L'impact du bruit sur la dynamique d'un neurone

Catherine Beauchemin, Department de Physique, Université d'Ottawa

6 décembre 2006

Résumé

Étude de la dynamique d'un neurone à travers des simulations utilisant le modèle de Morris-Lecar. Le modèle simulé est d'abord testé et comparé aux résultats obtenus par Rinzel et Ermentrout [7]. Le bruit est ensuite introduit dans le système à travers le courant d'entrée synaptique du neurone. À partir d'impulsions d'entrée arrivant à des temps aléatoires, nous avons pu tracer la relation entre la fréquence moyenne des entrées et la fréquence moyenne des potentiels d'action. Finalement, nous avons modifié l'aire du neurone afin d'observer comment elle influence la fréquence moyenne des potentiels d'action. Ces derniers résultats sont en accord avec la recherche faite, à ce sujet, par Ronald F. Fox [3].

Table des matières

1 Introduction

Ce projet consiste à étudier, au moyen de simulations numériques, la dynamique d'un neurone en utilisant le modèle de Morris-Lecar et en y introduisant des variantes. Ces variantes sont introduites à travers la fréquence d'arrivée des courants d'entrée et l'introduction d'une variable modifiant l'aire du neurone afin d'étudier les effets du bruit sur les neurones en fonction de leur surface (ou densité de canaux ioniques).

Pour mieux comprendre le comportement du modèle simulé, il fallait d'abord acquérir une certaine intuition par rapport aux systèmes neuronaux, et c'est la synthèse de ce que j'ai appris que j'aimerais tout d'abord présenter.

1.1 Propriétés des neurones

Un neurone est principalement composé d'un axone, qui est spécialisé dans le transfert d'informations intracellulaires, de dendrites, qui sont souvent le site de réception de l'information en provenance des autres neurones et finalement, les synapses, qui sont le point de transfert d'informations entre les neurones (voir Figure 1).

La surface des densités postsynaptiques d'un neurone est recouverte de petites ouvertures ou pores à travers lesquels les ions peuvent passer ; il s'agit des canaux ioniques. Chaque canal ionique, qu'il soit perméable au Sodium (Na^+), au Potassium (K^+), au Calcium(Ca^{2+}), etc. peut être représenté par une ouverture munie d'une porte d'activation et d'une porte d'inactivation. La probabilité d'ouverture ou de fermeture de ces deux portes peut être prise en compte par un facteur compris entre 0 et 1 qui dépend du potentiel de membrane.

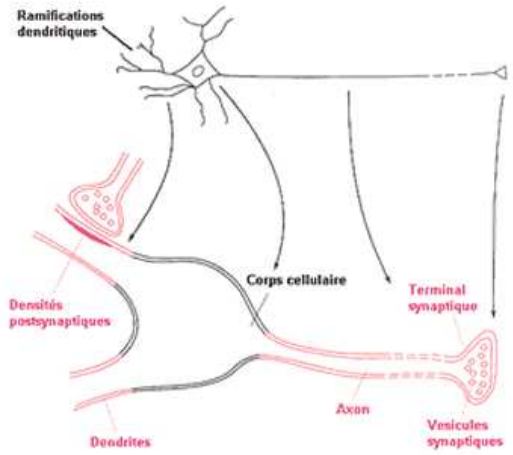

FIG. 1 – Structure fondamentale d'un neurone (extrait de [5]).

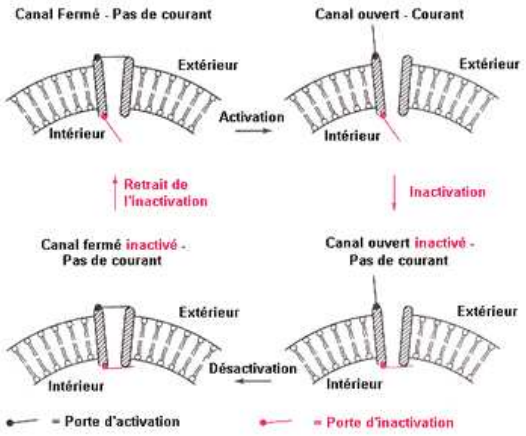

FIG. 2 – L'activation vs l'inactivation (extrait de [5]).

Lorsque qu'on fait varier le potentiel de membrane brusquement, la probabilité que les canaux soient ouverts peut changer rapidement, jusqu'à ce qu'un nouvel état stable de probabilité de canaux ouverts soit atteint pour le nouveau potentiel. Cette augmentation dans la probabilité de trouver des canaux ouverts, suite à un changement brusque du potentiel de membrane, s'appelle l'*activation*. Certains canaux, après que leur ouverture aura été induite par un changement de potentiel, vont maintenir leur facteur d'activation pour une période de temps prolongée. Par contre, pour d'autres canaux, la période d'activation est suivie d'une période de diminution de la probabilité de canaux ouvert qu'on appelle l'*inactivation* (voir Figure 2).

Pour les neurones, un potentiel de membrane nul, i.e. $V = 0\,\text{mV}$, est défini comme étant le cas où la concentration d'ions est la même, de part et d'autre de la membrane. Quand l'axone est au repos, i.e. quand aucun ion ne circule à travers la membrane, la valeur du potentiel de membrane est appelée *potentiel de repos*. Dans les neurones, le potentiel de repos est de l'ordre de -40 à $-90\,\text{mV}$ [5]. Lorsque le potentiel de membrane est supérieur au potentiel de repos, on dit que le neurone est *dépolarisé*, lorsqu'il est inférieur, le neurone est *hyperpolarisé*. Dans le cas d'un stimulus dépolarisant de faible amplitude ou d'un stimulus hyperpolarisant, peu importe son amplitude, le neurone réagit de façon passive. Par exemple, si le courant

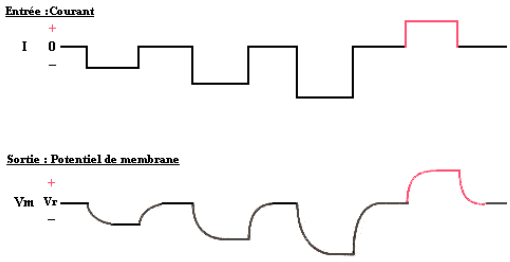

FIG. 3 – Reponse passive (extrait de [5]).

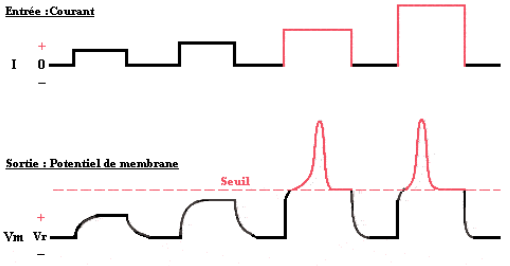

FIG. 4 – Reponse active (extrait de [5]).

d'entrée est négatif, le potentiel de membrane va s'hyperpolariser et l'amplitude du changement reflètera simplement l'amplitude du courant appliqué (voir ci-dessous Figure 3).

Cependant, dans le cas d'un stimulus dépolarisant plus important, on peut voir survenir ce qu'on appelle un potentiel d'action. Ce dernier survient lorsque l'amplitude de la dépolarisation dépasse une certaine valeur critique : le *seuil*. Toute amplitude de dépolarisation inférieure au seuil appelle une réponse passive, mais lorsque l'amplitude de la dépolarisation dépasse le seuil, on peut alors observer une réponse active de la membrane : un influx nerveux appelé *potentiel d'action*. C'est de lui que dépend le transfert d'information d'une partie du neurone à l'autre. Le potentiel d'action est caractérisé par un mode de fonctionnement tout ou rien ; il répond de manière invariable en amplitude, durée et forme quelle que soit l'importance du stimulus, pourvu que le seuil soit atteint (voir Figure 4).

Même si l'amplitude du potentiel d'action ne dépend pas de l'intensité du stimulus, il en est autrement de ces autres propriétés. C'est d'ailleurs ces autres propriétés qui nous renseignent sur la nature (amplitude, fréquence, etc.) du stimulus. Par exemple, la *période de latence*, i.e. le délai entre l'influx de courant et le maximum du potentiel d'action, dépend directement de l'amplitude du stimulus. Plus le stimulus est grand, plus la période de latence est courte (observer attentivement Figure 4).

Il est également important de réaliser que le seuil n'est pas une propriété fixe du système ou du neurone. Ceci nous amène à introduire une autre propriété du potentiel d'action : la *période réfractaire*. Immédiatement après un potentiel d'action, il est impossible d'en déclencher un suivant, peu importe l'amplitude du stimulus. On peut alors dire que le seuil est infini : c'est la période réfractaire absolue. Le seuil redescend ensuite mais pendant ce temps, le seuil est tout de même supérieur à sa valeur de repos : c'est la période réfractaire relative. Finalement, après quelque temps, le seuil aura repris sa valeur de repos (voir Figure 5).

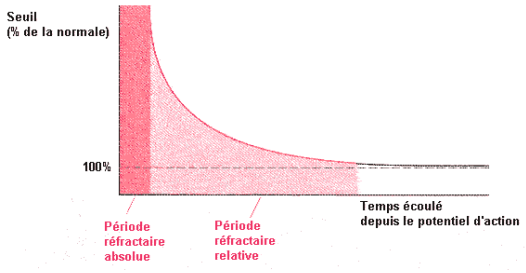

FIG. 5 – Période réfractaire (extrait de [5]).

1.2 Le modèle Morris-Lecar

Le modèle de Morris-Lecar tient bien évidemment compte de plusieurs des caractéristiques des neurones. Il a également ses limites en ce qu'il s'agit, entre autre, d'un modèle exprimant le comportement d'un neurone seul et que l'on ne considère principalement que deux types de canaux ioniques (Potassium (K^+) et Calcium (Ca^{2+}). Il s'agit d'un hybride du modèle de Hodgkin-Huxley et de celui de FitzHugh-Nagumo qui fut formulé et étudié pour la première fois par Morris et Lecar (1981) dans le contexte de l'activité électrique de la fibre musculaire de la balane (mollusque). Le modèle est décrit par les équations ci-dessous (tirées de Rinzel et Ermentrout 1998, [7]) :

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = -I_{\mathrm{ion}}(V,w) + I \tag{1}$$

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \phi\frac{w_\infty(V) - w}{\tau_w(V)} \tag{2}$$

où

$$I_{\mathrm{ion}}(V,w) = \bar{g}_{\mathrm{Ca}} m_\infty(V)\,[V - V_{\mathrm{Ca}}] + \bar{g}_{\mathrm{K}} w\,[V - V_{\mathrm{K}}] + \bar{g}_{\mathrm{L}}\,[V - V_{\mathrm{L}}] \tag{3}$$

$$m_\infty(V) = 0.5\left[1 + \tanh\left(\frac{V - V_1}{V_2}\right)\right] \tag{4}$$

$$w_\infty(V) = 0.5\left[1 + \tanh\left(\frac{V - V_3}{V_4}\right)\right] \tag{5}$$

$$\tau_w(V) = \mathrm{sech}\left(\frac{V - V_3}{2V_4}\right) \tag{6}$$

Dans ces formules, V est le potentiel (mV) et C est la capacité de la membrane ($\mu F/cm^2$). Comme on peut le voir, ce modèle comprend des canaux de Potassium (K^+) et de Calcium (Ca^{2+}) et regroupe dans les termes de perte (désignés par l'indice L pour «leakage») les canaux de moindre importance pour ce modèle. Les variables $\bar{g}_{\mathrm{Ca}}$, $\bar{g}_{\mathrm{K}}$ et $\bar{g}_{\mathrm{L}}$ représentent les conductances maximales des différents canaux (i.e. en considérant tous les canaux ouverts). Elles sont en fait le produit de la conductance d'un seul canal ouvert (pS) et de la densité de canaux (#canaux/mm^2), d'où leurs unités de densité de conductance (mS/cm^2) [6].

Les variables w et m_∞, respectivement décrites par (2) et (4), toutes deux sans dimension, sont les paramètres d'état et représentent la fraction de canaux ouverts. Tel que discuté ci-dessus, les canaux peuvent se trouver dans différents états (activés/désactivés ou inactivés/non-inactivés). Ainsi, les conductances des canaux doivent être «modulées» par leur état ; c'est la fonction que remplissent les paramètres d'états.

Le w est le facteur d'activation des canaux K^+ et représente donc la fraction de canaux de Potassium qui sont ouverts en fonction de la température (la variable ϕ), du potentiel et du temps. Étant donné que la fraction de canaux de Potassium ouverts en fonction de V n'atteint pas sa valeur d'équilibre de façon instantanée, w est donné en fonction de w_∞ , la valeur d'équilibre pour un potentiel donné, et de τ_w, la constante de temps représentant le temps nécessaire pour atteindre cet équilibre.

Comme les canaux Ca^{2+} répondent très rapidement à V, on considère ici qu'ils ont une activation instantanée, i.e. on néglige leur dynamique transitoire et on s'intéresse seulement à la valeur d'équilibre de la fraction de canaux ouverts, donc m_∞ ne dépend que du potentiel et aucunement du temps.

Lorsqu'on trace le graphique de w_∞ et m_∞ , on remarque qu'ils ont une forme sigmoïdale alors que τ_w a plutôt une forme gaussienne. Les potentiels V_1, V_2, V_3 et V_4, respectivement -1.2 mV, 18 mV, 2 mV et 30 mV, sont des paramètres expérimentaux qui dépendent du modèle étudié. Les potentiels V_{Ca}, V_{K} et V_{L}, respectivement 120 mV, -84 mV et -60 mV, sont les potentiels de renversement («reversal potentials») qui caractérisent tout simplement le potentiel auquel aucun ion de cette espèce ne circule à travers les canaux, qu'ils soient ouverts ou fermés. La proportion de canaux de chaque espèce ainsi que leurs potentiels de renversement respectifs définiront le potentiel de repos du neurone.

Finalement, le courant ionique, I_{ion} (3), fait référence au courant inhérent au neurone. Il s'agit du courant résultant du flux des ions traversant la membrane par l'intermédiaire des canaux. Ce courant est donc induit par le potentiel de membrane qui résulte en l'ouverture et la fermeture des canaux ioniques. Le dernier paramètre, I, est généralement le seul paramètre libre et nous l'utiliseront afin de simuler les entrées de courant provenant des neurones voisines.

1.3 Le bruit

Tel que mentionné au tout début, nous voulons étudier l'effet du bruit sur le comportement excitatif des neurones, i.e. comprendre de quelle façon le bruit fera la différence entre le déclenchement d'un potentiel d'action et une simple réponse passive.

Dans notre modèle, c'est le terme I qui sera utilisé afin d'introduire du bruit dans le système. Physiologiquement, le paramètre I (1) correspond aux courants provenant des neurones voisins. Pour simuler la réalité physiologique du modèle, on peut intégrer le comportement des quelques voisins immédiats ou introduire un courant bruyant, qui reflèterait la variabilité du système résultant des nombreux voisins réels du neurone d'intérêt.

Comme nous l'avons dit, le bruit est introduit dans notre système à travers le courant externe, I, par opposition au courant inhérent au neurone, I_{ion}. Il peut prendre des formes diverses qui peuvent varier dans le temps et peut être représenté de la façon suivante :

$$I = I_0 + \sum_N f_N(t, t_N) + \text{bruit} \tag{7}$$

Le courant de base, I_0, peut tout simplement être une constante, mais pourrait également être une oscillation sinusoïdale, par exemple. Du bruit gaussien pourrait également être ajouté à ce courant de base. Finalement les fonctions $f_N(t, t_N)$ pourraient, par exemple, être de petites impulsions de formes et d'amplitudes variables arrivant à des temps t_N qui leur sont propres. Les fonctions $f_N(t, t_N)$ peuvent, elles aussi, être affectées d'un bruit gaussien.

Remarquez que pour être un peu plus près de la réalité physiologique du neurone, on aurait pu choisir un courant synaptique qui dépende du potentiel. En effet, les voisins proches ressentiront des valeurs du potentiel similaires à celles du neurone d'intérêt et produiront ainsi un courant qui dépend du potentiel. Ceci est discuté en un peu plus de détail ci-dessous (voir Section 4).

```
 v1 = ...
...
M = 0.5 * [ 1 + tanh ( y(1) - v1 ) / v2)  ]
W = 0.5 * [ 1 + tanh ( y(1) - v3 ) / v4)  ]
Tau = 1 / cosh[ (y(1) - v3) / (2*v4) ]
dy(1) = [ -gCa*M*(y(1)-vCa) - gK*y(2)*(y(1)-vK) - gL*(y(1)-vL) + I ] / C
dy(2) = phi * ( W - y(2) ) / Tau
```

TAB. 1 – Pseudo-code pour la sous-routine `ml`.

1.4 Le but

Il s'agissait donc de modéliser la dynamique d'un neurone à partir du système d'équation de Morris-Lecar. Une fois le modèle testé, nous nous sommes intéressés à l'impact qu'aurait la fréquence moyenne d'arrivée des entrées sur la fréquence moyenne de déclenchement de potentiels d'action.

Finalement, nous voulions voir de quelle façon l'influence qu'a le bruit sur le système dépend de la surface du neurone (densité de canaux ioniques). Dans un article de DeFelice et Goolsby [1] puis un autre de Strassberg et DeFelice [8], on nous présente un graphique de la fréquence de décharge neuronale (potentiel d'action) en fonction de la surface d'un neurone. On remarque que la fréquence de décharge augmente d'abord en fonction de la surface (qui augmente aussi), atteint un maximum puis diminue ensuite.

Nous sommes intéressés à comprendre pourquoi la fréquence de décharge moyenne diminuerait lorsque la surface du neurone diminue (la première partie du graphe – Figure 21.2 [1]). Mais principalement, nous voudrions vérifier si la forme de la courbe est pertinente, malgré l'erreur dont elle est affectée.

2 Méthode

La première étape du projet a été d'élaborer et de tester un programme qui permettrait de simuler l'activité du système à l'aide du modèle de Morris-Lecar. Nous avons écrit, à cette fin, un programme en `Fortran77`. Le modèle est entièrement défini par les équations présentées ci-dessus (1) à (6) et est propagé dans le temps à l'aide de l'algorithme d'Euler (voir ci-dessous, Section 2.2).

2.1 Sous-routine `ml`

La sous-routine `ml`, telle qu'on la retrouve dans tous les programmes se trouvant en Annexe, calcule la valeur de toutes les variables du modèle de Morris-Lecar pour un temps, un potentiel, un w et un courant donné (voir pseudo-code, Tab. 1).

Les paramètres utilisés sont ceux fournis dans « Analysis of Neural Excitability and Oscillations (Rinzel et Ermentrout) » [7]. Les noms donnés aux variables indiquent clairement ce qu'elles représentent avec, par exemple, `M`, `W`, et `Tau` représentant $m_\infty(V)$, $w_\infty(V)$ et $\tau_w(V)$. Les `dy(n)` représentent les valeurs non-intégrées (dérivée par rapport au temps), alors que les `y(n)` représentent les valeurs qui ont été intégrées numériquement. Par exemple, `dy(1)` représente $\mathrm{d}V/\mathrm{d}t$ et `y(1)` représente V, avec l'indice «1» représentant V, le potentiel, et l'indice «2» représentant w, le facteur d'activation des canaux K^+.

```
Pour n allant de 1 à nvar
    y(n) = y(n) + dt * dy(n)
```

TAB. 2 – Pseudo-code pour la sous-routine `euler`.

```
Pour un courant constant :
    I = une constante quelconque
                    OU
Pour des impulsions carrées de courant :
    Si le temps est dans l'intervalle [t1, t2]
        Alors I = courant de base + courant du pulse
        Sinon I = courant de base
```

TAB. 3 – Pseudo-code pour la sous-routine `defineI` (1re implémentation).

2.2 Sous-routine `euler`

L'algorithme utilisé pour l'intégration numérique est Euler. Il est calculé par la sous-routine `euler` telle qu'on la retrouve dans tous les programmes présentés en Annexe. La propagation d'une solution dans le temps à l'aide de l'algorithme d'Euler se fait tel qu'illustré par le pseudo-code présenté, Tab. 2.

L'algorithme est assez simple. Les variables `y(n)` et `dy(n)` sont les mêmes que celles de la sous-routine `ml` et sont d'ailleurs passées à la sous-routine euler à travers le programme principal (voir codes en Annexe). La variable `nvar` contient le nombre de variables intégrables emmagasinées dans les tableaux `y(5)` et `dy(5)`. Cette flexibilité additionnelle a été ajoutée au programme afin de pouvoir ajouter facilement, au besoin, des variables dynamiques (dépendantes du temps) additionnelles au modèle.

Il est important de justifier l'utilisation d'Euler comme algorithme d'intégration. Ce choix est en fait motivé par l'utilisation que l'on compte faire du modèle. En effet, étant donné la nature stochastique que nous comptons ajouter à certains des paramètres du modèle, il n'était pas envisageable d'utiliser une méthode comme, par exemple, Runge-Kutta, compte tenu du fait que les termes stochastiques sont d'ordre $\sqrt{\texttt{dt}}$. Bien sûr, d'autres algorithmes auraient pu être utilisés, comme par exemple Euler Mid-Point, qui auraient pu prendre en compte le courant stochastique. Mais le temps alloué au projet étant limité et la stabilité d'Euler semblant satisfaisante pour un pas d'intégration $\texttt{dt} = 0.005$, et ce, sans trop alourdir le temps de calcul des simulations, nous avons décidé de nous en satisfaire.

2.3 Sous-routine `defineI` (1re implémentation)

Une sous-routine à part, `defineI`, permet de définir le courant de façon à simplifier le code et le garder simple (ajouter le calcul du courant à la sous-routine `ml` aurait alourdit le code et l'aurait rendu moins clair). La sous-routine `defineI` prends comme argument d'entrée `t`, le temps, et comme argument de sortie `I`, qui est bien-sûr le courant au temps `t`. La sous-routine `defineI` est appelée à chaque itération du programme principal pour un nouveau temps, i.e. `t+dt`. La première implémentation du code de `defineI`, pour un courant, constant se trouve dans le programme `neuron.f` (voir Annexe A). Elle a été écrite afin de tester le programme en comparant les simulations obtenues aux figures présentées dans l'article de Rinzel et Ermentrout [7].

La sous-routine `defineI` prenait la forme illustrée par le pseudo-code (voir Tab. 3). Selon le type de courant désiré (courant constant, impulsions de courant, etc.), les parties inutiles étaient mises en commentaire,

Pour `t` allant de `min` à `max` par bonds de `dt` : appelle la sous-routine `defineI` appelle la sous-routine `ml` appelle la sous-routine `euler` Si `k = okw`, Alors : les résultats sont inscrits dans un fichier de sortie et `k` est initialisé Sinon : rien n'est inscrit et `k` est incrémenté

TAB. 4 – Pseudo-code pour le programme principal (1re implémentation).

et seules les parties pertinentes au problème étaient exécutées. Les résultats des simulations visant à tester cette première implémentation du programme sont présentés aux Figures 6 et 7.

2.4 Programme principal (1re implémentation)

Toutes ces sous-routines étaient appelées par le programme principal, tel que vous le trouverez ci-dessous en Annexe. Le programme contient plusieurs variables et il est intéressant de clarifier ici leur fonction. La variable model, est utilisée afin de choisir entre deux groupes différents de paramètres pour le modèle Morris-Lecar. Ces deux groupes de paramètres sont ceux proposés et utilisés dans l'article de Rinzel et Ermentrout [7].

Les variables `min`, `max`, `dt` et `t` sont toutes reliées au temps. La variable `min` représente le temps auquel la simulation est initiée, `max`, celui auquel elle se termine, `dt`, l'intervalle de temps représenté par chaque itération et finalement `t`, le temps lors de l'itération.

Les variables `k` et `okw` servent à accélérer le programme et à alléger les fichiers de données en limitant l'enregistrement des données à toutes les 0.05 ms (les unités de temps du programme étant en millisecondes). Ainsi, `k` est le compteur et `okw` est la valeur de `k` pour laquelle les données seront inscrites dans le fichier de sortie.

Le programme principal, décrit en pseudo-code (voir Tab. 4), m'a permis de comparer mes résultats à ceux de l'article mais également d'explorer le comportement du modèle dans différentes conditions. La première implémentation du programme principal est présentée à l'Annexe A. Le programme principal ne demeurera pas inchangé au cours des différentes implémentations qui suivront, mais conservera sa forme générale. Tous les programmes implémentés et utilisés sont présentés en Annexe.

2.5 Simulation #1 : courant constant

La première simulation a très bien fonctionnée car les auteurs, Rinzel et Ermentrout [7], ont pris soin de donner, dans leur article, les conditions initiales pour le courant (`I` = 0), le facteur d'activation (`w` = 0.015) et les divers potentiels (`V` = -10.0 mV, -13.9 mV, -14.0 mV et -16.0 mV). Les résultats obtenus étaient identiques à ce qui était présenté dans l'article (voir Figures 6 et 7 et comparer aux Figures 7.1 et 7.3B dans [7]).

Figure 6(a) représente le potentiel de membrane (mV) en fonction du temps (ms) alors que Figure 6 représente le facteur d'activation des canaux potassium ($w(t, V)$) en fonction du potentiel de membrane ($V(t)$). Ce dernier type de graphe (w vs V) est appelé profil dans le plan de phase, un terme employé pour désigner tout graphe de variables dont le temps paramétrise les courbes. On remarque que pour les conditions initiales $V_0 = -10$ mV et -13.9 mV, il y a potentiel d'action alors que pour $V_0 = -14$ mV et

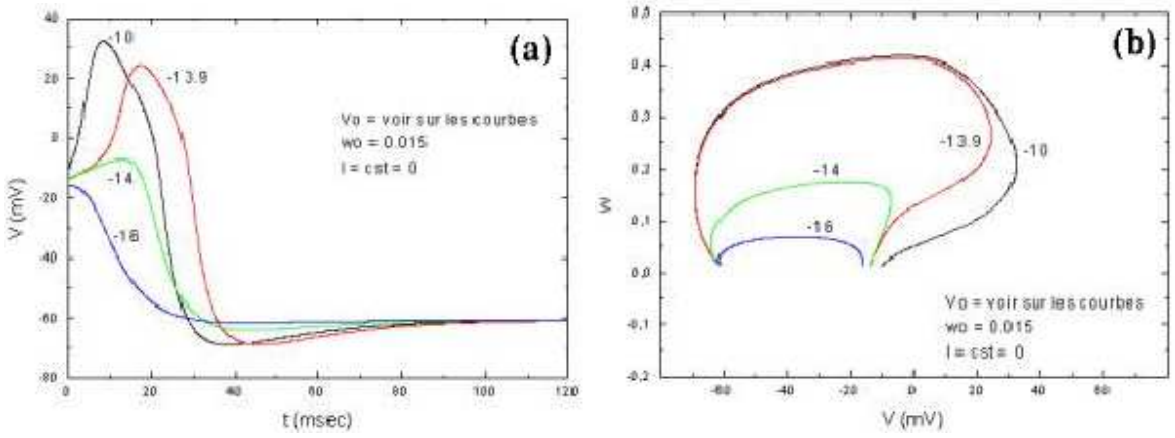

FIG. 6 – Simulation à courant constant

-16 mV, la réponse est passive.

2.6 Simulation #2 : impulsions carrées de courant

La seconde simulation consistait à reproduire la Figure 7.3 du même article [7]. Cette figure est particulièrement intéressante car elle représente le comportement du système dans le cas où l'on donnerait deux impulsions de courant de 30 μA/cm^2 d'une durée de 5 ms aux temps $t = 100$ ms et $t = 470$ ms. Ces impulsions sont superposées à un courant de base de 90 μA/cm^2. Dans la sous-routine `defineI` tel qu'implémentée dans `neuron.f` (voir Annexe A), cette situation a été traduite en `Fortran77`.

La dynamique ainsi obtenue est présentée à la Figure 7. Les conditions initiales n'ayant pas été mentionnées dans l'article, il a été un peu plus difficile d'obtenir une figure identique à celle présentée dans l'article. En essayant différents paramètres et en corrigeant selon les résultats obtenus, nous en sommes venus à utiliser les paramètres initiaux suivants : $V_0 = -26.631$ mV et $w_0 = 0.13$.

Remarquez que la première impulsion de courant fait entrer notre système dans un régime oscillatoire alors que la seconde impulsion le ramène à un point fixe (foyer). Ceci signifie que pour les valeurs de courant utilisées ici, notre système démontre une bistabilité : un point fixe stable et un état oscillatoire stable coexistent.

2.7 Sous-routine `defineI` (2e implémentation)

Après avoir fait quelques autres comparaisons satisfaisantes avec les figures de l'article (notamment Figures 7.4, 7.5, 7.8 dans [7]), nous avons pu commencer à implémenter de façon plus poussée le code de la sous-routine `defineI`. Le but était maintenant de superposer à un courant de base des impulsions de courant ayant la forme d'une fonction alpha, soit :

$$I = I_0 + \sum_N C \cdot (t - t_N) \cdot \mathrm{e}^{-\alpha(t-t_N)} \tag{8}$$

Pour débuter, nous avons décidé d'implémenter le cas le plus simple : des impulsions de courant survenant à des intervalles de temps constants. De plus, il nous fallait nous débarrasser des temps trop anciens lorsque

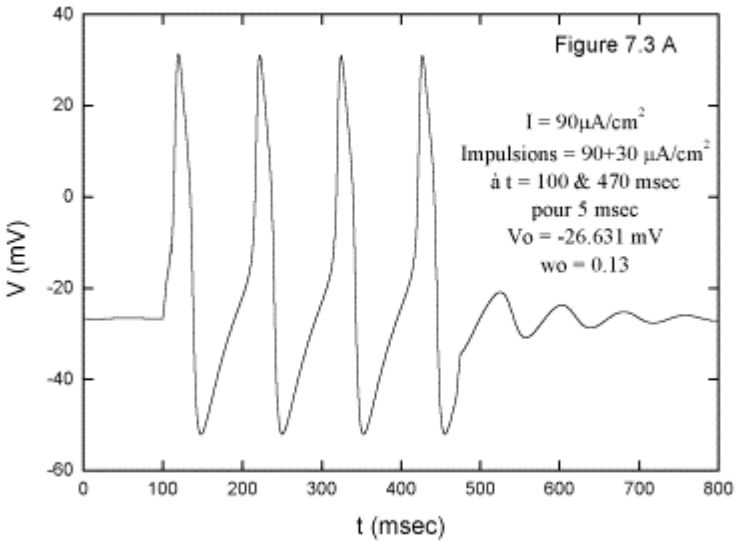

FIG. 7 – Impulsions carrées de courant (simulation)

leur contribution n'était plus significative. Le code pour les impulsions de courant alpha est implémenté dans la sous-routine `defineI`, telle qu'elle est définie dans le programme `npoiscon.f` (voir Annexe B). Elle est également illustrée par le pseudo-code (voir Tab. 5).

La variable `t` contient le temps présent et `itvl` est l'intervalle de temps qui s'écoule entre deux impulsions de courant. La variable `tp(0:100)` est un tableau qui mémorise les temps d'arrivée de chaque impulsion et `tmax` est l'indice du temps d'arrivé le plus récent.

Les variable `C` et `a` représentent respectivement l'amplitude et le paramètre alpha de la fonction alpha tel qu'illustré en (8). La valeur mise en mémoire à l'indice zéro du tableau `tp` correspond à I_0 de (8), le courant de base auquel est ajouté les impulsions de courant.

Lorsque `tp(i)` est parcouru pour sommer la contribution de chacune des impulsions, le programme vérifie en même temps si cette contribution est encore significative. Si elle ne l'est plus, i.e. si la contribution est plus petite qu'un certain facteur de tolérance, mémorisé par la variable `tol`, alors tous les termes de `tp` suivant sont décalés à un indice inférieur et `tmax` est décrémenté. Étant donné qu'à ce point-ci, les impulsions ont toutes la même forme et la même amplitude, il aurait suffit de vérifier la contribution du temps le plus ancien puisque celle des temps ultérieurs est inévitablement supérieure. Cependant, comme nous comptons éventuellement faire varier l'amplitude et le paramètre alpha des impulsions (les variables `C` et `a`) de façon aléatoire, il valait mieux écrire tout de suite le code en conséquence.

De plus, `tnext`, le temps de la prochaine impulsion est toujours défini à l'avance. Ainsi, lorsque `t` correspond à `tnext`, on ajoute un temps d'impulsion à `tp`, on incrémente `tmax` et on détermine tout de suite `tnext` suivant. Dans le cas d'impulsions à intervalles constants, `tnext` devient tout simplement `tnext = t + itvl`.

Pour `n` allant de `1` à `tmax`, faire :
    `tempo = C(n)*( t - tp(n) )*exp[- a(n)*( t - tp(n) )]`
    Si `tempo` n'est plus significatif (`tempo` $<$ `tol`)
        Alors : décrémente `tmax` et retire le $\mathtt{n}^{\text{ième}}$ terme de `tp(n)`
        Sinon : `I = I + tempo`

Ensuite,
    `I` = `I` + courant de base

Lorsque `t = tnext`,
    On incrémente `tmax` (`tmax = tmax + 1`)
    On ajoute `t` aux temps d'arrivés (`tp(tmax) = t`)
    On détermine le `tnext` (`tnext = tnext + itvl`)

TAB. 5 – Pseudo-code pour la sous-routine `defineI` (2e implémentation).

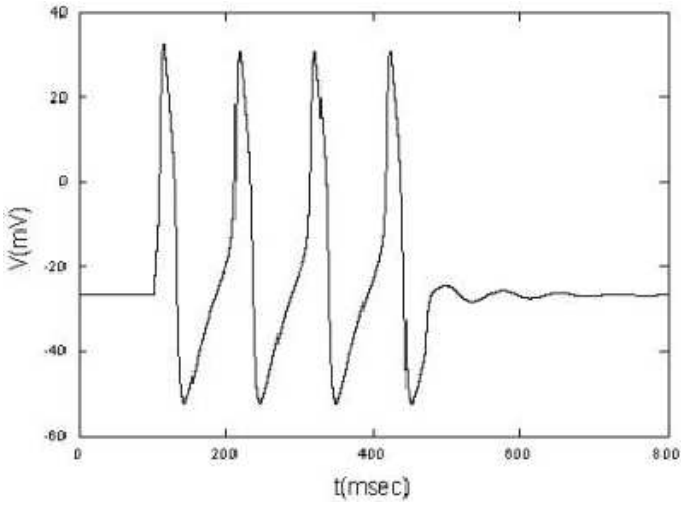

FIG. 8 – Impulsions carrées de courant (simulation)

2.8 Simulation #3 : impulsions de courant de forme alpha

Calquer les conditions de la Figure 7 était un bon moyen de vérifier comment se comportait les impulsions de courant de forme alpha. Cette fois, plutôt que de faire survenir des impulsions carrées, i.e. des pulses de 30 μA/cm^2 durant 5 ms, le programme allait produire des impulsions ayant la forme de fonctions alpha avec les paramètres suivants : $C = 65\ \text{ms}^{-1} \cdot \mu\text{A/cm}^2$ et $a = 0.8\ \text{ms}^{-1}$. Ces paramètres ont été choisis afin que l'aire sous la courbe de chacune des impulsions soit d'environ $150\ \text{ms} \cdot \mu\text{A/cm}^2$, imitant ainsi l'aire sous la courbe des impulsions carrées vues précédemment. Et les résultats sont assez satisfaisants (voir Figure 8). En effet, la Figure 8 est assez fidèle à ce que l'on attendait ; elle ne diffère que très peu de la Figure 7.

Pour `n` allant de `1` à `tmax`, faire :
    `tempo = C(n)*( t - tp(n) )*exp[- a(n)*( t - tp(n) )]`
    Si `tempo` n'est plus significatif (`tempo` $<$ `tol`)
        Alors : décrémente `tmax` et retire le $\mathtt{n}^{ième}$ terme de `tp(n)`,`C(n)` et `a(n)`
        Sinon : `I = I + tempo`

Ensuite,
    `I` = `I` + courant de base + bruit gaussien

Lorsque `t = tnext`,
    On incrémente `tmax` (`tmax = tmax + 1`)
    On ajoute `t` aux temps d'arrivés (`tp(tmax) = t`)
    On détermine l'amplitude et le paramètre alpha de la nouvelle impulsion
        (`C(tmax)` = `C` + bruit gaussien
        `a(tmax)` = `a` + bruit gaussien)
    On détermine le `tnext` d'intervalle aléatoire
        (`tnext = t + myexpdev`)

TAB. 6 – Pseudo-code pour la sous-routine `defineI` (3e implémentation).

2.9 Sous-routine `defineI` (3e implémentation)

Le code de cette 3e implémentation se trouve dans le programme `naire.f` présenté à l'Annexe C. La première chose à faire était d'ajouter au programme les trois générateurs de nombres aléatoires dont nous allions avoir besoin, soit : `ran1(idum)`, `gasdev(idum)` et `myexpdev(idum)`. Les deux premières fonctions sont extraites de « Numerical Recipes in Fortran » et la troisième est une version modifiée de la fonction `expdev(idum)` proposée dans le même ouvrage [2].

La variable `idum` est un entier. Si `idum` est négatif, la fonction `ran1(idum)` est initialisée. La fonction `ran1(idum)` prend un entier et renvoie un nombre aléatoire de distribution uniforme entre 0 et 1. La fonction `gasdev(idum)` prend un entier et retourne un nombre aléatoire de distribution gaussienne avec une moyenne de zéro et une variance unitaire. Finalement, `myexpdev(ia, idum)` prends deux entiers, `ia` et `idum`, et renvoie un nombre aléatoire de distribution exponentielle de moyenne et d'écart-type `ia`. Les fonctions `gasdev(idum)` et `myexpdev(ia, idum)` utilisent toutes les deux `ran1(idum)`.

Dans cette nouvelle implémentation de `defineI`, on remarque que les variables `C` et `a` des fonctions alpha (voir Tab. 6) sont devenues des tableaux de même dimension que `tp` et contiennent maintenant l'amplitude et le paramètre alpha de l'impulsion de forme alpha arrivée au temps `t(n)`. Chaque fois qu'une des fonctions alpha ne contribue plus significativement, elle doit maintenant être retirée des tableaux `tp(n)`, `a(n)` et `C(n)`.

Quand une nouvelle impulsion de forme alpha survient (i.e. `t = tnext`), `tmax` doit être incrémenté, l'amplitude et le paramètre alpha sont déterminés par `C(tmax) = 87.0d0 + gasdev(rc)` et `a(tmax) = 6.0d-1 + gasdev(ra)`. Puis on détermine le prochain temps d'arrivé de la façon suivante : `tnext = t + myexpdev(ria, rdt)`. La fonction `myexpdev(ia, idum)` est une modification de `expdev(idum)` proposée dans [2]. Plutôt que de produire un nombre aléatoire de distribution exponentielle de moyenne et d'écart-type unitaire, il produit un nombre aléatoire de distribution exponentielle de moyenne et d'écart-type `ia`. Autrement dit, le temps d'attente moyen entre deux impulsions sera de `ia` ms et la fréquence moyenne des impulsions d'entrée sera donnée par 1/`ia`.

Finalement, un bruit gaussien `gasdev(ribase)` est ajouté au courant de base `tp(0)` et au courant résultant de la somme des impulsions dont la contribution est encore significative.

Pour une `fmo = 1/ia` donnée,

Pour le temps `t` entre `min` et `max` :
    S'il y a eu potentiel d'action (i.e. `y(1)` > `thresh`) au temps `t`, alors (
        `nisi = nisi + 1`
        `isi(nisi)` = temps actuel - temps du PA précédent
        temps du PA précédent = temps actuel
        `fmo = fmo + isi(nisi)`
)

Finalement, la fréquence moyenne de sortie (`fmo`) :
    `fmo = nisi / fmo`

Répéter le tout pour «`nbrfile`» réalisations.

TAB. 7 – Calcul de la `fmo` (fréquence moyenne de sortie).

Pour une vue d'ensemble de `defineI` à ce point, le code complet peut être consulté en Annexe (voir Annexe C, programme `naire.f`).

3 Résultats

3.1 La fréquence moyenne de sortie vs la fréquence moyenne d'entrée

Il s'agissait ensuite d'implémenter la routine permettant de vérifier l'influence de la fréquence moyenne d'entrée (ci-après désignée par `fmi`) sur la fréquence moyenne de sortie (ci-après désignée par `fmo`). Nous avons procédé en utilisant des impulsions d'entrée de forme alpha, telles qu'implémentées dans `naire.f`, mais avons conservé les paramètres textttC et texttta constants (avec `a` $= 0.6$ ms^{-1} et `C` $= 87.0$ ms$^{-1} \cdot \mu$A/cm^2). Cependant, ces impulsions arrivaient à des intervalles de temps aléatoires de distribution exponentielle de moyenne et d'écart-type `ia` ms (donc `fmi` $= 1/$`ia` mHz). Le calcul de la `fmo` était fait automatiquement par le programme selon le pseudo-code illustré au Tab. 7.

Les entrées étant aléatoires, il était nécessaire de simuler pour chaque `ia` sur plusieurs réalisations et de calculer ensuite les moyennes des `fmo` obtenues à chacune des réalisations. Les résultats présentés à la Figure 9, ont été moyennés sur 30 réalisations de temps maximal `max` $= 10$ s, de pas d'intégration `dt` $= 5\text{e}-3$ ms et pour différents `ia` allant de 5, 10, 15, 20, 25, 50, 75 à 275 ms.

On voit que lorsque la `fmi` augmente, la `fmo` augmente aussi. On remarque également que pour des `fmi` plus élevées, une augmentation de la `fmi` résulte en de plus en plus petites augmentations de la `fmo`. Ce ralentissement dans la croissance de la `fmo` en fonction de la `fmi` est une conséquence directe de la période réfractaire. En d'autres mots, la période réfractaire module en quelque sorte la relation entre la `fmi` et la `fmo` et son influence est d'autant plus marquée pour des `fmi` élevées.

3.2 Le paramètre κ

Le comportement de la `fmo` en fonction de la `fmi` ayant été exploré, nous possédions assez d'informations sur la dynamique du système pour vérifier l'effet de la densité de canaux ioniques sur la `fmo` pour différentes

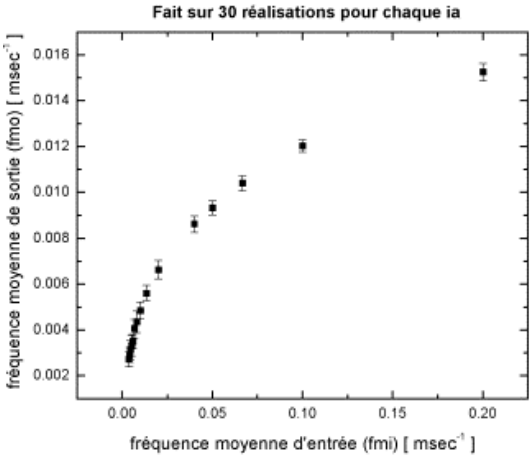

FIG. 9 – `fmo` en fonction de `fmi` (1/`ia`)

`fmi`, tel que l'a fait DeFelice dans les articles susmentionnés (voir Section 1.4 et [1, 8]).

Vous remarquerez que DeFelice parle de modifier l'aire du neurone, alors que nous préférons dire que c'est la densité de canaux qui est modifiée. Cependant, les deux énoncés sont dynamiquement équivalents. Considérer deux neurones d'aires égales et de densités de canaux différentes, ou deux neurones d'aires différentes mais de densités de canaux équivalentes, revient au même ; dans les deux cas, le nombre de canaux varie. Or, la dynamique du système dépend du nombre de canaux, et non de la densité des canaux ou de l'aire du neurone. Par contre, pour que ceci soit vrai, la proportion de canaux de chaque espèce doit être conservée lorsque la quantité de canaux est changée.

Afin d'augmenter la densité de canaux, nous avons introduit la variable κ, dans les équations du modèle de M.-L. présentées précédemment (1) à (6), de la façon suivante :

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \frac{-\bar{g}_{\mathrm{Ca}} m_\infty(V)\,[V - V_{\mathrm{Ca}}] - \bar{g}_{\mathrm{K}} w\,[V - V_{\mathrm{K}}] - \bar{g}_{\mathrm{L}}\,[V - V_{\mathrm{L}}] + I}{C\kappa} \tag{9}$$

Tel que mentionné à la Section 1.2, les variables $\bar{g}_{\mathrm{Ca}}$, $\bar{g}_{\mathrm{K}}$ et $\bar{g}_{\mathrm{L}}$ sont le produit de la conductance d'un seul canal ouvert (pS) et de la densité de canaux (#canaux/mm^2). La conductance d'un canal d'une espèce donnée étant une propriété physiologique du canal en question, elle ne peut varier. Ainsi, la division par la variable κ en (9) correspond physiologiquement à une réduction de la densité des canaux. La proportion de canaux de chaque espèce (Potassium, Calcium et autres) est donc conservée. De plus, vous remarquerez que le courant externe, I, doit également être divisé par κ car une quantité plus restreinte de canaux captera une fraction proportionnellement plus petite du courant externe.

Il est intéressant de remarquer que la variable κ peut également être perçue comme étant l'échelle caractéristique de temps de la dynamique du système. En effet, un κ plus petit rendra $\mathrm{d}V/\mathrm{d}t$ plus grand et accélèrera ainsi la dynamique du système. Finalement, κ peut également être vu comme étant un changement de l'amplitude de la capacitance. En effet, on aurait pu redéfinir la capacitance comme étant $C' = C \cdot \kappa$.

Ces diverses façons de définir le rôle de la variable κ sont mathématiquement équivalentes et, conséquemment, résultent en un changement équivalent de la dynamique du système. Cependant, étant donné que nous désirons comparer nos résultats à ceux obtenus par DeFelice [1] [8], nous considérerons ici κ comme étant une modification de la densité de canaux.

La Figure 10 révèle que la `fmo` augmente lorsque κ diminue et que la croissance est d'autant plus impor-

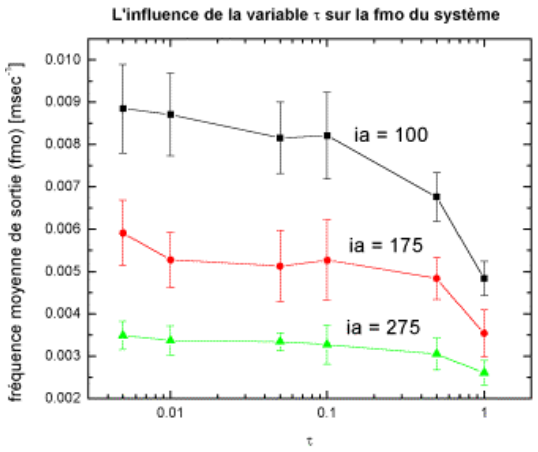

FIG. 10 – `fmo` en fonction de κ pour différentes `fmi` (1/`ia`)

tante lorsque la `fmi` augmente. Nous verrons, à la Section 3.3, pour quelle raison il en est ainsi.

3.3 Un modèle simple qui explique la dynamique du modèle de M.-L.

Les résultats vus ci-dessus (Sections 3.1 et 3.2) s'expliquent bien en terme d'une version simplifiée du modèle de M.-L., telle que proposée par Christof Koch [4]. Le modèle est le suivant :

$$\kappa C \frac{\mathrm{d}V}{\mathrm{d}t} = \frac{-V}{R} + a_e \frac{\mathrm{d}N}{dt} + I \tag{10}$$

$$\frac{\mathrm{d}V}{\mathrm{d}t} = -\frac{V}{\kappa C R} + \frac{a_e}{\kappa C}\frac{\mathrm{d}N}{dt} + \frac{I}{\kappa C} \tag{11}$$

où $R = g^{-1}$.

Dans ce modèle, le courant ionique, désigné par I_{ion} en (1) et (3) a été extrêmement simplifié par $I_{\text{ion}} = gV = V/R$, où la conductance a été remplacée par son inverse, le terme $R = g^{-1}$. Le courant externe (désigné par I en (1)) a été décomposé en un courant de base, I, et des impulsions de courant définies par $a_e \mathrm{d}N/\mathrm{d}t$. La trajectoire de la variable N est discontinue, car elle varie instantanément de ± 1 chaque fois qu'une entrée synaptique (impulsion) survient [4], de sorte que $\mathrm{d}N/\mathrm{d}t$ peut être modélisé par une fonction delta de Dirac, à chaque impulsion. Si a_e est suffisamment petit et $\mathrm{d}N/\mathrm{d}t$ est suffisamment grand, le processus ponctuel $a_e/\kappa C * \mathrm{d}N/\mathrm{d}t$ peut être approximé par un processus Gaussien de moyenne et de variance,

$$\text{moyenne} = \frac{a_e \mu_e}{\kappa C} \tag{12}$$

$$\text{vairance} = \frac{{a_e}^2 \mu_e}{\kappa^2 C^2} \tag{13}$$

où μ_e est la fréquence des entrées.

Ce résultat vient du théorème de Campbell qui veut que la variance = fréquence $\cdot \int h^2(t)\mathrm{d}t$ où $h(t)$ est la forme des impulsions. En notant que dans le cas du modèle simplifié de Koch $\int h^2(t)\mathrm{d}t = 1$, on peut réécrire (11) de la façon suivante :

$$\frac{\mathrm{d}V}{\mathrm{d}t} = -\frac{V}{\kappa C R} + \frac{I}{\kappa C} + \frac{a_e \mu_e}{\kappa C} + \xi(t)\sqrt{\frac{a_e^2 \mu_e}{\kappa^2 C^2}} \tag{14}$$

Posant $\sigma \equiv \sqrt{\frac{a_e^2 \mu_e}{\kappa^2 C^2}}$, $\alpha \equiv \frac{I}{\kappa C} + \frac{a_e \mu_e}{\kappa C}$ et $\gamma \equiv \kappa C R$, (14) devient :

$$\frac{\mathrm{d}V}{\mathrm{d}t} = -\frac{V}{\gamma} + \alpha + \xi(t)\sigma \tag{15}$$

On peut ensuite faire un changement de variables, en posant $x = V - \alpha\gamma$, et on peut réécrire (15) sous la forme :

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -\frac{x}{\gamma} + \alpha + \xi(t)\sigma \tag{16}$$

C'est un système pour lequel on doit rechercher une distribution de probabilité stationnaire en l'absence de potentiels d'action ou de réponse passive, i.e. en l'absence d'impulsions de courant. Remarquez que $x = 0$ (ou $V = \alpha\gamma$) est un point fixe déterministe.

En réalisant qu'on peut approximer le bruit (sans sa moyenne), $\xi(t)$, par du bruit blanc gaussien de moyenne nulle, on peut écrire l'équation de Fokker-Planck de notre système pour $\rho(x,t)$ de sorte que :

$$\frac{\partial \rho(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[-\frac{x}{\gamma}\rho(x,t)\right] + \frac{\sigma^2}{2}\frac{\partial^2 \rho(x,t)}{\partial x^2} \tag{17}$$

Or, comme nous l'avons déjà mentionné, nous voulons considérer une densité de probabilité stationnaire, soit $\rho^*(x)$ tel que $\lim_{t\to\infty} \rho(x,t) = \rho^*(x)$. En remplaçant $\rho(x,t)$ par $\rho^*(x)$ dans (17), on obtient :

$$\frac{\partial}{\partial x}\left[-\frac{x}{\gamma}\rho^*(x)\right] = \frac{\sigma^2}{2}\frac{\partial^2 \rho^*(x)}{\partial x^2} \tag{18}$$

On réalise que les dérivées partielles sont en fait des dérivées totales. De plus, elles peuvent être simplifiées et on obtient alors :

$$-\frac{x}{\gamma}\rho^*(x) = \frac{\sigma^2}{2}\frac{\mathrm{d}\rho^*(x)}{\mathrm{d}x} \tag{19}$$

qui se résout facilement :

$$\frac{\mathrm{d}\rho^*(x)}{\rho^*(x)} = -\frac{2x}{\gamma\sigma^2}\mathrm{d}x \quad \Longrightarrow \quad \ln[\rho^*(x)] = -\frac{x^2}{\gamma\sigma^2} + \text{constante} \quad \Longrightarrow \quad \rho^*(x) = A\exp\left[-\frac{x^2}{\gamma\sigma^2}\right]$$

Puis, si on exprime $\rho^*(x)$ en fonction de V, plutôt que x, et on obtient :

$$\rho^*(V) = C\exp\left[-\frac{(V-\alpha\gamma)^2}{\gamma\sigma^2}\right] \tag{20}$$

où A est une constante quelconque. On reconnaît en (20) l'équation de la Gaussienne dont la moyenne, $\tilde{y}$, et la variance, $\tilde{\sigma}^2$, sont telles que :

$$\text{Gaussienne}(y) = A\exp\left[-\frac{(y-\tilde{y})^2}{2\tilde{\sigma}^2}\right] \tag{21}$$

De cette façon, en reliant la moyenne et la variance de la Gaussienne en (21) aux variables de (20), on peut obtenir, pour (20), la moyenne

$$\tilde{y} \;=\; \gamma\alpha \;=\; \kappa C R\left[\tfrac{I}{\kappa C} + \tfrac{a_e\mu_e}{\kappa C}\right] \;=\; R(I + a_e\mu_e)$$

et la variance

$$\tilde{\sigma}^2 \;=\; \tfrac{\gamma\sigma^2}{2} = \tfrac{\kappa C R}{2}\tfrac{a_e^2\mu_e}{\kappa^2 C^2} \;=\; \tfrac{R}{2\kappa C}a_e^2\mu_e$$

La moyenne, (3.3), et la variance, (3.3), sont à même d'expliquer les relations observées entre la `fmi`, κ, et la `fmo`. Tout d'abord, on constate que la moyenne ne dépend pas de κ. Cependant, la variance diminue quand κ augmente. Donc, il y a effectivement moins de bruit quand κ augmente (la moyenne ne bouge pas), et ceci explique pourquoi les courbes des fmo vont vers zéro quand κ augmente. Car, à moins que le courant moyen soit nécessaire à faire décharger le système, une plus petite variance signifie que le courant d'entrée ne va que très rarement varier suffisamment pour induire un potentiel d'action, d'où une diminution de la `fmo`.

De plus, on a vu que la `fmo` augmente quand la `fmi` augmente. Ceci prédit que les courbes des `fmo` en fonction de κ auront une échelle plus élevée pour des `fmi` plus grandes. En effet, lorsque la `fmi` augmente (i.e. μ_e augmente), le potentiel moyen de même que la variance (taux de bruit) augmentent. Par conséquent, il est normal d'observer alors une augmentation de la `fmo`. Physiquement, pour une `fmi` plus grande, un plus grand nombre d'impulsions d'entrée vont se superposer (car l'intervalle de temps moyen entre les impulsions est plus petit), augmentant ainsi le courant moyen d'entrée et sa variance. La `fmo` ne pourra faire autrement que d'augmenter aussi.

Si on décide de voir la variable κ comme un changement de l'échelle caractéristique de temps de la dynamique du système (étant donné que $1/\kappa$ multiplie $\mathrm{d}V/\mathrm{d}t$), on ne s'étonne pas de voir que la `fmo` augmente lorsque κ diminue. En effet, pour un κ plus petit, $\mathrm{d}V/\mathrm{d}t$ est plus grand et ainsi la dynamique du système se trouve accélérée, d'où une `fmo` plus élevée.

Finalement, les résultats présentés par DeFelice, dans ses articles [1, 8], ont été obtenus à partir de l'étude de la dynamique du modèle de Hodgkin-Huxley, un modèle qui considère chaque canal individuellement (Single-Channel model). Bien que ses résultats soient en bien des points similaires aux nôtres, nous ne sommes pas parvenus à obtenir de la courbe qu'elle redescende pour un κ très petit (dans les articles de DeFelice, la variable κ est désignée comme étant l'aire). La forme des courbes, obtenues par DeFelice, pour de très petits κ, résulte des fluctuations stochastiques qui surviennent lorsque seulement un très petit nombre de canaux ioniques s'ouvrant et se fermant est considéré. Dans cette limite, l'approximation statistique faite par des modèles comme celui de Morris-Lecar, où la dynamique considérée représente une approximation statistique de la dynamique individuelle de chaque canal, n'est plus valide. Ainsi, c'est la nature du modèle choisi qui nous empêche d'observer la forme prédite ou même de considérer des quantités de canaux trop petites.

4 Discussion

Les résultats obtenus sont en accord avec ce que la théorie prévoit (voir [1], [4], [8] et, ci-dessus, Section 3). La `fmo` se comporte tel qu'attendu en fonction de la `fmi` et κ. Il serait maintenant intéressant de faire varier d'autres paramètres du modèle et voir de quelle façon ils influencent la dynamique du système.

Au cours du projet, nous avons implémenté diverses formes de courant d'entrée, I. Afin de vérifier la validité de notre modèle, nous avions modélisé un courant d'entrée constant ainsi que des impulsions de courant carrées. Par la suite, nous avons utilisé un courant d'entrée de forme alpha d'amplitude et de paramètre alpha constants. Ces impulsions de forme alpha arrivaient à notre neurone à des temps aléatoires de distribution exponentielle.

Mais, de toutes ces implémentations, aucune ne tient compte la dépendance, pourtant bien réelle, du courant d'entrée sur le potentiel de membrane, V. Le courant d'entrée représente le courant produit par les potentiels d'action des neurones voisins. La réponse active des neurones voisins dépend également du courant d'entrée qui leur parvient. Pour être absolument précis, il faudrait intégrer la dynamique de tout un système de neurones ; ceci serait trop complexe pour ce projet. En revanche, approximer le courant d'entrée par un courant de base et des impulsions aléatoires, donne des résultats satisfaisants. Mais ajouter un courant

d'entrée synaptique de forme

$$I = I_0 + g_{\text{syn}}(V - V_{\text{syn}})$$

, où V_{syn} est une constante serait physiquement plus juste. Il serait intéressant d'implémenter cette situation et d'observer la dynamique qui découlerait de cette modification.

Il serait également intéressant de voir l'influence qu'aurait l'ajout de bruit aux conductances des canaux ioniques. Ceci a été fait par Fox pour le modèle de Hodgkin-Huxley [3]. Dans son article, il utilise un bruit blanc gaussien dont il définit la moyenne et la variance. Il s'agirait donc d'adapter la moyenne et la variance suggérés pour ce bruit, en fonction des variables que l'on retrouve dans le modèle de Morris-Lecar.

5 Conclusion

Le comportement de la fréquence moyenne de sortie, i.e. la fréquence moyenne à laquelle surviennent les potentiels d'action, est entièrement expliqué par la théorie (voir ci-dessus Section 3.3). Le modèle utilisé et la dynamique simulée nous ont permis de constater que la fréquence moyenne à laquelle surviennent les potentiels d'action augmente quand :

1. la fréquence moyenne d'arrivée des impulsions d'entrée de courant augmente ;
2. la quantité de canaux augmente (la variable κ diminue).

En effet, une fréquence moyenne plus élevée d'arrivée des impulsions d'entrée de courant peut être perçue comme une augmentation d'un courant de base moyen auquel un bruit est ajouté. Lorsque l'on augmente la fréquence d'entrée, on augmente la valeur moyenne du courant de base, ainsi que sa variance et de cette façon, le bruit est plus souvent susceptible de franchir le seuil, ce qui entraîne une hausse de la fréquence moyenne de sortie.

De la même façon, une augmentation de la densité de canaux permet au système d'absorber une plus grande proportion du courant généré par les neurones voisins. Ainsi, pour un même courant de base, la variance du courant ressenti par le neurone est plus grande et ceci entraînera aussi une hausse de la fréquence moyenne de sortie.

Références

[1] Louis J. DeFelice and William N. Goolsby. Order from randomness : Spontaneous firing from stochastic properties of ion channels. In Mark Millonas, editor, *Fluctuations and Order : The New Synthesis*, Institute for Nonlinear Science, chapter 21. Springer, 1996.

[2] William H. Press et al. *Numerical Recipes in Fortran, 2nd Edition*. Cambridge University Press, 1992.

[3] Ronald F. Fox. Stochastic versions of the hodgkin-huxley equations. *Biophysical Journal*, 72 :2068–2074, 1990.

[4] Christof Koch. *Biophysics of Computation : Information Processing in Single Neurons*. Oxford University Press, New York, 1999.

[5] Irwin B. Levitan and Leonard K. Kaczmarek. *The Neuron : cell and molecular biology 2nd Edition*. Oxford University Press, New York, 1997.

[6] John Rinzel. Discussion : Electrical excitability of cells, theory and experiment : Review of the hodgkin-huxley foundation and an update. *Bulletin of Mathematical Biology*, 52 no 1/2 :5–23, 1990.

[7] John Rinzel and Bard Ermentrout. Analysis of neural excitability and oscillations. In Christof Koch and Idan Segev, editors, *Methods in Neuronal Modeling : From Ions to Networks, 2nd Edition*, chapter 7. MIT Press, Cambridge, MA, 1998.

[8] Adam F. Strassberg and Louis J. DeFelice. Limitations of the hodgkin-huxley formalism : Effects of single channel kinetics on transmembrane voltage dynamics. *Neural Computation 5*, 23 :843–855, 1993.

A Code de neuron.f

```
c Catherine Beauchemin
c 1638044
c PHY4005: Projet en neurophysique
c 7 novembre 2000
c Pour: André Longtin
c ---------------------------------------------------------------------


program Neuron

c Declaration des variables
implicit none
integer nvar,model,k
real*8 min,max,dt,t
real*8 y(5),dy(5),I,okw


c Ouverture du fichier d'écriture (sortie)
open(9,file='neuron.dat',status='unknown')


c formats d'écriture pour les variables
100 format(a6,3(4x,a3))
101 format( e12.6,3(2x,e16.10) )



c Initialisation des variables
min = 0.d0
max = 8.0d2
dt = 0.05d0
nvar = 2
model = 2
y(1) = -30.0d0
y(2) = 0.002d0
I = 40.76d0
okw = 0.05d0/dt
k = okw


c Initialisation du fichier d'écriture
write(9,100) 't','V','w','I'
write(9,*) '-----------------------------------------------'

write(*,*) 'With Vo=',y(1),' Wo=',y(2),' and Io=',I

do 31 t=min,max,dt
   if(k.eq.okw) then
k = 0
```

```
write(9,101) t,y(1),y(2),I
   endif
   call defineI(t,I)
   call ml(y,dy,model,I)
   call euler(y,dy,dt,nvar)
   k = k + 1
31 continue


close(9)

Stop 'Resultats dans neuron.dat'
End
c
c
c ----------------FIN DU PROGRAMME PRINCIPAL-------------------

c Les Équations de Morris-Lecar
subroutine ml(y,dy,model,I)
implicit none
integer model
real*8 M,W,Tau,y(5),dy(5)
real*8 v1,v2,v3,v4
real*8 gCa,gK,gL,vCa,vK,vL,I,C,phi

v1 = -1.2d0
v2 = 18.d0
gK = 8.d0
gL = 2.d0
vK = -84.d0
vL = -60.d0
vCa = 120.d0
C = 20.d0

if(model.eq.1) then
   v3 = 2.d0
   v4 = 30.d0
   gCa = 4.4d0
   phi = 0.04d0
else
   v3 = 12.d0
   v4 = 17.4d0
   gCa = 4.d0
   phi = 1.d0/15.d0
c    phi = 0.23d0
endif

M = 0.5d0 * ( 1.d0 + dtanh( (y(1)-v1) /v2) )
W = 0.5d0 * ( 1.d0 + dtanh( (y(1)-v3) /v4) )
Tau = 1.d0 / dcosh( (y(1)-v3) / (2*v4) )
```

```
dy(1) = ( (-gCa*M*(y(1)-vCa))-(gK*y(2)*(y(1)-vK))
      1 -(gL*(y(1)-vL)) + I ) / C

dy(2) = phi * ( W - y(2) ) / Tau

Return
End
c
c -------------------------------------------------------------
c
c Méthode d'Euler
subroutine euler(y,dy,dt,nvar)
implicit none
integer n,nvar
real*8 y(5),dy(5),dt

do 55 n=1,nvar
   y(n) = y(n) + (dt * dy(n))
55 continue

Return
End
c
c -------------------------------------------------------------

c Définir le courant I
subroutine defineI(t,I)
implicit none
real*8 t,I

c Pour figure 7.1, 7.4, 7.5
I = I


c Pour figure 7.3
c if(((t.ge.100.d0).and.(t.le.105.d0)).or.
c      1    ((t.ge.470.d0).and.(t.le.475.d0))) then
c I = 90.d0+30.d0
c else
c I = 90.d0
c endif


c Pour figure 7.8
c if((t.ge.100.d0).and.(t.le.105.d0)) then
c I = 37.5d0
c elseif((t.ge.300.d0).and.(t.le.305.d0)) then
c I = 98.d0
c elseif((t.ge.430.d0).and.(t.le.435.d0)) then
c I = 40.d0
c else
```

```
c I = 37.5d0
c endif

Return
End
c
c ---------------------------------------------------------
```

B Code de npoiscon.f

```
c Catherine Beauchemin
c 1638044
c PHY4005: Projet en neurophysique
c 7 novembre 2000
c Pour: André Longtin
c ---------------------------------------------------------------------


program Neuron

c Declaration des variables
implicit none
integer nvar,model,k
real*8 min,max,dt,t
real*8 y(5),dy(5),I,okw


c Ouverture du fichier d'écriture (sortie)
open(9,file='neuron.dat',status='unknown')


c formats d'écriture pour les variables
100 format(a6,3(4x,a3))
101 format( e12.6,3(2x,e16.10) )



c Initialisation des variables
min = 0.d0
max = 8.0d2
dt = 0.05d0
nvar = 2
model = 2
y(1) = -30.0d0
y(2) = 0.002d0
I = 40.76d0
okw = 0.05d0/dt
k = okw
```

```
c Initialisation du fichier d'écriture
write(9,100) 't','V','w','I'
write(9,*) '-----------------------------------------------'

write(*,*) 'With Vo=',y(1),' Wo=',y(2),' and Io=',I

do 31 t=min,max,dt
   if(k.eq.okw) then
k = 0
write(9,101) t,y(1),y(2),I
   endif
   call defineI(t,I)
   call ml(y,dy,model,I)
   call euler(y,dy,dt,nvar)
   k = k + 1
31 continue


close(9)

Stop 'Resultats dans neuron.dat'
End
c
c
c ----------------FIN DU PROGRAMME PRINCIPAL-------------------

c Les Équations de Morris-Lecar
subroutine ml(y,dy,model,I)
implicit none
integer model
real*8 M,W,Tau,y(5),dy(5)
real*8 v1,v2,v3,v4
real*8 gCa,gK,gL,vCa,vK,vL,I,C,phi

v1 = -1.2d0
v2 = 18.d0
gK = 8.d0
gL = 2.d0
vK = -84.d0
vL = -60.d0
vCa = 120.d0
C = 20.d0

if(model.eq.1) then
   v3 = 2.d0
   v4 = 30.d0
   gCa = 4.4d0
   phi = 0.04d0
else
```

```
   v3 = 12.d0
   v4 = 17.4d0
   gCa = 4.d0
   phi = 1.d0/15.d0
c    phi = 0.23d0
endif

M = 0.5d0 * ( 1.d0 + dtanh( (y(1)-v1) /v2) )
W = 0.5d0 * ( 1.d0 + dtanh( (y(1)-v3) /v4) )
Tau = 1.d0 / dcosh( (y(1)-v3) / (2*v4) )

dy(1) = ( (-gCa*M*(y(1)-vCa))-(gK*y(2)*(y(1)-vK))
      1 -(gL*(y(1)-vL)) + I ) / C

dy(2) = phi * ( W - y(2) ) / Tau

Return
End
c
c ---------------------------------------------------------------
c
c Méthode d'Euler
subroutine euler(y,dy,dt,nvar)
implicit none
integer n,nvar
real*8 y(5),dy(5),dt

do 55 n=1,nvar
   y(n) = y(n) + (dt * dy(n))
55 continue

Return
End
c
c ---------------------------------------------------------------

c Définir le courant I
subroutine defineI(t,I)
implicit none
real*8 t,I

c Pour figure 7.1, 7.4, 7.5
I = I


c Pour figure 7.3
c if(((t.ge.100.d0).and.(t.le.105.d0)).or.
c      1    ((t.ge.470.d0).and.(t.le.475.d0))) then
c I = 90.d0+30.d0
c else
c I = 90.d0
```

```
c endif


c Pour figure 7.8
c if((t.ge.100.d0).and.(t.le.105.d0)) then
c I = 37.5d0
c elseif((t.ge.300.d0).and.(t.le.305.d0)) then
c I = 98.d0
c elseif((t.ge.430.d0).and.(t.le.435.d0)) then
c I = 40.d0
c else
c I = 37.5d0
c endif

Return
End
c
c -------------------------------------------------------------
```

C Code de naire.f

```
c Catherine Beauchemin
c 1638044
c PHY4005: Projet en neurophysique
c 7 novembre 2000
c Pour: André Longtin
c ---------------------------------------------------------------------


program Neuron

c Declaration des variables
implicit none
integer nvar,model,k
real*8 min,max,dt,t
real*8 y(5),dy(5),I,okw


c Ouverture du fichier d'écriture (sortie)
open(9,file='neuron.dat',status='unknown')


c formats d'écriture pour les variables
100 format(a6,3(4x,a3))
101 format( e12.6,3(2x,e16.10) )
```

```
c Initialisation des variables
min = 0.d0
max = 8.0d2
dt = 0.05d0
nvar = 2
model = 2
y(1) = -30.0d0
y(2) = 0.002d0
I = 40.76d0
okw = 0.05d0/dt
k = okw


c Initialisation du fichier d'écriture
write(9,100) 't','V','w','I'
write(9,*) '-----------------------------------------------'

write(*,*) 'With Vo=',y(1),' Wo=',y(2),' and Io=',I

do 31 t=min,max,dt
   if(k.eq.okw) then
k = 0
write(9,101) t,y(1),y(2),I
   endif
   call defineI(t,I)
   call ml(y,dy,model,I)
   call euler(y,dy,dt,nvar)
   k = k + 1
31 continue


close(9)

Stop 'Resultats dans neuron.dat'
End
c
c
c ----------------FIN DU PROGRAMME PRINCIPAL-------------------

c Les Équations de Morris-Lecar
subroutine ml(y,dy,model,I)
implicit none
integer model
real*8 M,W,Tau,y(5),dy(5)
real*8 v1,v2,v3,v4
real*8 gCa,gK,gL,vCa,vK,vL,I,C,phi

v1 = -1.2d0
v2 = 18.d0
gK = 8.d0
gL = 2.d0
```

```
vK = -84.d0
vL = -60.d0
vCa = 120.d0
C = 20.d0

if(model.eq.1) then
   v3 = 2.d0
   v4 = 30.d0
   gCa = 4.4d0
   phi = 0.04d0
else
   v3 = 12.d0
   v4 = 17.4d0
   gCa = 4.d0
   phi = 1.d0/15.d0
c    phi = 0.23d0
endif

M = 0.5d0 * ( 1.d0 + dtanh( (y(1)-v1) /v2) )
W = 0.5d0 * ( 1.d0 + dtanh( (y(1)-v3) /v4) )
Tau = 1.d0 / dcosh( (y(1)-v3) / (2*v4) )

dy(1) = ( (-gCa*M*(y(1)-vCa))-(gK*y(2)*(y(1)-vK))
      1 -(gL*(y(1)-vL)) + I ) / C

dy(2) = phi * ( W - y(2) ) / Tau

Return
End
c
c -------------------------------------------------------------
c
c Méthode d'Euler
subroutine euler(y,dy,dt,nvar)
implicit none
integer n,nvar
real*8 y(5),dy(5),dt

do 55 n=1,nvar
   y(n) = y(n) + (dt * dy(n))
55 continue

Return
End
c
c -------------------------------------------------------------

c Définir le courant I
subroutine defineI(t,I)
implicit none
real*8 t,I
```

```
c Pour figure 7.1, 7.4, 7.5
I = I


c Pour figure 7.3
c if(((t.ge.100.d0).and.(t.le.105.d0)).or.
c      1    ((t.ge.470.d0).and.(t.le.475.d0))) then
c I = 90.d0+30.d0
c else
c I = 90.d0
c endif


c Pour figure 7.8
c if((t.ge.100.d0).and.(t.le.105.d0)) then
c I = 37.5d0
c elseif((t.ge.300.d0).and.(t.le.305.d0)) then
c I = 98.d0
c elseif((t.ge.430.d0).and.(t.le.435.d0)) then
c I = 40.d0
c else
c I = 37.5d0
c endif

Return
End
c
c -------------------------------------------------------------
```