque position de la fenêtre, puis l'écart type non biaisé de cette moyenne. Nous avons choisi 3 comme valeur de la fréquence de décharge instantanée utilisée pour générer le processus de Poisson. Nous traçons ensuite la courbe des fréquences de décharge résultats et celle de la fréquence de décharge instantanée, et comme prévu, les barres d'erreur des fréquences de décharge résultats entourent la valeur de la fréquence de décharge instantanée. (Figure 16) Nous reprenons ensuite la même étude dans le cas d'un processus de Poisson inhomogène.(Figure 17) Les résultats nous permettent de valider notre algorithme.

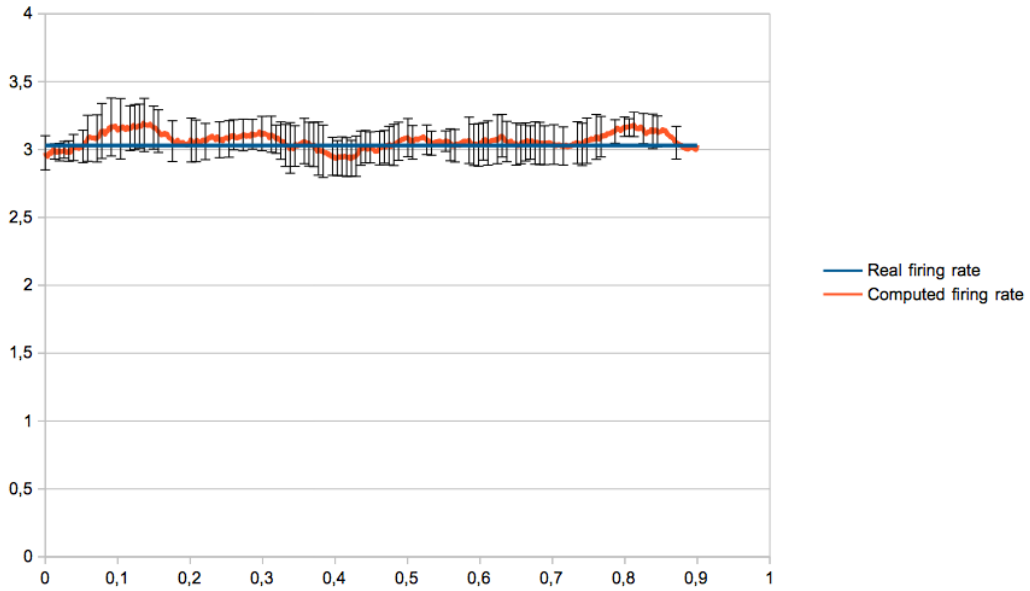

FIGURE 16 – Fréquences de décharge calculées dans le cas d'un processus de Poisson homogène

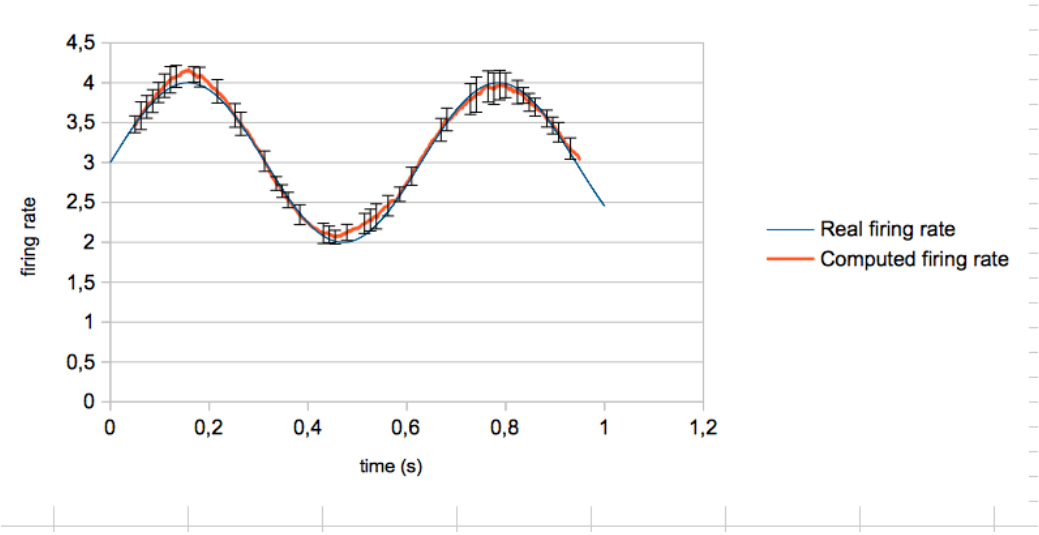

FIGURE 17 – Fréquences de décharge calculées dans le cas d'un processus de Poisson inhomogène

3.2.3 Caractérisation des champs récepteurs

Afin de configurer la rétine émulée par VirtualRetina, nous avons besoin de connaître les paramètres des champs récepteurs des neurones utilisés dans l'expérience de Berry. Olivier Marre (IDV) nous a envoyé des fichiers Matlab desquels nous pouvont extraire les paramètres des champs récepteurs et les adapter aux paramètres du modèle utilisé dans VirtualRetina.

La partie qui nous intéresse particulièrement, et qui va nous permettre de définir les champs récepteurs, est la couche plexiforme externe (Outer Plexiform Layer). C'est la base du traitement rétinien, celle où l'architecture en centre-contour de la rétine apparaît. Il existe plusieurs version du modèle, celle que nous utiliserons dans un premier temps est la version linéaire.

Le champ récepteur spatio-temporel est enregistré dans une variable globale RFtot en 4D. Deux projections ont été faites de cette variable, la première sur les dimensions spatiales dans une variable Spatial en 3D et une deuxième sur le profil temporel. Nous nous intéressons dans un premier temps à la variable Spatial. Celle-ci contient 601 sections temporelles dont 280 correspondent aux champs récepteurs des 280 neurones enregistrés dans l'expérience de Berry. Les indices des sections concernées sont à 1 dans une variable Tagged, les autres sont à 0. Nous devons donc faire un ajustement de ces données brutes avec une différence de gaussiennes afin de définir les paramètres des champs récepteurs. Nous choisissons dans un premier temps, dans un soucis de simplification, des champs récepteurs concentriques. La partie spatiale des champs récepteurs est donnée par l'équation suivante :

$\frac{a}{\pi * \sigma_1^2} \exp(-(\frac{(x-b)^2}{2*\sigma_1^2} + \frac{(y-c)^2}{2*\sigma_1^2})) - \frac{d}{\pi \sigma_2^2} * \exp(-(\frac{(x-b)^2}{2*\sigma_2^2} + \frac{(y-c)^2}{2*\sigma_2^2}))$

avec a, b, c, d, e et f les paramètres trouvés par le fit.

Nous écrivons ensuite un script Matlab pour faire l'ajustement. (Cf Annexes)
Nous commençons par chercher la valeur absolue maximale de la hauteur z de section par laquelle nous forçerons la courbe d'ajustement à passer. Nous appliquons ensuite un filtre médian [9] afin d'améliorer le rapport signal sur bruit. Enfin nous faisons un ajustement avec l'équation de la différence de gaussiennes introduite auparavant. Certaines données étant plus bruités que d'autres, nous obtenons des résultats plus ou moins exploitables. La première section par exemple nous permet d'obtenir un bon ajustement. Nous obte-

9. Filtre numérique non linéaire, souvent utilisé pour la réduction de bruit. L'idée principale du filtre médian est de remplacer chaque entrée par la valeur médiane de son voisinage.

nons en effet un R-square [10] de 0,9109 (la valeur maximale étant de 1 pour un ajustement parfait) et un RMSE [11] de 0.006118 ce qui représente une erreur quadratique très faible.(Figure 18)

La section 6 quant à elle est très bruitée, et même après filtrage ne permet pas d'avoir un résultat utilisable.Le R-square a une valeur 0.04232 et le RMSE de 0.05272, d'où une mauvaise qualité du fit. (Figure 19)

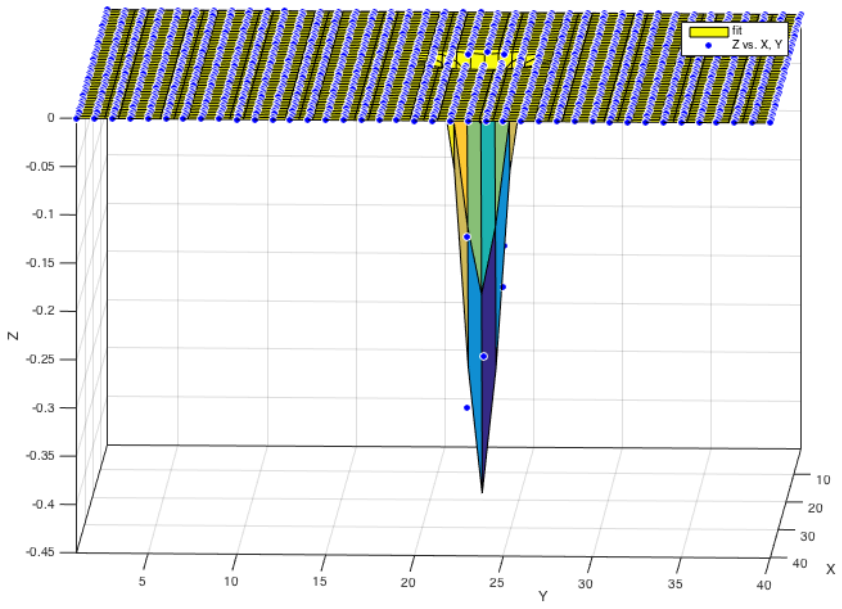

FIGURE 18 – Exemple d'un bon ajustement de la section temporelle avec Matlab

10. Coefficient de détermination, mesure de la qualité de prédiction d'un ajustement.Il est défini comme 1 moins le ratio entre l'erreur avec les valeurs prédites et la variance des données

11. Ou RMSD, déviation de la racine de la moyenne des carrés, est une mesure qui consiste à comparer, grâce à une formule propre, des valeurs théoriques avec des valeurs observées pour voir à quel point le modèle théorique représente bien la réalité. Plus le RMSD est faible, plus le modèle est réaliste.

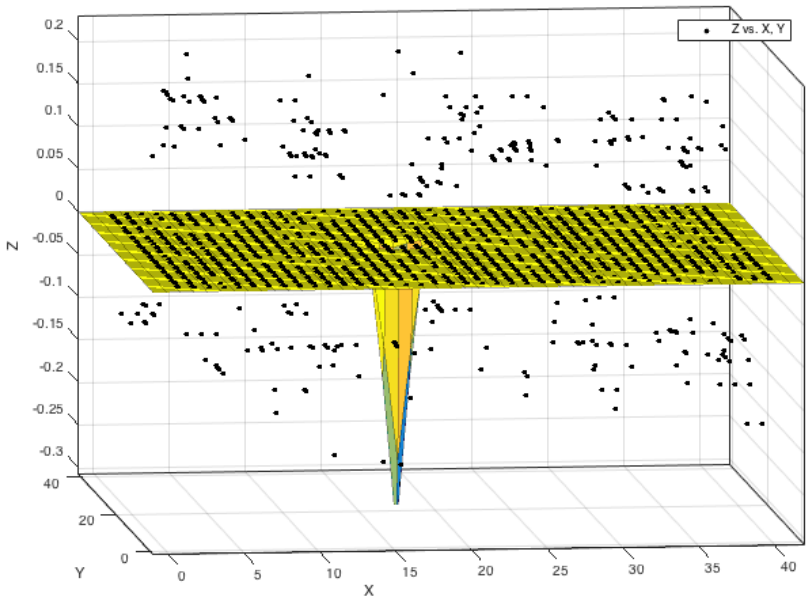

FIGURE 19 – Exemple d'un mauvais ajustement de la section temporelle

La projection du champ récepteur sur l'espace temporel caractérise l'influence d'un stimulus perçu il y a un certain temps τ sur le champ récepteur de la cellule. Le noyau temporel est souvent modélisé par l'équation :

$$D(\tau) = \alpha \exp(-\alpha\tau)\left(\frac{(\alpha\tau)^5}{5!} - \frac{(\alpha\tau)^7}{7!}\right)$$

Les profiles temporels non nuls des données reçues ont pratiquement tous la même allure.

Pour faire un ajustement de la projection temporelle des champs récepteurs, nous utilisons cette fois-ci l'utilitaire d'ajustement Matlab Cftool. Nous réduisons la projection à la partie qui nous intéresse afin d'optimiser l'ajustement, puis nous la multipilions par -0.01 pour que la portion la plus haute soit positive et pour respecter également la forme propre au noyau temporel. Nous obtenons des ajustements avec des R-square autour de 0.5, valeur assez faible mais acceptable.(Figure 20)

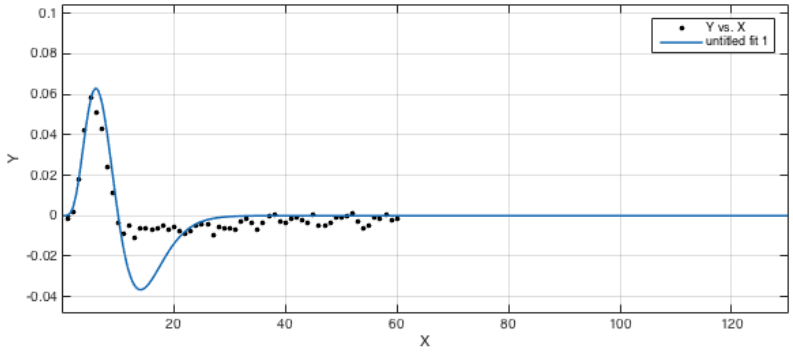

FIGURE 20 – Ajustement de la projection temporelle

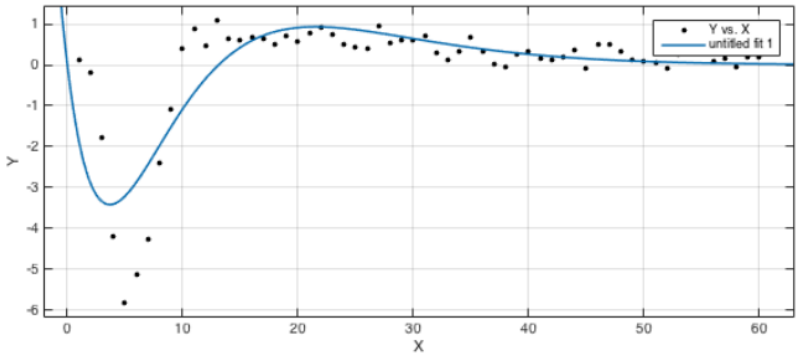

FIGURE 21 – Ajustement de la projection temporelle en utilisant la partie temporelle du noyau utilisé dans VirtualRetina

Les projections temporelles des champs récepteurs étant similaires pour toutes les cellules, nous devons étudier la répartition des neurones selon les valeurs des variables intervenant dans l'équation spatiale. Cette répartition nous permet de constater que les valeurs les plus communes pour σ_1 se situent dans l'intervalle [0.5 ;1] et pour σ_2 dans l'intervalle [1 ;1.5].

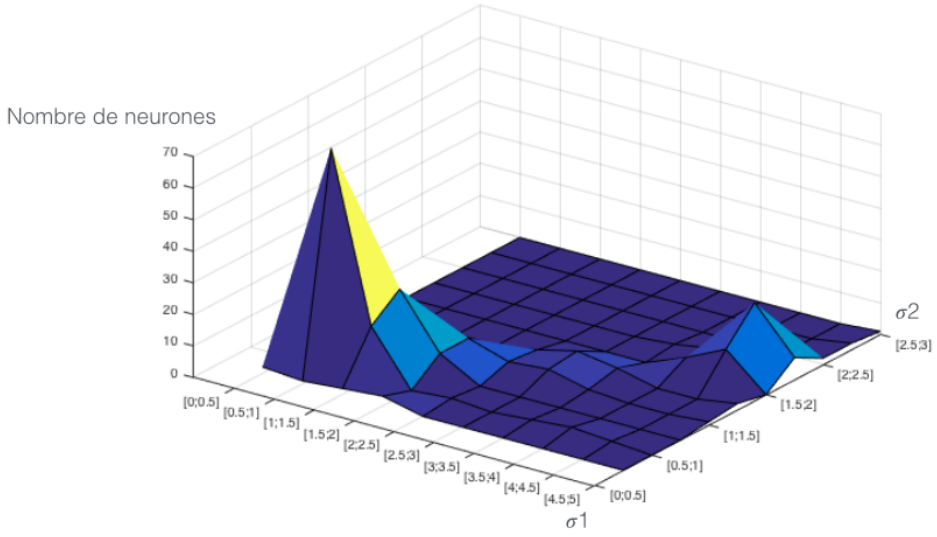

FIGURE 22 – Répartition des cellules selon les valeurs de σ_1 et σ_1

3.2.4 Reproduction de l'expérience avec VirtualRetina

Afin d'étudier le rôle du mécanisme de contrôle de gain dans l'anticipation du mouvement telle que décrit par M. Berry, nous avons décidé d'effectuer deux simulations sur VirtualRetina, avec et sans mécanisme de contrôle de gain. Nous avons notamment utilisé les paramètres trouvés grâce aux différents ajustements, et choisi d'étudier les cellules se situant au centre de la rétine. Les pics de réponse des cellules ganglionnaires paramétrées dans l'expérience sans mécanisme de contrôle de gain interviennent au même instant, et l'activité est initiée après le moment du flash. Dans l'expérience avec mécanisme de contrôle de gain, la réponse des cellules commence avant le moment du flash mais les pics d'activité continuent à se superposer en temps, intervenant légèrement plus tôt que dans la première expérience mais toujours après le moment du flash. Le mécanisme de contrôle de gain tel qu'implémenté par VirtualRetina ne permet donc pas de reproduire l'anticipation du mouvement. Nous allons donc étudier le mécanisme de contrôle de gain utilisé par M. Berry & al. et le substituer à celui de VirtualRetina.(Figure 25)

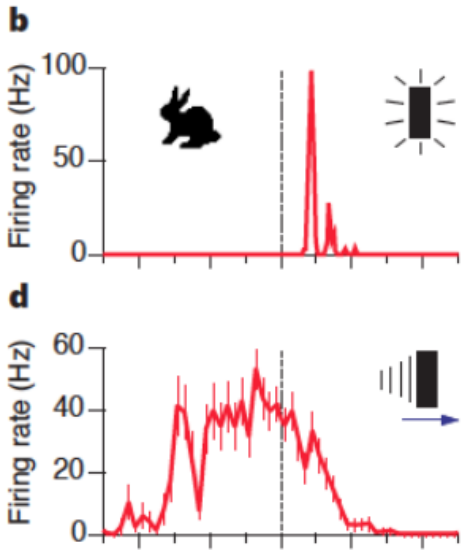

FIGURE 23 – Expérience de M. Berry : réponse des cellules ganglionnaires de lapin de type fast OFF à une barre flashée et une barre en mouvement.[1]

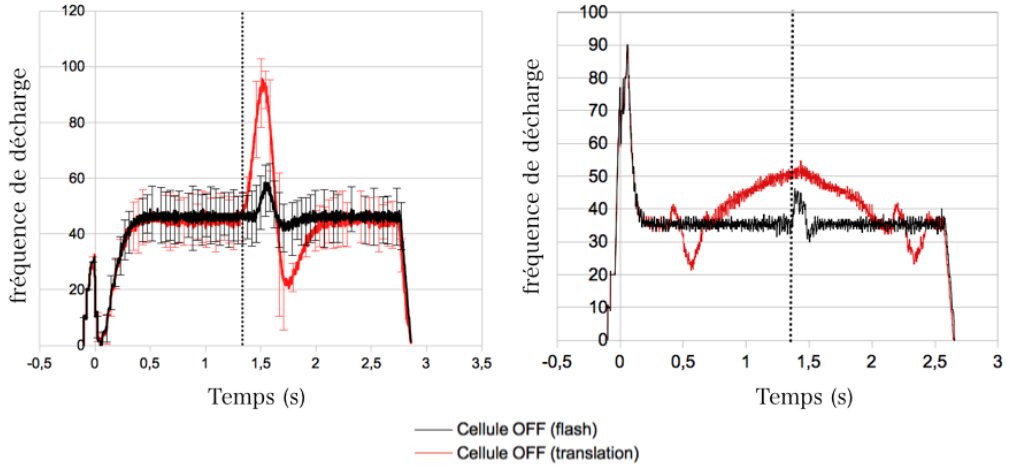

FIGURE 24 – Résultats des simulations avec VirtualRetina sans et avec mécanisme de contrôle de gain. En présence du contôle de gain, la fréquence de décharge de la cellule commence à augmenter avant le moment du flash, mais le pic de réponse de réponse ne survient toujours pas avant le flash.

3.2.5 Etude et reproduction du mécanisme de contrôle de gain utilisé par M. Berry & al.

La réponse d'une cellule ganglionnaire à un stimulus est donnée par la convolution de ce stimulus et du noyau spatio-temporel représentant le champ récepteur suivi par une non-linéarité. Dans l'article de M. Berry, le mécanisme de contrôle de gain consiste à appliquer au résultat de la convolution un filtre exponentiel de rétroaction suivi par une fonction décroissante g(v). Les équations utilisées sont :

$$u(t) = g(v) \int_{-\infty}^{\infty} dx \int_{-\infty}^{t} dt'\, s(x,t') k(x,t-t')$$

$$v(t) = \int_{-\infty}^{t} dt'\, u(t') B \exp\left(-\frac{t-t'}{\tau}\right)$$

$$g(v) = \begin{cases} 1 & v < 0 \\ 1/(1+v^4) & v > 0 \end{cases}$$

$$F(u) = \begin{cases} 0 & u < \theta \\ \alpha(u-\theta) & u > \theta \end{cases}$$

où u(t) représente la convolution spatio-temporelle du stimulus au champ récepteur. Le résultat passe ensuite par une boucle de rétroaction. Il est d'abord convolué au filtre exponentiel, on obtient alors v(t), qui est ensuite soumise à la fonction de contrôle de gain g(v). u(t) est ensuite corrigée par une non-linéarité F(u) qui permet d'obtenir la fréquence de décharge r(t).

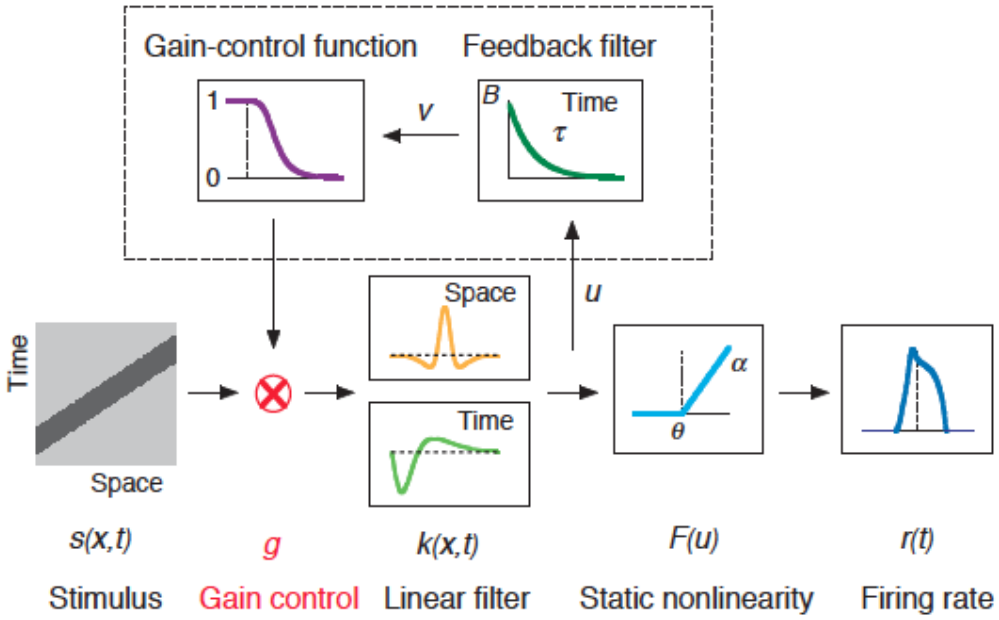

FIGURE 25 – Modèle de réponse d'une cellule ganglionnaire. Le filtre k(x,t) correspond au champ récepteur. Le modèle possède 4 paramètres : le seule θ et le coefficient directeur α de la non-linéarité, l'amplitude B et la constante de temps τ du filtre de rétroaction.[1]

Afin de simuler le mécanisme, nous utilisons la librairie GNU Scientific Library (GSL). La GSL fournit des outils de calcul numériques en mathématiques appliquées. Nous aurons notamment besoin de calculer des intégrales et de résoudre une équation différentielle. La forme intégrale proposée par Berry & al. est peu pratique d'un point de vue numérique. De fait, c'est une convolution et il est plus facile de la résoudre sous forme différentielle. En effet, l'intégrale qui définit la fonction v(t) peut être écrite sous forme d'une équation différentielle :

$$v(t) = \int_{-\infty}^{t} Bu(t') \exp(-\frac{t-t'}{\tau})dt'$$

$$\iff e^{\frac{t}{\tau}}v(t) - e^{\frac{t_0}{\tau}}v(t_0) = \int_{-t_0}^{t} Bu(t')\exp(\frac{t'}{\tau})dt' \quad avec \quad t_0 - > -\infty$$

$$\iff \frac{d}{t}(e^{\frac{t}{\tau}}v(t))) = Be^{(}\frac{t}{\tau})u(t))$$

$$\iff \frac{dv}{t} = \frac{v}{\tau} + Bu(t)$$

$$u(t) = [s * k]g(v) \quad donc \quad \frac{dv}{t} = \frac{v}{\tau} + B[s * k]g(v)$$

Le programme simulant le contrôle de gain consiste donc à itérer le processus suivant : calculer g(v) (v est donnée par une condition initiale), calculer l'intégrale double qui définit u(t), puis résoudre l'équation différentielle de v(t). (Cf Annexes)

Afin de simuler la barre en translation et de tester l'algorithme avant son implémentation dans Enas, on définit la fonction du stimulus s(x,t) comme une fonction porte dont le centre translate en fonction du temps. Le pas de translation est choisi égal au centième du pas d'intégration (dt=0.01s), de manière à avoir, une vitesse de 10 mm.s-1, si l'on considère que toutes les équations utilisent le système international d'unités. La hauteur de la fonction porte équivaut quant à elle à l'intensité du stimulus, qui correspond également au contraste étant donné que l'arrière plan blanc a une valeur d'intensité constante M=11 mWm-2. Pour une barre d'intensité I, la valeur du contrast est donnée par : $C = \frac{M-I}{I}$.

Dans un premier temps, nous étudions la relation entre la valeur du contraste, autrement dit la hauteur de la fonction porte, et l'instant où survient le pic de réponse de la cellule. Nous utilisons pour cela les mêmes paramètres que M. Berry & al., et les champs récepteurs que nous avons caractérisé précédemment. Les résultats montrent que l'anticipation du mouvement est d'autant plus importante que le contrast est élevé, le pic d'activité de la cellule ayant lieu plus tôt avec une une fréquence plus élevée.(Figure 27)

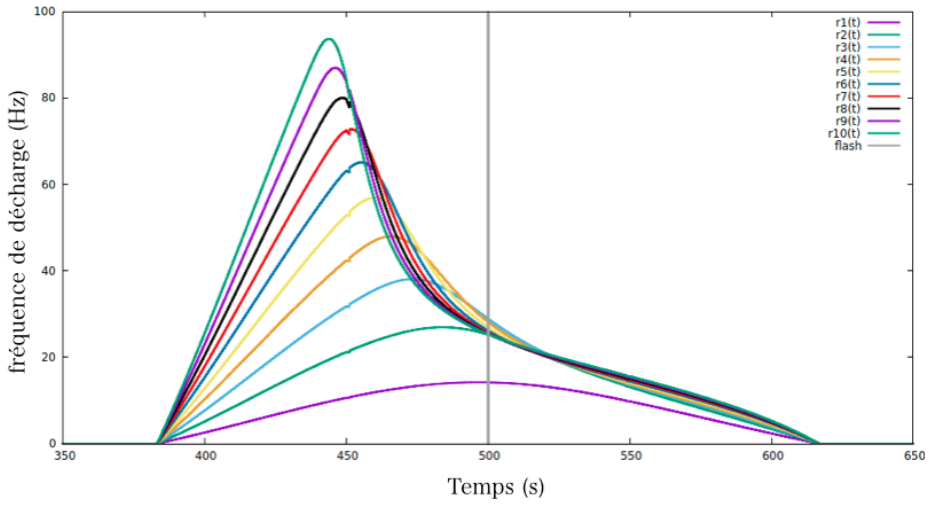

FIGURE 26 – Résultats : fréquence de décharge d'une cellule ganglionnaire de type OFF en fonction du contraste. Les valeurs du contraste varient entre 0.09 (courbe violette basse) et 0.91 (courbe verte haute).

Nous étudions ensuite le comportement spatial d'une population de 10 neurones se situant au centre de la rétine en faisant varier les paramètres des champs récepteurs.

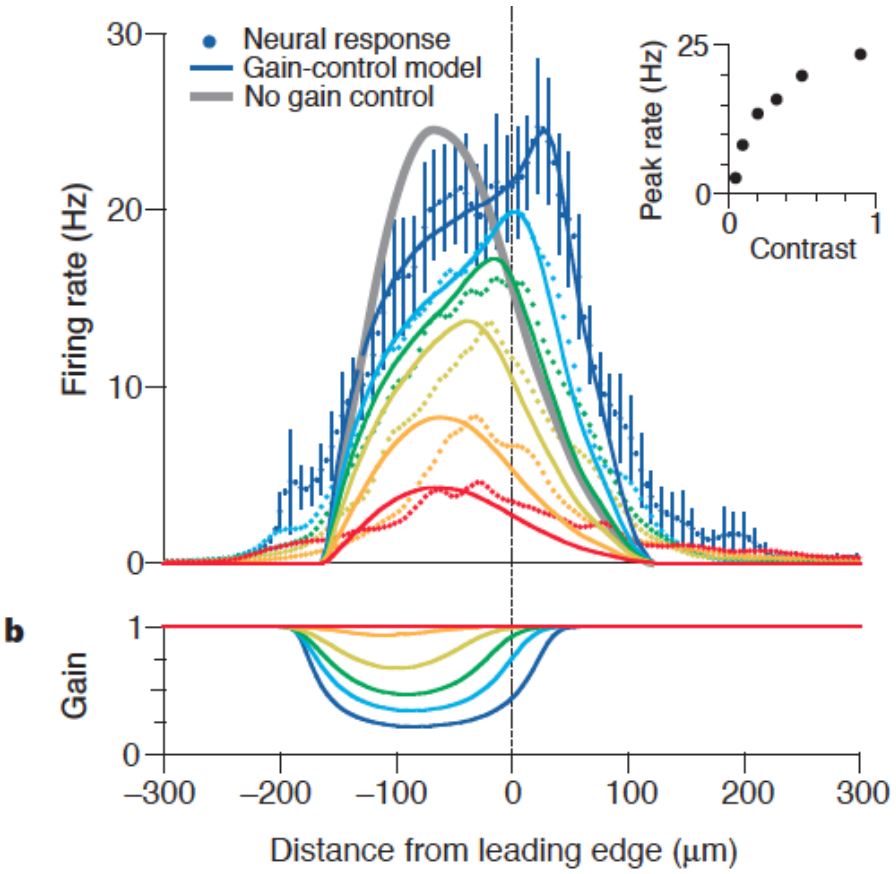

FIGURE 27 – L'extrapolation du mouvement dépend du contraste. Simulation avec des barres sombres en faisant varier le contraste : 5, 10, 20, 33, 50 et 90%. Les paramètres du modèle sont : $\theta = 0$, $\alpha = 85Hz$, $B = 45s^{-1}$ et $\tau = 170ms$. [1]

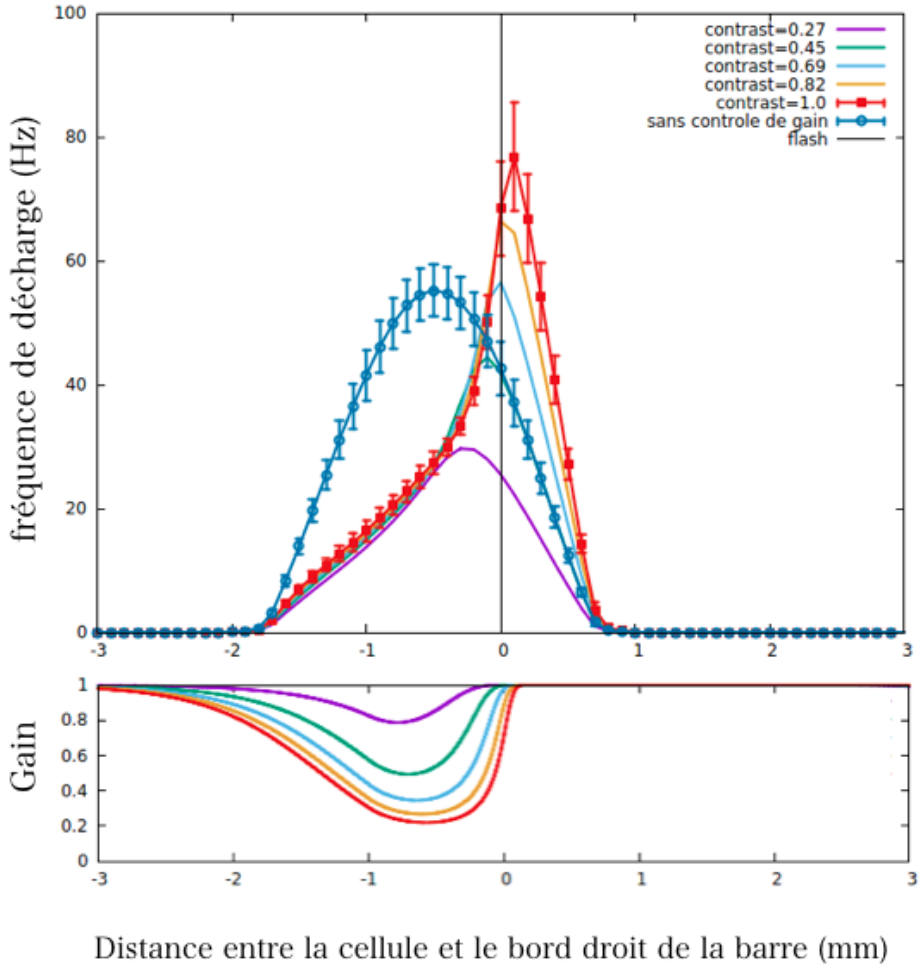

FIGURE 28 – Résultats de la reproduction de l'expérience de M. Berry avec des valeurs de contraste allant de 0.27 à 1.0

Les résultats que nous obtenons sont similaires à ceux obtenus par M. Berry. Nous choisissons une échelle de distance égale à 10 fois celle utilisée par M. Berry étant donné que la vitesse de translation dans notre cas est égale à 10 mm.s-1 alors qu'elle est limitée dans l'article à 1.76 mm.s-1. La différence d'échelle peut également être justifiée par un facteur multiplicatif intervenant entre les paramètres des champs récepteurs car nous n'avons pas accès aux valeurs exactes utilisées dans l'article. L'expérience de M. Berry & al. a été reproduite par O. Marre (IDV). Ce sont en effet les enregistrements biologiques qu'il nous a fourni qui nous ont permis de déterminer les valeurs des paramètres de nos fonctions du champ récepteur spatio-temporel, mais nous ne sommes pas sûrs si les fonctions utilisées ressemblent à celles utilisées par M. Berry & al.

Enfin, la différence dans la partie temporelle du noyau de convolution pourrait expliquer la différence entre les valeurs maximales des fréquences de décharge.

3.2.6 Utilisation de VirtualRetina intégré à Enas

Le logiciel VirtualRetina n'étant plus maintenu, l'équipe Biovision l'a intégré à Enas. La seule modification dans le code de VirtualRetina concerne la couche des cellules ganglionnaires, où le modèle Integrate and Fire de la version originale a été remplacé par le modèle à temps discret (BMS). Avant de reproduire l'expérience de Berry avec cette version de VirtualRetina, nous devons d'abord effectuer les conversions nécessaires entre les variables du modèle Integrate and Fire et celles du modèle à temps discret (BMS). En effet, le modèle BMS, initialement étudié mathématiquement, utilisait des unités arbitraires en posant notamment la capacité membranaire égale à 1. Ici, puisqu'il s'agit de reproduire des grandeurs physiques réalistes, on doit fixer les paramètres du modèle conformément à la biophysique. Nous devons nous assurer ensuite du bon fonctionnement du modèle BMS en utilisant les valeurs de conversion, et reproduire enfin une des expériences utilisées pour valider la version originale afin de valider la nouvelle version.

Calcul des paramètres du BMS à partir des valeurs des paramètres de VirtualRetina

L'équation générale du potentiel de membrane utilisée dans VirtualRetina pour la couche des cellules bipolaires et celle des cellules ganglionnaires est la suivante :

$$C.\frac{dV}{dt} = \sum_i I_i - \sum_j g_j(V - V_j)$$

avec V le potentiel de membrane en Volts, C la capacité de membrane en Farads dont la valeur standard est d'environ 20 pF, I_i les courants synaptiques en Ampères dont la valeur typique est d'environ 100 pA et V_j le potentiel de membrane dont la valeur typique est -70mV.

En posant $\sum_i I_i = I(t)$, $\sum_j g_j(V - V_j) = g_L(V - V_L)$ (en général, cette somme contient beaucoup plus de termes correspondant à des courants ioniques ou à des courants synaptiques. Ici, on fait l'ajustement des paramètres biophysiques dans le cas le plus simple.), $\tau_L = \frac{C}{g}$, $J_L = \frac{I_L}{C}$ et $J(t) = \frac{I(t}{C}$ on obtient l'équation réduite suivante :

$$\frac{dV}{dt} = -\frac{V}{\tau_L} + J_L + J(t)$$

En réalisant une discrétisation de l'équation précédente avec un schéma d'Euler on obtient l'équation du modèle BMS :

$$V(t+dt) = V(t)[1 - \frac{dt}{\tau_L}] + J_L dt + J(t)dt$$

ou encore $V(t+dt) = \gamma V(t) + j_L + j(t)$ en posant $\gamma = 1 - \frac{dt}{\tau_L}$, $j_L = J_L dt$ et $j(t) = J(t)dt$. Cette équation est valable lorsque le potentiel est inférieur au seuile de décharge. Ici se pose une question importante. Ce seuil ne peut pas être fixé de manière arbitraire.Trop petit le neurone décharge tout le temps ; trop grand, il ne décharge jamais. Au contraire des autres paramètres (C, γ) qui sont des paramètres biophysiques connus pour les cellules ganglionnaires, θ dépend de la fréquence de décharge des neurones que l'on souhaite obtenir pour qu'elle soit conforme aux valeurs observées expérimentalement. On obtient ainsi une relation entre θ et la fréquence de décharge :

$$\theta = V_L + (1 - \gamma^{1/\mu})\frac{I}{\tau_L}$$

avec mu la fréquence de décharge moyenne de la cellule étudiée.

Une analyse d'ordre de grandeur nous permet de retrouver le système unitaire dans lequel nous allons travailler : le potentiel en mV, le courant en pA, le temps en ms, la conductance en nS et la capacité en pF.

Certaines conversions peuvent être effectuer directement dans le fichier XML de paramétrisation. Le parsing du fichier XML et l'interface graphique ont été adaptés par l'ingénieur informatique en charge de la maintenance du logiciel afin d'intégrer les nouveaux paramètres. C'est le cas pour la capacité qui est prise égale à 0.1 nF dans VirtualRetina, soit 100 pF dans notre nouveau système unitaire. En prenant $g_L = 5$nS (valeur des cellules X de chat), on peut calculer τ_L qui a donc une valeur de 20 ms. La valeur de γ doit être comprise entre 0 et 1. En choisissant la valeur de 0.9, on peut estimer la valeur de dt à 2 ms. Enfin, on peut calculer la valeur de θ si l'on connait la fréquence de décharge typique de la cellule.

La conversion du courant doit cependant se faire dans le code. On commence par tracer le courant en sortie de la couche des cellules bipolaires dans le cas d'un stimulus constant pour vérifier que celui-ci a le bon ordre de grandeur qui est d'environ 100 pA. (Figure 29)

Si l'on pose $j_t = j_L + j(t)$, $J_t = J_L + J(t)$ et $I_t = I_L + I(t)$ on obtient l'équation suivante : $j_t = \frac{Idt}{C}$. Il suffit donc de multiplier le courant à la sortie de la couche bipolaire par 2 ms et diviser par 100 nF.

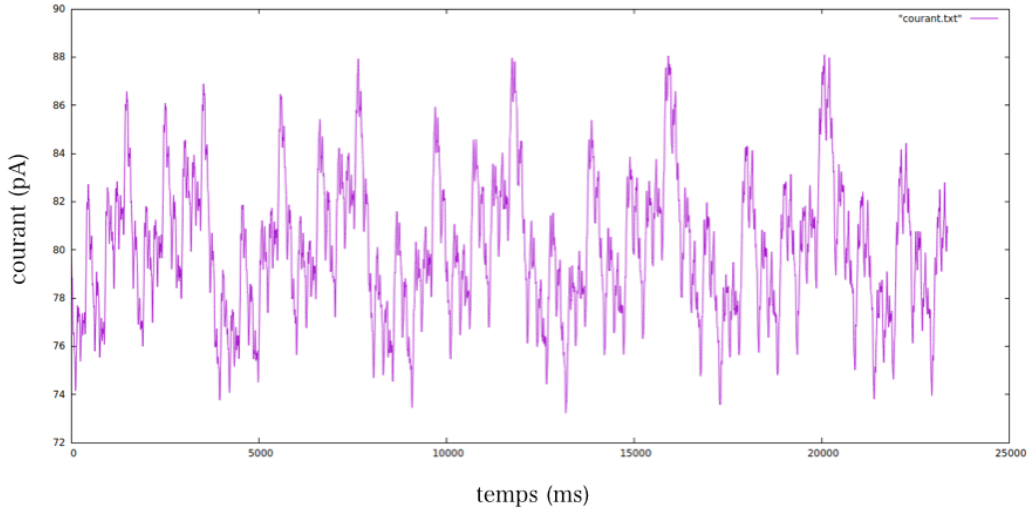

FIGURE 29 – Variation du courant en sortie de la couche des cellules bipolaires.

Vérification du modèle BMS

Après la conception d'Enas, plusieurs tests ont été effectués afin de valider le fonctionnement des différentes classes. Avant d'entamer les expériences, mon encadrant et moi avons décidé d'écrire un nouveau script de test dans le but est de vérifier que l'implémentation du modèle BMS permet de retrouver les fréquences de décharge théoriques en faisant varier la valeur de theta. (Cf Annexes)

En effet, l'équation $\theta = V_L + (1 - \gamma^{1/\mu})\frac{I}{\tau_L}$ peut également s'écrire sous la forme : $\mu = \frac{log(\gamma)}{log[1-\frac{g_L(\theta - V_L)}{I}]}$.

Le test consiste à générer des rasters d'une longueur de 2000 ms avec le modèle BMS en introduisant les nouvelles valeurs de paramètres (C, g_L ...) pour des valeurs de θ entre 0 et 40 mV (Dans Enas, l'origine des potentiels est translatée de sorte que le potentiel de repos (-70mV) est à 0. Cela donne une valeur maximale de -30 mV pour θ, qui correspond à la valeur biophysique) , calculer la fréquence de décharge correspondante puis la comparer à la valeur théorique. Les courbes obtenues nous permettent de valider le fonctionnement du BMS car les points expérimentaux et théoriques coïncident.

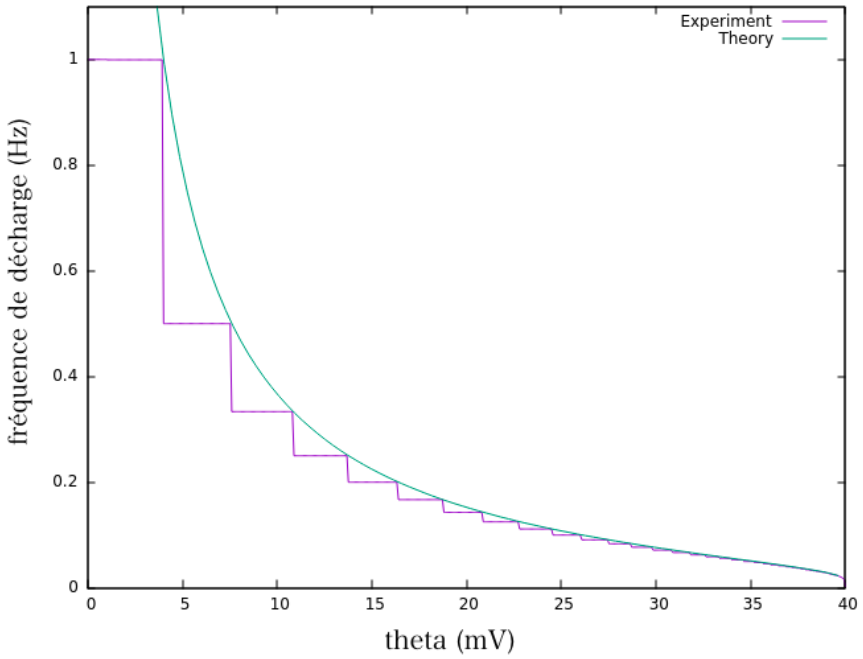

FIGURE 30 – Courbes théorique et expérimentale des fréquences de décharge en fonction du seuil.

Reproduction de l'expérience de Shapley & Victor

L'expérience de Shapley & Victor donne les premières mesures quantitatives du contrôle de gain dans la rétine. Elle a été appliquée à des cellules X de chat de type ON [12]. Le stimulus L(x,y,t) correspond à un réseau de luminance moyenne fixe $\tilde{L} = 20cd/m^2$, modulé en temps par une somme de sinusoïdes pondérée par des contrastes.

$$L(x, y, t) = \tilde{L}(1 + Gr(x, y) \sum_{i=1}^{8} c_i sin(f_i t))$$

avec Gr(x,y) la fonction du réseau sinusoïdal avec une amplitude normalisée. Les f_i correspondent à des fréquences qui varient sur une échelle logarithmique entre 0.2 Hz et 32 Hz. Enfin les c_i correspondent aux différentes valeurs de contraste.

L'expérience reproduite dans la thèse d'Adrien Wohrer [14] consiste à générer le stimulus de Shapley & Victor pour différente valeurs de contraste, lancer la simulation sur VirtualRetina en utilisant les paramètres correspon-

dant au type de cellule choisi, et faire enfin une analyse de Fourier de la fréquence de décharge obtenue.

Les résultats de cette expérience montrent que la réponse au stimulus est d'autant plus importante que le contraste est élevé. Les courbes d'amplitude et de phase révèlent une non-linéarité. Si le mécanisme était linéaire, la réponse des cellules aurait était proportionnelle à la valeur de c, les courbes d'amplitude auraient été parallèles espacées par log(2) lorsque le contraste est double, et les courbes de phase de superposeraient. Les courbes obtenues par Shapley & Victor montrent que l'espace entre les courbes d'amplitude augmente pour des valeurs de fréquences se situant entre 3.2 et 10 Hz, et une avance de phase pour les contrastes plus élevés. (Figure 31, 32)

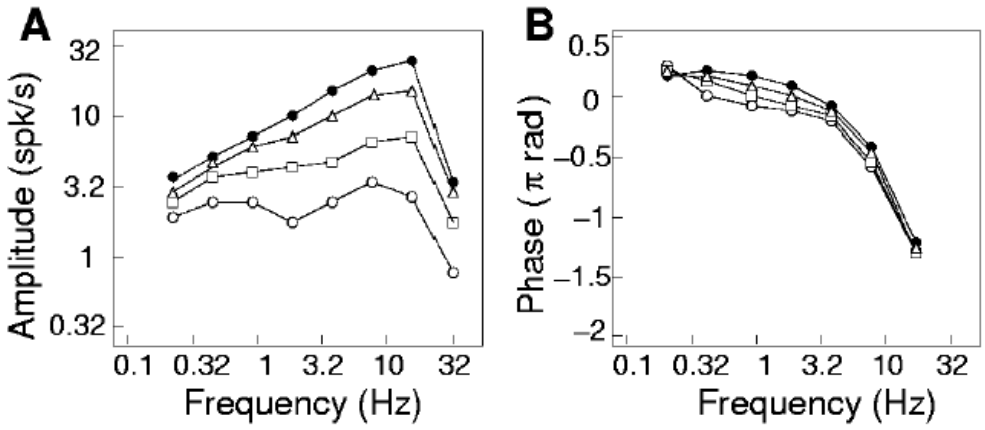

FIGURE 31 – Mécanisme de contrôle de gain dans une cellule ganglionnaire X de chat de type ON. Réponse au stimulus multi-sinus à différentes valeurs de contraste : cercle vide (0.0125), carré (0.025), triangle (0.05) et cercle plein (0.1).[12]

Le but est d'essayer de reproduire ces expériences avec VirtualRetina d'abord avec l'ancien contrôle de gain, et ensuite en implémentant le nouveau mécanisme utilisé par M. Berry. On commence donc par isoler la partie du code qui permet de générer le stimulus, en modifiant le format d'écriture de la sortie. (Cf Annexes)

En effet, le format Inrimage (.inr) original est inhérent à l'utilisation de la librairie CImg [12] dans le développement de VirtualRetina, toutes les lectures

12. Librairie qui définit des classes et des méthodes pour traiter les images dans un code C++. Elle permet de charger et d'enregistrer des images avec différents formats, d'appliquer des filtres, des transformations, de dessiner des primitives d'images, de faire

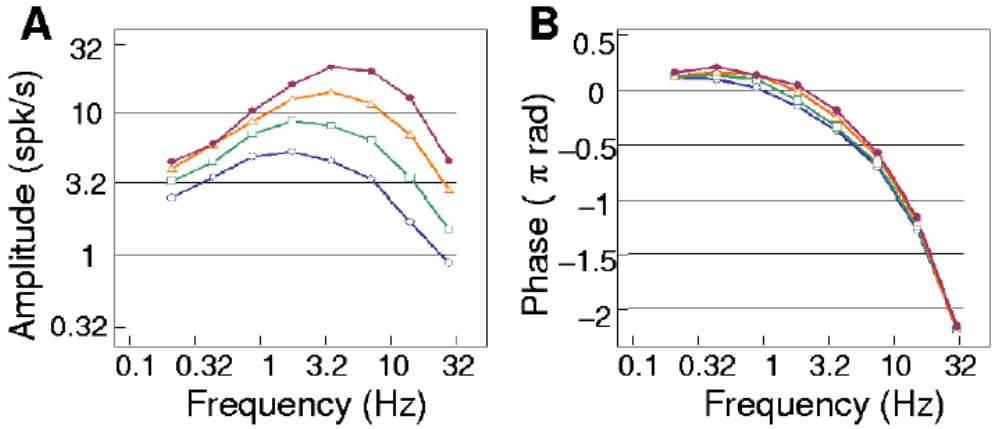

FIGURE 32 – Reproduction de l'expérience de Shapley & Victor par Adrien Wohrer. Les valeurs de contraste sont les mêmes.[14]

et écritures d'images se faisant grâce à cette librairie. Ce format est assez peu portable, et ne peut être converti en utilisant la simple commande convert, c'est pour cela qu'il a été nécéssaire d'effectuer ce changement dans le code en enregistrant l'image sous le format .png. Cela nécessite également une normalisation, mais à cause d'un problème de tone mapping [13] (le format .png peut uniquement coder des valeurs entre 0 et 255), nous ne sommes pas parvenus à reproduire le stimulus tel qu'enregistré sous le format Inrimage, qui n'impose pas de contraintes de gamme. En effet, un léger changement de la valeur du contraste du stimulus ne se retrouve pas dans dans les images png. Nous avons tout de même reproduit l'expérience, et nous obtenons des valeurs d'amplitudes qui correspondent globalement à l'ordre de grandeur de celles de l'expérience, mais la différence entre les courbes n'est significative qu'en prenant une grande valeur de contraste (0.9). (Figure 33, 34)

Nous reproduisons la même expérience en introduisant cette fois-ci le nouveau mécanisme contrôle de gain. Nous changeons pour cela le code du filtre non linéaire implémenté par VirtualRetina pour simuler le mécanisme. (Cf Annexe)

---

des calculs statistiques ...

13. Le tone mapping, appelé aussi mise en correspondance tonale, est une technique utilisée dans le traitement de l'image pour mettre en correspondance une palette de couleurs avec une autre, dans le but de convertir une image de grande gamme dynamique vers une image de dynamique plus restreinte.

Les résultats montrent que l'amplitude de la réponse est supérieure pour un contraste de 0.9, sauf pour deux valeurs de fréquences. Cela peut être dû à une sur-adaptation de la réponse par le modèle qui a pour effet d'induire des composantes plus faibles du signal pour un contraste plus élevé.

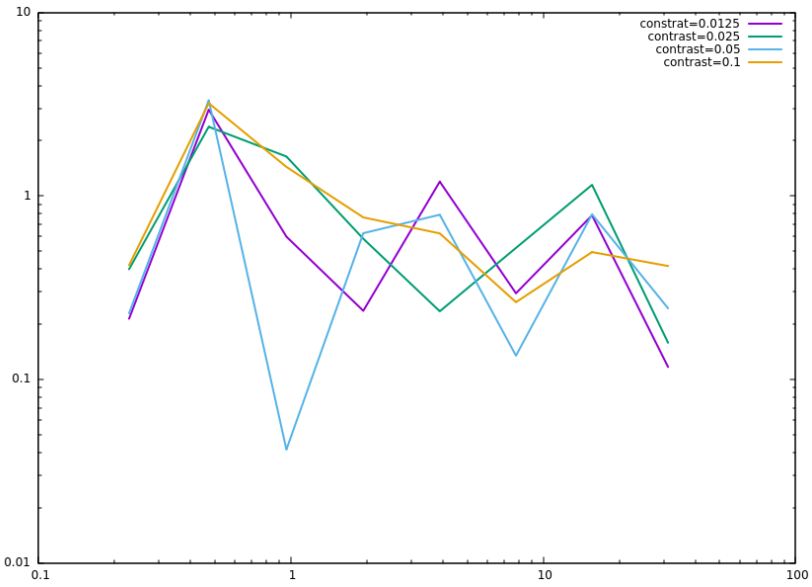

FIGURE 33 – Reproduction de l'expérience de Shapley & Victor avec Enas pour les mêmes valeurs de contraste. L'expérience ne donne pas le résultat escompté car le stimulus n'est pas correctement produit, Enas ne prenant pas en entrée le format Inrimage.

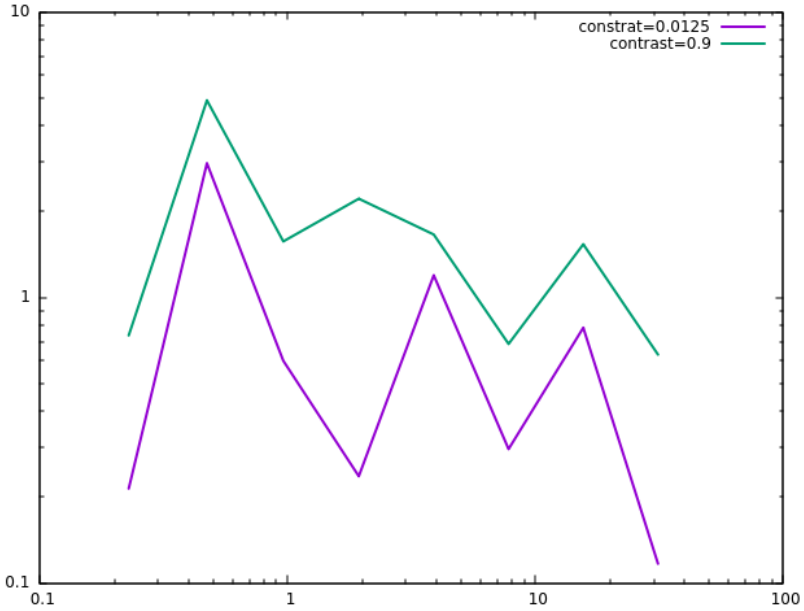

FIGURE 34 – Reproduction de l'expérience de Shapley & Victor avec Enas pour deux valeurs de contrastes, 0.0125, et 0.9. En raison du problème de tone mapping, la valeur 0.9 n'est pas la valeur réelle de contraste.

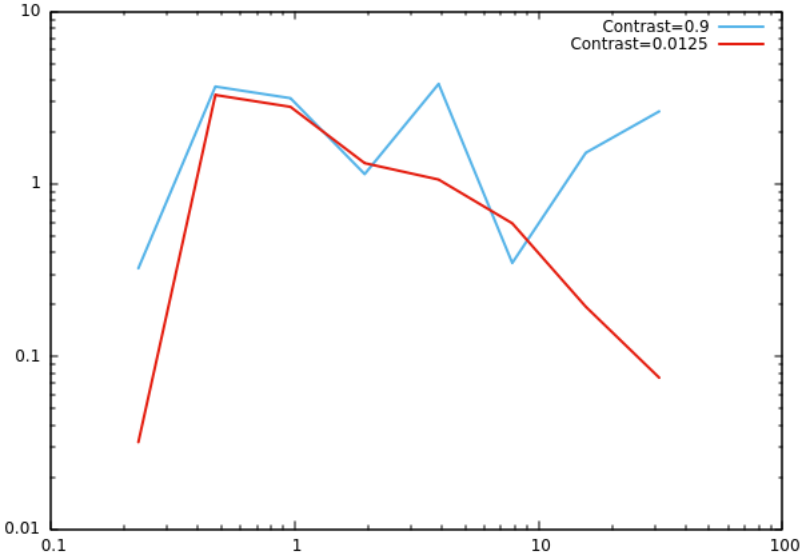

FIGURE 35 – Reproduction de l'expérience de Shapley & Victor avec Enas en intégrant le nouveau mécanisme de contrôle de gain.

3.2.7 Reproduction de l'expérience de M. Berry & al. avec le nouveau mécanisme de contrôle de gain

Après avoir reproduit le stimulus de l'expérience, caractérisé les champs récepteurs des cellules utilisées, simulé le mécanisme de contôle de gain de M. Berry & al. et l'avoir introduit dans Enas, nous pouvons maintenant reproduire l'expérience en utilisant les conversions déjà établies et les valeurs d'ajustement des champs récepteurs spatio-temporels pour paramétrer Enas. Nous avons également reproduit l'expérience avec l'ancien mécanisme de contôle de gain. Les résultats que nous obtenons montrent qu'il y a, dans les deux cas, anticipation du mouvement, les pic d'activité survenant au même moment. Pour rappel, nous ne sommes pas parvenus à reproduire l'expérience avec le web service, peut-être à cause d'une mauvaise paramétrisation du logiciel. Dans le cas du nouveau mécanisme de contrôle, la fréquence de décharge maximale ne dépasse pas 80 Hz, ce qui est cohérent avec les résultat de M. Berry & al. où la valeur maximale est entre 50 et 75 Hz. L'ancien mécanisme de contrôle de gain produit un pic plus élevé, ce qui montre que le nouveau induit une meilleure adaptation. Le nouveau mécanisme semble également baisser le niveau de l'activité spontanée de la cellule (inférieure à 20 Hz), valeur plus plausible biologiquement.

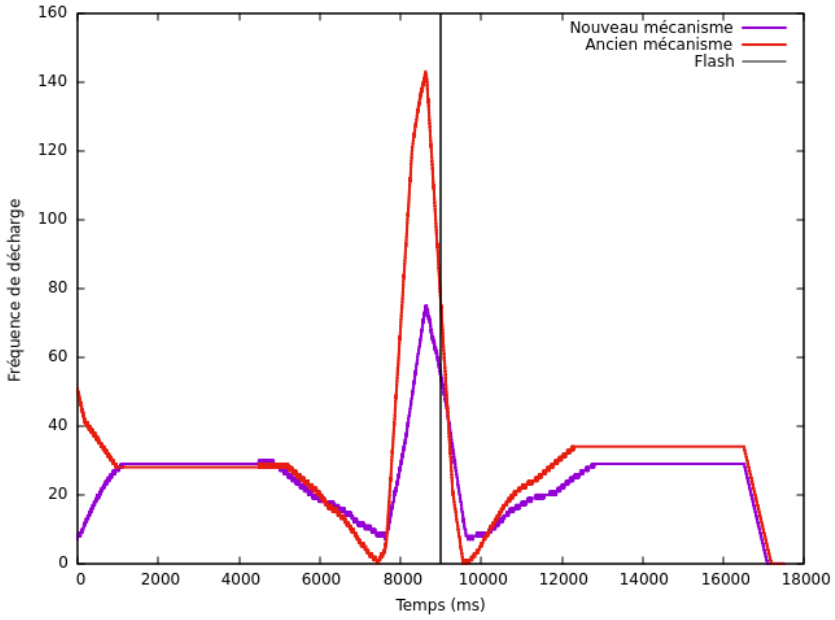

FIGURE 36 – Reproduction de l'expérience de Shapley & Victor avec Enas en intégrant le nouveau mécanisme de contrôle de gain.

3.3 Participation à l'école d'été CNS sur les neurosciences computationnelles à Goettingen

J'ai été admise à participer à l'école d'été CNS, le thème de cette année étant les approches mathématiques à la dynamique des systèmes neuronaux. L'interaction de milliers de cellules nerveuses représente, traite et stocke les informations dans le cortex. Plusieurs techniques ont été développées récemment afin d'observer et d'analyser l'activité des populations de neurones. Le formation proposée par l'école est axée sur les approches mathématiques mais traite également de l'interaction entre la théorie et l'expérimentation en neurosciences.
Durant deux semaines, l'école propose des cours intensifs en blocs thématiques dédiés à une approche particulière à la dynamique des circuits de neurones, pour ensuite travailler en groupe sur des problèmes relatifs aux thèmes abordés, pour une compréhsension approfondie des techniques mathématiques présentées. Les groupes doivent ensuite présenter et discuter leurs résultats. Enfin, le soir, un cours tourné plus vers l'expérimentation est donné.
Dans ce qui suit, nous allons rapidement présenter les cours vus au cours de l'école et présenter quelques résultats obtenus lors de la résolution des problèmes.

3.3.1 Variations autour du bruit dans les modèles de neurones par Magnus Richardson, Warwick University

La première partie du cours a abordé l'étude du bruit principalement dans le modèle Leaky Integrate and Fire. Un bruit de type "shot noise" donne une série d'exponentielles qui se superposent et n'est pas évident à traiter. Il est plus intéressant de le transformer en bruit Gaussien qui lui est plus facile à manipuler. Des calculs de moyenne et de variance permettent d'établir l'équivalence. Lorsque le bruit est faible, on obtient une limite quasi-déterministe de l'état permanent. Lorsqu'on module la moyenne ou la variance des fluctuations synaptiques (bruit), on obtient une réponse qui dépend de ces modulations. Dans le cas où la moyenne est modulées, on peut remarquer que l'évolution de la fréquence de décharge est d'abord rapide puis devient linéaire. (Cf Annexe) (Figure 37)

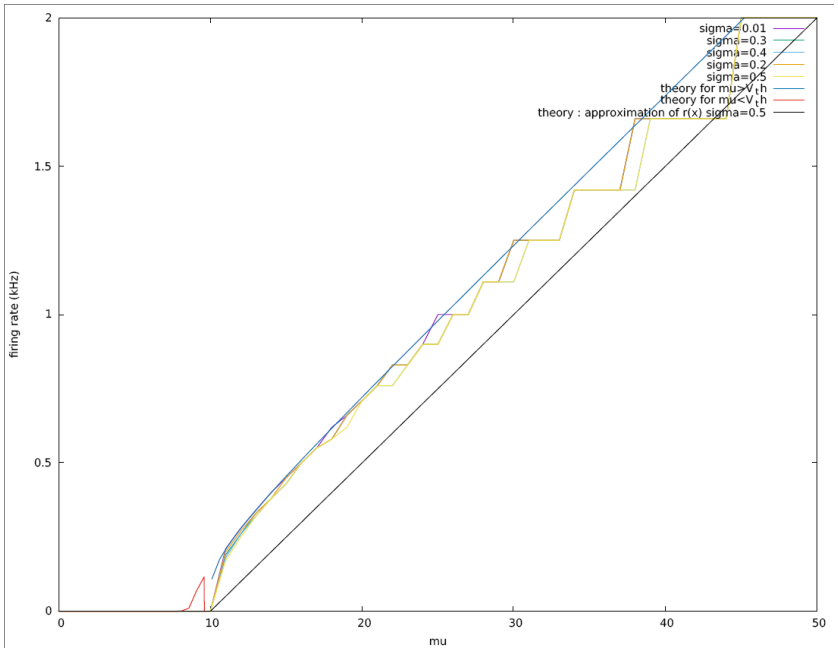

FIGURE 37 – Simulation du modèle Leaky Integrate and Fire entraîné par du bruit Gaussien, modulé en moyenne pour différentes valeurs de variance.

Une étude plus exhaustive a ensuite été faite du "shot noise", qui a permis de dériver l'équation principale de ce type de bruit, dont la transformée de Laplace est similaire à l'équation de Fokker-Planck, équation aux dérivées partielles linéaire qui concerne un processus de Markov. Cette équation permet, dans des cas particuliers, d'étudier le bruit coloré.
Dans la troisième partie du cours, d'autres modèles dérivés du modèle Integrate and Fire ont été présentés, à l'exemple du modèle quadratique et du modèle exponentiel. Enfin, la dernière partie a introduit les modèles de deux variables.

3.3.2 Mécanique statistique des réseaux de neurones par Helias Moritz, Institute of Neuroscience and Medicine, Julich

Ce cours a présenté les points fondamentaux de l'utilisation des méthodes issues de la mécanique statistique des systèmes classiques pour théoriser les réseaux de neurones. Dans la première partie, les notions fondamentales de probabilité ont été introduites, les moments, cumulants et leur relation par le théorème de Wick. Ensuite, la formulation graphique de la théorie des perturbtions à l'aide des diagrammes de Feynman a été abordée.

Dans la deuxième parties, des outils tels que "l'action efficace", "les fonctions de vertex" ont été présentés afin de dépasser l'approximation du champ moyen. Enfin, les concepts des systèmes désordonnés ont été utilisés pour étudier les réseaux avec une connectivité aléatoire, déduire leur théorie du champs moyen et expliquer les statistiques des fluctuations de ces réseaux.

3.3.3 La mémoire à court et à long terme par Misha Tsodyks, Weizmann Institute of Science

La première partie de ce cours a traité de la plasticité synaptique et de l'approche phénoménologique de la transmission synaptique. Il existe deux équations différentielles fondamentales, la première rend compte de la facilitation et la deuxième de la dépression. L'interaction des deux permet de retrouver le potentiel excitatoire postsynaptic. L'étude de bifurcation du système réduit au cours des séances de travail permet t'établir, pour certaines valeurs de paramètres, la convergence vers un cycle limite. Ce résultat a été démontré analytiquement puis simulé. (Cf Annexe) (Figure 38)

Dans la deuxième partie, le concept de la working memory a été introduit, concept qui fait référence à la partie de la mémoire qui stocke des informations temporaires nécessaires au planning et à l'exécution d'une tâche. Ce modèle de mémoire repose sur la facilitation synaptique.

La dernière partie a abordé le rappel de l'information à partir de la mémoire à long terme. Une expérience reproduite par le professeur stipule que sur une liste de 16 mots énoncés avec 3 s d'intervalle, le sujet peut se rappeler uniquement de 8 mots en moyenne. La modélisation de ce phénomène repose sur deux mécanisme : le codage et le rappel de l'information. Le codage se fait avec un patterne de mémoire aléatoire auquel on peut affecter une probabilité, et qui joue un rôle dans la dynamique globale du réseau. La capacité d'enregistrement est uniquement fonction de cette probabilité et de la taille du réseau. Le sujet, lorsqu'il subit l'expérience plusieurs fois, présente une capacité d'apprentissage. Dans la modélisation, ce mécanisme est modélisé par le "Hebbian learning".

L'expérience a également montré qu'il existe certains sujets exceptionnels qui peuvent se rappeler des 16 mots. En étudiant l'ordre de rappel, deux types ont été identifiés. Les sujets qui mémorisent les mots dans l'ordre exact, et ceux qui les mémorisent par blocs de 3 ou 4 dans l'ordre inverse. L'étude de cette observation n'a pas encore été formalisée.

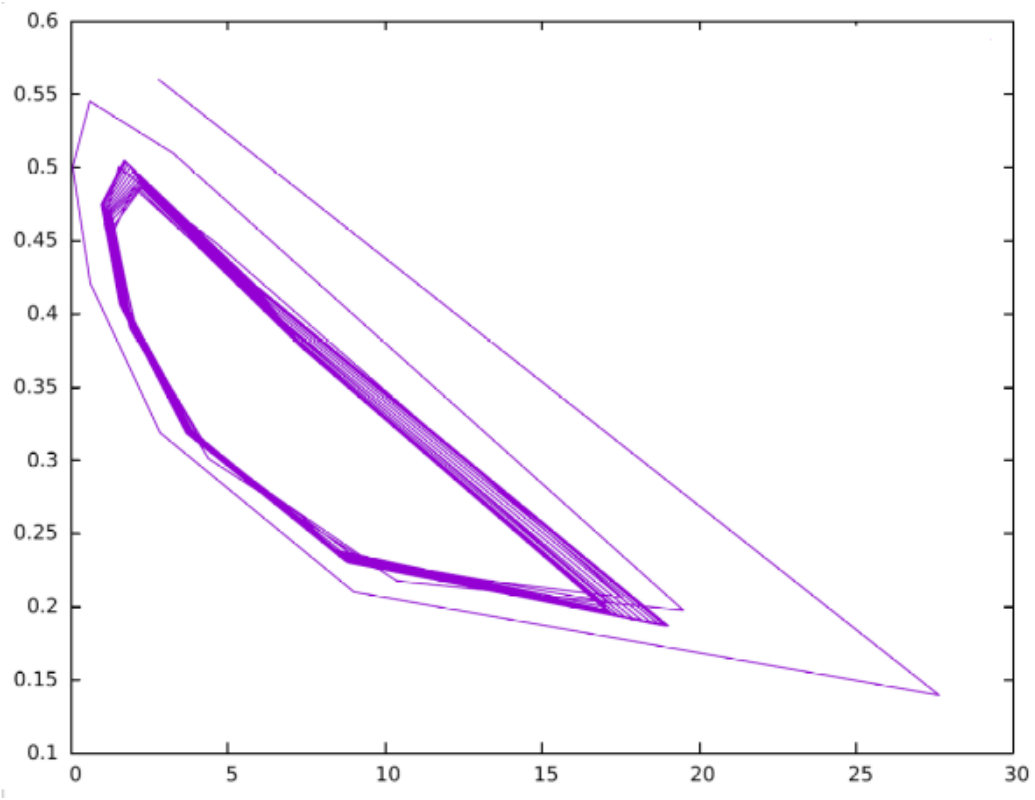

FIGURE 38 – Convergence du circuit récurrent homogène aux équations réduites vers un cycle limite.

3.3.4 Des cartes dans le cerveau par Vijay Balasubramanian, University of Pennsylvania

Le cerveau possède une diversité d'unités fonctionnelles dont l'activité coordonnée produit la fonction globale souhaitée. Après une présentation des calculs fondamentaux impliqués dans la biophysique, le cours s'est intéressé à la rétine et aux différents types de cellules qui la constituent. Il exisite notamment 10 types de cellules bipolaires, 30 interneurones et 20 cellules ganglionnaires. Chacune de ces cellules a une fonction spécifique : détection de contours, sensibilité à une orientation, une couleur ... L'efficacité énergétique du cerveau en général est en partie assurée grâce à la spécialisation des cellules. Les cellules ne répondent qu'au type de stimulus qui "les intéresse". En particulier, dans la rétine, le principe de codage efficace explique l'organisation spatiale de mosaïque rétinienne. A titre d'exemple, les proportions des cellules ON et OFF représent une balance parfaite (les cellules OFF sont 1.3 à 2 fois plus nombreuses que les cellules ON), car il existe plus de parties sombres dans les scènes naturelles, et l'information y est généralement plus dense.

La deuxième partie du cours s'est intéressée à la théorie de l'information visant à quantifier et qualifier le contenu en information d'un ensemble de données. Des mesures telles que la quantité d'information, l'entropie, et la capacité du système ont été manipulées, mais l'application de cette théorie aux neurosciences reste controversée car ne tenant pas compte de la réalité biologique. Il est notamment difficile d'appliquer cette théorie au système olfactif où la perception des odeurs est difficilement quantifiable.

Conclusion

L'étude du mécanisme d'anticipation du mouvement tel que décrit par M. Berry et la modification du code de VirtualRetina en conséquence nous ont permis de reproduire cette anticipation dans le cas d'une trajectoire simple. Pour ce type de trajectoire, le mécanisme de contrôle de gain joue un rôle prépondérant. Pour les trajectoires plus complexes, le contrôle de gain ne suffit plus et il faut introduire la connectivité latérale assurée par les cellules amacrines. Nous n'avons malheureusement pas eu le temps d'aborder cette thématique au cours du stage, et nous l'entamerons dès le début de la thèse. Il faudra notamment réfléchir, en collaboration avec nos partenaires expérimentalistes, à une forme complexe de trajectoire qui permettrait de mettre en évidence le mécanisme d'anticipation selon des critères bien définis. Il s'agira ensuite d'utiliser les différents types de connectivité intégrés dans Enas afin de reproduire l'expérience et de définir le type qui se rapproche le plus de la réalité biologique.

La rétine ne constitue qu'une étape préliminaire du traitement visuel. Il faudra également axer le travail autour de la région fonctionnelle V1 du cortex visuel, impliquée dans la détection du contraste. Si le temps le permet, on pourra également élargir l'étude à l'aire V3 qui est aussi hautement sensible au contraste. Le but sera de développer un logiciel qui puisse simuler correctement une partie du système visuel, en particulier le mécanisme d'anticipation du mouvement.

Ce stage a été très enrichissant dans la mesure où il m'a permis de me familiariser avec le monde de la recherche, par la nature même du travail mais également par la participation à la conférence ICMNS (International Conference on Mathematical Neuroscience) organisée par l'Inria, à différents séminaires et à l'école d'été en neurosciences mathématiques.

Je tiens à remercier encore une fois mon encadrant, B. Cessac, pour m'avoir offert la possibilité de faire une thèse doctorale sous sa direction. Je suis convaincue que le sujet proposé présente un grand intérêt pour une meilleure compréhension du système visuel dans sa globalité.

Références

[1] M.J. Berry, I.H. Brivanlou, T.A. Jordan, and M. Meister. Anticipation of moving stimuli by the retina. *Nature*, 398(6725) :334–338, 1999.

[2] Bruno Cessac. A discrete time neural network model with spiking neurons. rigorous results on the spontaneous dynamics. *J. Math. Biol.*, 56(3) :311–345, 2008.

[3] Eric Y. Chen Olivier Marre Clark Fisher Greg Schwartz Joshua Levy Rava Azeredo da Silviera and Michael J. Berr. Alert Response to Motion Onset in the retina. *The Journal of Neuroscience*, (120) :120–132, January 2013.

[4] Peter Dayan and L. F. Abbott. *Computational and Mathematical Modeling of Neural Systems*. The MIT Presss, 2005.

[5] G.T. EINEVOLL and P. HEGGELUND. Mathematical models for the spatial receptive-field organization of nonlagged X-cells in dorsal lateral geniculate nucleus of cat. *Visual Neuroscience*, June 2000.

[6] Matteo Carandini Jonathan B. Demb Valerio Mante David J. Tolhurst Yang Dan Bruno A. Olshausen Jack L. Gallant and Nicole C. Rus. Computations in the early visual cortex. *The Journal of Neuroscience*, page 10577 ?10597, November 2005.

[7] Professor David Heeger. Poisson Model of Spike Generation. September 2000.

[8] Werner M. Kistker and Wulfram Gestner. *Spiking Neuron Models : Single Neurons, Populations Plasticity*. Cambridge University Press, 2002.

[9] Tony Lindeberg. A computational theory of visual receptive fields. November 2013.

[10] O. Marre, D. Amodei, N. Deshmukh, K. Sadeghi, F. Soo, T.E Holy, and M.J. Berry II. Mapping a Complete Neural Population in the Retina. *The Journal of Neuroscience*, 43(32) :14859–14873, October 2012.

[11] Valerie Ventura Robert E Kass and Can Cai. Statistical smoothing of neuronal data. January 2003.

[12] R. M. Shapley and J. D. Victor. The effect of contrast on the transfer properties of cat retinal ganglion cells. *The Journal of Physiology*, 285 :275–298,, 1978.

[13] Shigeru Shinomoto. *Estimating the Firing Rate*. Springer, 2010.

[14] Adrien Wohrer. *Model and large-scale simulator of a biological retina, with contrast gain control.* PhD thesis, University of Nice-Sophia Antipolis, 2008.

[15] Adrien Wohrer and Pierre Kornprobst. Virtual retina : A biological retina model and simulator, with contrast gain control. *Journal of Computational Neuroscience*, 26(2) :219, 2009. DOI 10.1007/s10827-008-0108-4.

Annexes

1. Code Java pour générer la séquence d'images

```
package barre_translation;

import java.awt.Color;

public class panneau extends JPanel {

    private static final long serialVersionUID = 1L;
    private int posX = -5;
    private int posY = 25;

    public void paintComponent(Graphics g){
    g.setColor(Color.black);
    g.fillRect(posX, posY, 5, 15);
    this.setSize(60,60);
  }

  public int getPosX() {
    return posX;
  }

  public void setPosX(int posX) {
    this.posX = posX;
  }

  public int getPosY() {
    return posY;
  }

  public void setPosY(int posY) {
    this.posY = posY;
  }
}
```

```java
package barre_translation;
import java.awt.AWTException;

@SuppressWarnings("unused")
public class Fenetre extends JFrame{

    public static void main(String[] args) throws IOException, AWTException, InterruptedException {
            new Fenetre();
          }
    private static final long serialVersionUID = 1L;
    private panneau pan = new panneau();


    public Fenetre() throws IOException, AWTException, InterruptedException{
    this.setTitle("Animation");
    this.setSize(90,90);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setLocationRelativeTo(null);
    this.setContentPane(pan);
    this.setVisible(true);
    this.setVisible(true);
    go();

  }

  private void go() throws AWTException, IOException, InterruptedException{

  for(int i = 0; i < 180; i++){
      int x = pan.getPosX(), y = pan.getPosY();
      if (i>=31 && i%2!=0) x++;
      pan.setPosX(x);
      pan.setPosY(y);
      long start = System.nanoTime();
      pan.revalidate();
      pan.repaint();
      long end = System.nanoTime();
      System.out.println(end - start);
      Thread.sleep(100);
      BufferedImage image = ScreenImage.createImage(pan);
      ScreenImage.writeImage(image, "panel"+i+".png");
      try {
        Thread.sleep(10);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
}
```

```java
package barre_translation;

import java.awt.*;

public class ScreenImage
{
    private static List<String> types = Arrays.asList( ImageIO.getWriterFileSuffixes() );
    public static BufferedImage createImage(JComponent component)
    {
        Dimension d = component.getSize();
        if (d.width == 0 || d.height == 0)
        {
            d = component.getPreferredSize();
            component.setSize( d );
        }
        Rectangle region = new Rectangle(0, 0, d.width, d.height);
        return ScreenImage.createImage(component, region);
    }

    public static BufferedImage createImage(JComponent component, Rectangle region)
    {
        if (! component.isDisplayable())
        {
            Dimension d = component.getSize();
            if (d.width == 0 || d.height == 0)
            {
                d = component.getPreferredSize();
                component.setSize( d );
            }
            layoutComponent( component );
        }
        BufferedImage image = new BufferedImage(region.width, region.height, BufferedImage.TYPE_INT_RGB);
        Graphics2D g2d = image.createGraphics();
        if ( component.isOpaque())
        {
            g2d.setColor( component.getBackground() );
            g2d.fillRect(region.x, region.y, region.width, region.height);
        }
        g2d.translate(-region.x, -region.y);
        component.paint( g2d );
        g2d.dispose();
        return image;
    }
}
```

2. Programme C de calcul des fréquences de décharge

```c
#include "spiking.h"
#include <stdlib.h>
#include <stdio.h>

#define largeur_fenetre 0.1

typedef struct Element Element;
struct Element
{
    double nombre;
    Element *suivant;
};

typedef struct File File;
struct File
{
    Element *premier;
};

File *initialiser()
{
    File *file = malloc(sizeof(*file));
    file->premier = NULL;

    return file;
}

void ajouter(File *file, double spike)
{
    Element *nouveau = malloc(sizeof(*nouveau));
    if (file == NULL || nouveau == NULL)
    {
        exit(EXIT_FAILURE);
    }

    nouveau->nombre = spike;
    nouveau->suivant = NULL;

    if (file->premier != NULL)
    {
        Element *elementActuel = file->premier;
        while (elementActuel->suivant != NULL)
        {
            elementActuel = elementActuel->suivant;
        }
        elementActuel->suivant = nouveau;
    }
    else
    {
        file->premier = nouveau;
    }
}
```

```c
int main(int argc, char *argv[])
{
    FILE* fichier = NULL;
    fichier = fopen("spiking_seul.txt", "w+");

    FILE* input = NULL;
    input = fopen("flash_seul.txt", "r+");
    double i;
    int spike_nb=0;
    double ref;
    double firing_rate=0;
    double spike_time;
    int curseur=0;
    double min = 0;
    double max = 0;
    File *file = initialiser();

    if ( input != NULL )
    {
        int nb_lignes = 0;
        while(fscanf(input, "%lg", &spike_time ) != EOF)
        {
            nb_lignes++;
            ajouter(file, spike_time);
            if (min==0) min = spike_time;
            max = spike_time;
        }
        printf("%f\n",min);
        printf("%f\n",max);
        fseek(input, 0 , SEEK_SET);
        Element *element = file->premier;

        for(i=min-largeur_fenetre;i<max;i+=0.001)
        {
            if (element != NULL) {
                ref=element->nombre;

            }
            spike_nb=0;
            while((fscanf(input, "%lg", &spike_time ) != EOF) && (spike_time - i < largeur_fenetre))
            {
                spike_nb ++;
            }
            if(max-i>=1.0)fseek(input, -sizeof(double), SEEK_CUR);
            if ((i<min || i>=ref+0.002)) firing_rate += spike_nb/largeur_fenetre*1000;
            else firing_rate += (spike_nb-1)/largeur_fenetre*1000;
            if (firing_rate < 0) firing_rate = 0;
            fprintf(fichier,"%f\n",firing_rate);
            if (element != NULL && (element->suivant)->nombre < i ) element = element->suivant;

        }

            fclose ( input );
    }
    else
        {
            perror ( input );
        }

    fclose ( fichier );
    return 0;
}
```

3. Programme C de génération du processus de Poisson inhomogène

```c
#include "random_test.h"
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <math.h>

#define delta_t 0.0001
#define T 1.0

int main(int argc, char *argv[])
{
    double t ;
    double f=3;
    double r ;
    int i;
    char buf[20];

    srand((unsigned)time(NULL));

        for (i=0; i<10; i++)
        {
            FILE* output1 = NULL;
            sprintf(buf, "firing_output_%d.txt", i);
            output1 = fopen(buf, "w+");
            FILE* output2 = NULL;
            sprintf(buf, "random_spikes_%d.txt", i);
            output2 = fopen(buf, "w+");

            for (t=0; t<=T; t+=delta_t) {
                r = rand()*1.0/RAND_MAX;
                f=3+sin(t*10);
                if ((f*1000*delta_t) >= r)
                {
                    printf("%f\n",f*1000*delta_t);
                    fprintf(output2,"%f\n",t);
                }
        }
    }
}
```

4. Programme C de calcul de la variance sans biais

```c
#include "random_test.h"
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <math.h>

#define delta_t 0.0001
#define T 1.0
#define pas 0.001

int main(int argc, char *argv[])
{
    double t ;
    double f=3000 ;
    double r ;
    int i;
    char buf[20];
    int* y=calloc(5000,sizeof(int));
    double somme=0.0;
    double moyenne=0.0;
    double S_n=0.0;
    double erreur;
    double log_erreur;

    srand((unsigned)time(NULL));

        for (i=0; i<5000; i++)
        {

            for (t=0; t<=T; t+=delta_t) {
                r = rand()*1.0/RAND_MAX;
                if (f*delta_t >= r)
                y[i]++;

                }
            somme+=pow(y[i],2);
            moyenne+=y[i];
        }

    moyenne = moyenne/5000.0;
    printf("%lg\n",moyenne);
    S_n=(somme/4999.0)-(5000.0*pow(moyenne,2)/4999.0);
    erreur= sqrt(S_n)/sqrt(5000.0);
    log_erreur= log(erreur);
    printf("%lg\n",erreur);
    printf("%lg\n",log_erreur);
}
```

5. Fonction Matlab d'ajustement des sections temporelles

```
function [fitresult, gof] = createFit(X, Y, Z)


maxi=0;
maxj=0;
max=0;
for i=1:40
    for j=1:40
     if (abs(Z(i,j,1))>max) maxi=j;
         maxj=i;
         max=abs(Z(i,j,1));
     end
    end
end

Z=medfilt2(Z);
[xData, yData, zData] = prepareSurfaceData( X, Y, Z );

% Set up fittype and options.
ft = fittype( 'a/(pi*c^2)*exp(-(((x-b)^2/2*c^2)+(y-d)^2/2*c^2))-e/(pi*...
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.Lower = [0 maxi-1 0 maxj-1 0 1];
opts.MaxFunEvals = 6000;
opts.MaxIter = 10000;
opts.StartPoint = [max maxi 1 maxj max 1.5];
opts.Upper = [Inf maxi+1 20 maxj+1 Inf 5];

% Fit model to data.
[fitresult, gof] = fit( [xData, yData], zData, ft, opts );
 %Plot fit with data.
figure( 'Name', 'fit' );

h = plot( fitresult, [xData, yData], zData );
legend( h, 'fit', 'Z vs. X, Y', 'Location', 'NorthEast' );
%Label axes
xlabel X
ylabel Y
zlabel Z
grid on
view( 92.5, 14.0 );
```

6. Programme C de simulation du mécanisme de contrôle de gain

```c
#include <stdio.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_odeiv2.h>
#include <gsl/gsl_integration.h>
#include <math.h>
#include <string.h>


#define alpha 85.0
#define theta 0


struct param_type { // paramètre de l'équa diff de V
    double tau;
    double K;
    double B;
};

struct param_spa {  //paramètres du profil spatial
    double sigma1;
    double sigma2;
    double mu;
    double a;
    double g;
};

struct param_temp {  //paramètres du profil spatial
    double tau;
    double t;
    double n;
};

unsigned long factorial(unsigned long j)
{
    if ( j == 0 )
        return 1;
    return(j * factorial(j - 1));
}

int func (double t, const double y[], double f[],void *params) // Equation différentielle de V(t)
{
  (void)(t);
  struct param_type *my_params_pointer = (param_type *) params;
  double tau = my_params_pointer->tau;
  double K = my_params_pointer->K;
  double B = my_params_pointer->B;
  f[0] = y[1];
  f[0] = -y[0]/tau+K*B/(1+pow(y[0],4));
  return GSL_SUCCESS;
}
```

```
double k_temp (double tp, void * params) { //partie temporelle du noyau, fonction gamma
  struct param_temp *my_params_pointer = (param_temp *) params;
  double t=my_params_pointer->t;
  double tau=my_params_pointer->tau;
  double n=my_params_pointer->n;
  double f =  pow(n*(t-tp),n)*exp(-n*(t-tp)/tau)/(factorial(n-1)*pow(tau,n+1)); // à vérifier
  return f;
}


double stimulus (double x, double a, double b) { //stimulus, fonction porte
  double g;
  if ((a-50 <= x) && (x <= a+50)) g = b ;
  else g = 0.0;
  return g;
}

double k_spa (double x, void * params) {  //partie spatiale du noyau, DOG
  struct param_spa *my_params_pointer = (param_spa *) params;
  double mu = my_params_pointer->mu;
  double sigma1 = my_params_pointer->sigma1;
  double sigma2 = my_params_pointer->sigma2;
  double a = my_params_pointer->a;
  double g = my_params_pointer->g;
  double h = stimulus(x,a,g)*(exp(-pow((x-mu),2)/(2.0*pow(sigma1,2)))/(sigma1*sqrt(2.0*M_PI))-exp(-pow((x-
mu),2)/(2.0*pow(sigma2,2)))/(sigma2*sqrt(2.0*M_PI)));
  //printf("stim(%f,%f) =%f\n", x,a, stimulus(x,a));
  //printf("f(%f) =%f\n",x,h);
  return h;
}

int main (void)
{
  for (double j=0;j<=0;j++)
{
  struct param_spa params_s = {88,150,500.0,-1.0,5.0};
  struct param_temp params_t = {170.0,0.0,3}; //{tau, t, n}
  struct param_type my_params = {170.0,1,0.045}; // {tau, K, B}
  char str[15];
  sprintf(str, "neurone%d", (int)j);
  FILE * meca = fopen(str,"w+");
  FILE * ligne = fopen("ligne","w+");

  double V_ini=0;

  gsl_integration_workspace * w = gsl_integration_workspace_alloc (1000000);

  double result, error;
  double u;
  double g;
  double borne = 10000;
```

```
double temp;

double V = V_ini;
double r;

gsl_function F;
F.function = &k_temp;

//gsl_function L;
//L.function = &stimulus;

gsl_function G;
G.function = &k_spa;
G.params = &params_s;

gsl_odeiv2_system sys = {func, NULL , 2, &my_params};
gsl_odeiv2_driver * d =
  gsl_odeiv2_driver_alloc_y_new (&sys, gsl_odeiv2_step_rk8pd,
                                 1e-6, 1e-6, 0.0);
double dt=0.01;
double t = 0.0 ;
double y[2] = { 1.0, 0.0 };

//gsl_integration_qags (&G, 0, 40, 1e-11, 1e-7, 10000, w, &result, &error);

while (t<=10000)
      {
      fprintf (meca,"% .5f\t", params_s.mu-params_s.a-0.5);
      L.params = &t;
      gsl_integration_qags (&L, 0, 40, 1e-11, 1e-7, 1000, w, &result, &error);
      printf("result =% .8f\n", result);
      fprintf (meca,"% .5f\t", t);
      //gsl_integration_qags (&G, (double) j-5, (double) j+5, 1e-7, 1e-7, 1000, w, &result, &error);
      gsl_integration_qags (&G, 0, 10, 1e-7, 1e-7, 1000, w, &result, &error);
      if (t==300) params_s.a=0.0;
      if (t>300)params_s.a=t/10.0;

      G.params = &params_s;
      temp=result;
      //printf("temp =% .8f\n", temp);
      //printf("error =% .8f\n", error);
      params_t.t = t;
      F.params = &params_t;
      gsl_integration_qagil (&F, t, 0, 1e-7, 1000, w, &result, &error);

      if (V<0) g=1 ;
      else g=(double) 1/(1+pow(V,4));

      fprintf (meca,"% .5f\t", g);
```

```
        u = g*temp*result;
        //fprintf (meca,"% .5f\t", u);

        if (u>theta) r=alpha*(u-theta);
        else r=0;
        fprintf (meca,"% .5f\n", r);

        my_params.K=temp*result;
        gsl_odeiv2_system sys = {func, NULL , 2, &my_params};

        int status = gsl_odeiv2_driver_apply_fixed_step (d, &t,dt, 1, y);

        if (status != GSL_SUCCESS)
         {
          printf ("error, return value=%d\n", status);
          break;
         }
        V=y[0];
        //fprintf (meca,"%.5f\n", V);


    }
    fclose(meca);

  for (int i =0; i<=500; i++) {
                                double e=0;
                                double f=120;
                                if (i==500) fprintf (ligne,"%f\n", f);
                                else fprintf (ligne,"%f\n", e);
                                }


  fclose(ligne);
  gsl_odeiv2_driver_free (d);
  gsl_integration_workspace_free (w);
  }
  return 0;

}
```

7. Programme C++ de test du modèle BMS

```
#include "EnaS.h"
using namespace enas;
#include <iostream>
using namespace std;
#include "BMSPotential.h"
#include <fstream>

int main()
{
        //int Z=1000;
        //std::vector<double> theta(Z);
        //std::vector<double> firing_rate(Z);
        double gL=5; //Leak conductance (nS)
        double C=100; //Membrane capacity (pF)
        double tauL=C/gL; //Leak characteristic time (ms)
        //printf("tauL=%lg ms\n",tauL);exit(0);
        const double leak = 0.9;
        double dt=(1-leak)*tauL;
        //printf("dt=%lg ms\n",dt);exit(0);

        BMSPotential bms;

        int N=1;//Number of neurons
        int T=1000*dt;//Duration (in ms)
        double Iext=200;//External current (pA)
        double Jext=Iext*(1-leak)/gL;// BMS External forcing (mV)
        //printf("Jext=%lg mV\n",Jext);exit(0);

        int i=0;
        std::vector<double> Ie(N);
        for (i=0;i<N;i++) Ie[i]=Jext;
        std::ofstream f1;
        f1.open ("/user/ssouihel/home/Desktop/firing.txt");
        //std::string mon_fichier = "/user/ssouihel/home/enas/gui/build/bin/courant.txt";
        //std::ifstream fichier(mon_fichier.c_str(), ios::in);

        const double sigmaB = 0.001;
        int Nstep=1000;
        double theta_min=0,theta_max=20*Jext,theta_step=(theta_max-theta_min)/(double)Nstep;
        double theta=theta_min;
        //double Imin=1,Imax=100,Istep=0.1;
        //double theta=1.0;
        //double I=Imin;
        while (gL*theta<Iext){//theta loop
        //if (fichier)
        //{
        //while (I<=Imax){//I loop
        //for (i=0;i<=48000;i++){
```

```
                //fichier >> I;
                bms=BMSPotential() ;
                //cout<<bms.unit_threshold<<endl;
                bms.reset(N, leak, sigmaB,C,theta);
                //Jext=I*(1-leak)/gL;
                //for (i=0;i<N;i++) Ie[i]=I;
                for (i=0;i<N;i++) Ie[i]=Jext;

                bms.setUnitCurrents(Ie);
                RasterBlock *raster = bms.getRasterBlock(0,T);
                //for (int i=0;i<1000;i++)
                //{
                  // if(raster->getEvent(0,i)) cout<<i<<endl;
                //}

                String path = "/user/ssouihel/home/Desktop/rasters";
                char chaine[100];
                sprintf(chaine,"Raster_BMS_test_firing_rate_theta%lg",theta);
                std::string str="/user/ssouihel/home/Desktop/rasters/";
                str+=chaine;
                //raster->save( str , "unit-time", T);
                raster->load( "/user/ssouihel/home//Desktop/nv_gain.txt" , "unit-time");
                std::ifstream f2(str.c_str());
                std::string line;
                for ( i = 0; std::getline(f2, line); ++i);
                double firing_rate=(double)raster->getSpikeCount(0)/(double)(raster->getNumberOfTimeSteps()+1);
                f1 << firing_rate << "\n";
                //f1 << theta<< "\t" << firing_rate<<"\n";
                //f1 << Jext<< "\t" << firing_rate<<"\n";
                f2.close();
                delete raster;
                theta+=theta_step;
                //I+=Istep;

        }//End of theta loop

        f1.close();
        return 0;
}
```

8. Programme C de simulation du modèle LIF entraîné par un bruit Gaussien modulé en moyenne

```
double func(double mu, double sigma , double w) {
double V, tau, V_th, V_re, dt, firing;
tau=2.0;
dt=0.1;
V_th=10.0;
V_re=0.0;
V=V_re;
firing=0;
double times[10000];
int j=0;
for (int i=0; i<1000; i++)
        {
        std::default_random_engine generator;
        std::normal_distribution<double> distribution(0,1);
        double gaussian = distribution(generator);
        //V=V+dt*((mu+mu*cos(w*i*dt)/5)-V)/tau+sqrt(dt)*sigma*sqrt(2.0*tau)*gaussian/tau; // ex E
        V=V+dt*((mu+mu*cos(2*M_PI*w*i*dt)/5)-V)/tau+sqrt(dt)*sigma*sqrt(2.0*tau)*gaussian/tau; // ex E
        //V=V+dt*(mu-V)/tau+sqrt(dt)*sigma*sqrt(2.0*tau)*gaussian/tau;
        printf("%lg\n",V);

        if (V>V_th) {
                        times[j]=i*dt;
                        V=V_re;
                        j++;
                    }

        }
for (int k=0;k<=j;k++) firing+=cos(w*times[k]);
firing = abs(firing);
return firing;
}

int main()
{
FILE* output = NULL;
double mu, sigma;
mu=20;//ex E
double tmp;
for (sigma=3;sigma<=3;sigma += 0.1)
        {char str[15];
        sprintf(str, "firing%lg.", (double)sigma);
        output = fopen(str, "w+");
        //for (mu=0.0;mu<=200.0;mu+=1.0) //ex B
        for (double w = 0.0; w<=10000000.0;w+=100)
        {
        fprintf (output,"% .5f\t", w);
        //fprintf (output,"% .5f\n", func(mu,sigma));// ex B
        fprintf (output,"% .5f\n", func(mu,sigma,w)); // ex E

        }
}
return 0;
}
```

9. Programme C de simulation de la dynamique du potentiel excitatoire post-synaptique

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <fstream>
#include <iostream>
#include <random>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_odeiv2.h>

#define alpha 2.0
#define tau_r 0.4098


struct param_type1 {
    double tau_h;
    double e;
    double J;
    double x;
};

struct param_type2 {
    double tau_d;
    double h;
    double u;
};

int func_h (double t, const double y[], double f[],void *params)
{
  (void)(t);
  struct param_type1 *my_params_pointer = (param_type1 *) params;
  f[0] = y[1];

  f[0] = -y[0]/(my_params_pointer->tau_h)+(my_params_pointer->J)*(my_params_pointer->x)*
alpha*log(1+exp(y[0]/alpha))/(my_params_pointer->tau_h)+(my_params_pointer->e)/(my_params_pointer->tau_h);
  return GSL_SUCCESS;
}

int func_x (double t, const double w[], double f[],void *params)
{
  (void)(t);
  struct param_type2 *my_params_pointer = (param_type2 *) params;
  f[0] = w[1];
  f[0] = (1-w[0])/(my_params_pointer->tau_d)-(my_params_pointer->u)*
alpha*log(1+exp((my_params_pointer->h)/alpha))*w[0];
  return GSL_SUCCESS;
}
```

```
int main ()
{FILE * results = fopen("results","w+");
struct param_type1 my_params1 = {0.05,-2.9,4.5,0.0};
gsl_odeiv2_system sys1 = {func_h, NULL , 2, &my_params1};
struct param_type2 my_params2 = {0.409,1.0,0.8};
gsl_odeiv2_system sys2 = {func_x, NULL , 2, &my_params2};
gsl_odeiv2_driver * d1 = gsl_odeiv2_driver_alloc_y_new (&sys1, gsl_odeiv2_step_rk8pd,1e-6, 1e-6, 0.0);
gsl_odeiv2_driver * d2 = gsl_odeiv2_driver_alloc_y_new (&sys2, gsl_odeiv2_step_rk8pd,1e-6, 1e-6, 0.0);
  double dt=0.001;
  double t = 0.0, t1 = 10000.0;
  double temp;
  double y[2] = { 0.0, 0.0 };
  double w[2] = { 0.0, 0.0 };
  double h_ante=y[0];
  double x_ante=w[0];
  double dx=0, dh=0;
  int status ;
  double u;
  while (t<=t1)
    { status = gsl_odeiv2_driver_apply_fixed_step (d1, &t,dt, 100, y);
      if (status != GSL_SUCCESS)
        {
          printf ("error, return value=%d\n", status);
          break;
        }
      dh=y[0]-h_ante;

      h_ante=y[0];
      //printf("%f\n",y[0]);
      my_params2.h=y[0];
      status = gsl_odeiv2_driver_apply_fixed_step (d2, &t,dt, 100, w);
      if (status != GSL_SUCCESS)
        {
          printf ("error, return value=%d\n", status);
          break;
        }
      dx=w[0]-x_ante;
      x_ante=w[0];
      my_params1.x=w[0];
      u=(1.0-w[0])/(tau_r*w[0]*alpha*log(1+exp(y[0]/alpha)));
      printf("%f\n",u);
      if (y[0]>=0 && w[0]>=0){ fprintf (results,"%.5e %.5e\t", t, y[0]);
                  fprintf (results,"%.5e\t", y[1]);
                  fprintf (results,"%.5e\t", w[0]);
                  fprintf (results,"%.5e\n", w[1]);
                }
       t-=dt;
    }
  fclose(results);
  gsl_odeiv2_driver_free (d1);
  gsl_odeiv2_driver_free (d2);
  return 0;
}
```