

# Mémoires à court terme

Jérôme Casse et Silvain Rideau

26 mai 2009

## 1 Problématique

Les expériences ont montré que l'homme pouvait retenir à peu près 7 données sur une courte période. Les études physiologiques semblent dire que ces informations sont codées dans l'activité répétée de certains motifs de neurones.

La question est donc de savoir quel mécanisme neuronal permet la mise en place et le maintien de l'activité de ces motifs de neurones. La réponse apportée ici (proposée par Lisman et Idiart en 1995) se base sur divers phénomènes observés dans l'hippocampe : superposition de courtes oscillations (40 Hz) et d'oscillations plus lentes (6 Hz) et la présence de neurones sensibilisés après un potentiel d'action.

## 2 Modèle

Le modèle repose sur trois points importants. Le premier est les neurones ADP (after-depolarisation), le second est l'utilisation d'un potentiel oscillant et le troisième est l'inhibition du réseau entier de neurones .

Les neurones ADP sont des neurones qui, lorsqu'ils émettent un potentiel d'action, au lieu de s'hyperpolariser se dépolarisent (voir figure 1), et donc deviennent plus sensibles pour un temps. Si l'on sent bien que cette "mémoire" du neurone peut permettre la mémorisation de données, cette dépolarisation qui dure environ 200 ms ne peut clairement pas être le seul phénomène responsable de la mémorisation à courte durée.

Le potentiel oscillant permet de faciliter l'émission des potentiels d'action mémorisés ainsi que l'entrée des nouvelles mémoires. En effet, le temps que met l'ADP à atteindre son maximum est à peu près le même que celui de la période du potentiel oscillant. Si le neurone avait tiré au sommet de l'oscillation précédente, après une période, il tirera de nouveau, la somme du potentiel oscillant et de l'ADP le faisant passer au dessus de son seuil (en fait quelle que soit la phase de l'excitation, les potentiels suivant se synchroniseront avec l'entrée voir figure 2). De plus, ce potentiel forme l'onde à basse fréquence et les informations sont stockées au sommet de cette onde. Le couplage de ces deux mécanismes permet donc au neurone de mémoriser une excitation.

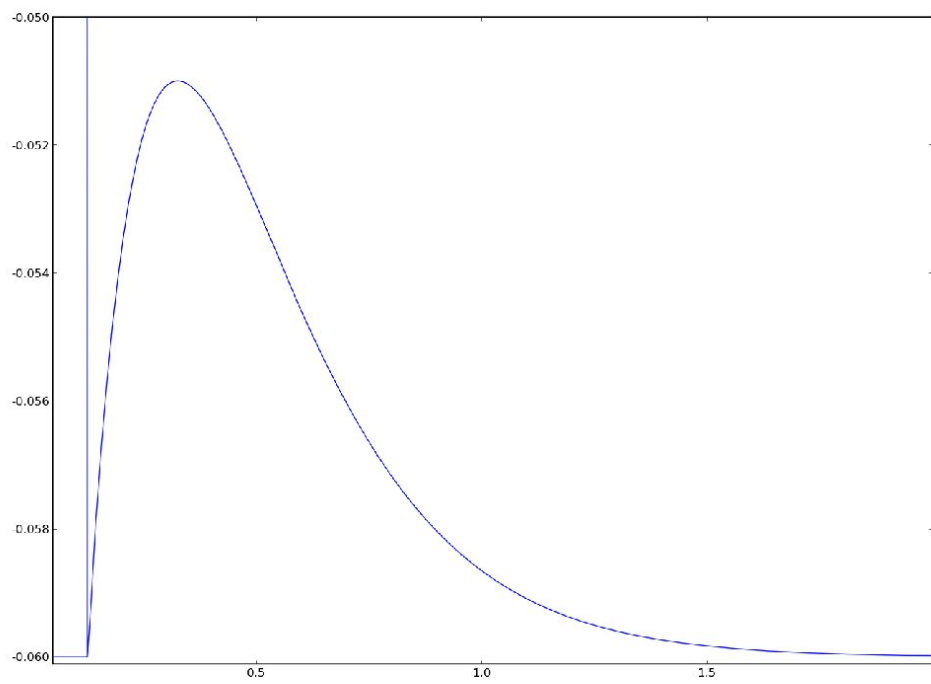


FIG. 1 – La dépolarisation d'un neurone ADP

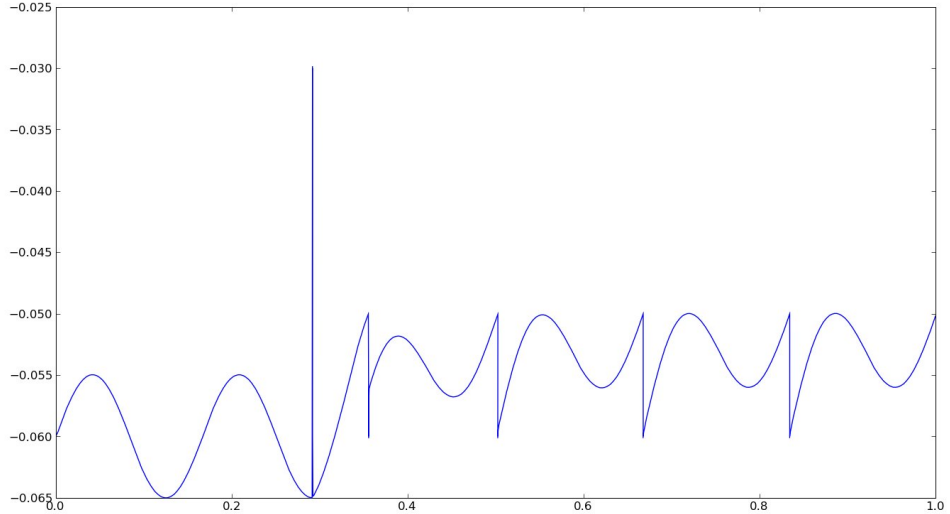


FIG. 2 – Un neurone à ADP soumis à une excitation

L'inhibition du réseau de neurones permet quant à elle de séparer les activités des neurones et ainsi éviter qu'ils se synchronisent. C'est donc cette dernière qui en créant une période d'environ 5 ms pendant laquelle les neurones sont moins excitables permet la création des sous-cycles au sommet de l'oscillation lente, sous-cycles qui contiendront chacun une donnée. En l'absence d'inhibition, les neurones actifs se synchronisent tous et on ne peut plus différencier les différents motifs. Lorsque l'inhibition est présente, l'activation du premier groupe de neurone, celui qui a tiré au premier sous-cycle du cycle précédent, empêche l'activation des autres qui ne pourront tirer qu'au plus tôt 5 ms secondes plus tard, période à laquelle les neurones les plus excitables sont ceux qui ont tiré au la deuxième sous-cycle. Leur activation inhibera celle des autres etc... (voire figure 11)

Le réseau de notre modèle est donc constitué d'une population de neurones intègre-et-tire à ADP qui stockent l'information, et d'un neurone inhibiteur qui reçoit les sorties de tous les neurones et est connecté à tous les neurones.

Les neurones principaux ont pour équation :

$$\frac{dV}{dt} = \frac{1}{\tau}(V + V_{osc} + V_{inh} + V_0 + V_{ADP})$$

Ils voient leurs potentiels osciller grâce à  $V_{osc} = B * \sin(2 * \pi * f * t)$  ce qui correspond dans le modèle de l'article à l'entrée oscillante que reçoit chaque neurone.

Le potentiel ADP de ces neurones qui était une fonction alpha à été modélisé

par ces 2 lignes d'équations :

$$\begin{cases} \frac{dV_{ADP}}{dt} = \frac{1}{\tau_{ADP}}(x_{ADP} - V_{ADP}) \\ \frac{dx_{ADP}}{dt} = -\frac{x_{ADP}}{\tau_{ADP}} \end{cases}$$

Elles permettent d'obtenir la fonction alpha souhaitée via l'initialisation du paramètre  $x_{ADP}$  à  $A_{ADP} * e$ .

L'inhibition des neurones a été modélisée de la même façon avec les variables indexées par  $inh$ . Cependant, si on ne fait que relier tous les neurones au neurone inhibiteur, quand un motif de neurones tire, le neurone inhibiteur peut déclencher plus d'une inhibition (car les potentiels d'action ne sont pas exactement synchronisés) pour un motif, ce qui provoque une désensibilisation plus longue des autres neurones et donc un dysfonctionnement du réseau qui stocke alors moins de données par cycle. La solution consiste à mettre une période réfractaire de 2,5 ms au neurone inhibiteur qui ne déclenche alors un potentiel alpha dans les neurones du réseau qu'une fois par sous-cycle.

Nous avons fait le choix de ne pas incorporer le potentiel de l'ADP dans une diminution de seuil afin de mieux observer les potentiels d'action émis lors des phases de rappel des éléments mémorisés.

### 3 Résultats et discussions

Tout d'abord, nous devons signaler que pour faire marcher le modèle, nous avons du, d'une part ajouter un tau infime pour le potentiel et non le laisser comme la somme des potentiels (ie on a redéfini l'approximation de l'article) et nous avons du baisser le pas d'intégration de brian à 0.02\*ms pour effectivement réussir à stocker 7 mémoires. Du fait, de ces déboires, nous avons pu observer quantité de phénomènes.

#### 3.1 Résultat du modèle de l'article

Les résultats de la simulation d'une population de 1000 neurones pendant 20 secondes au cours de laquelle 20 entrées ont lieu, sont présentés sur les figures 3, 4 et 5.

La première remarque à faire est que les temps d'excitation et le motif excités sont aléatoire, avec des motifs d'environ 1% de la population, ce qui permet de supposer raisonnablement que 2 motifs ne se superposent pas.

La seconde concerne les 5 premières secondes dont le résultat n'est pas affiché, cela est simplement dû au fait qu'il faut commencer par créer les sous-cycles dans le réseau (et donc mettre des motifs en place dans le réseau) avant de pouvoir vraiment l'utiliser. De plus avant que les 7 sous-cycles soient mis en place certains phénomènes bizarres ont lieu (oubli du dernier souvenir rentré quand un nouveau arrive, nouveau souvenir qui se met en fin de train au lieu de se mettre au début entre autre, voir figure 7) qui font qu'on a préféré ne pas enregistrer

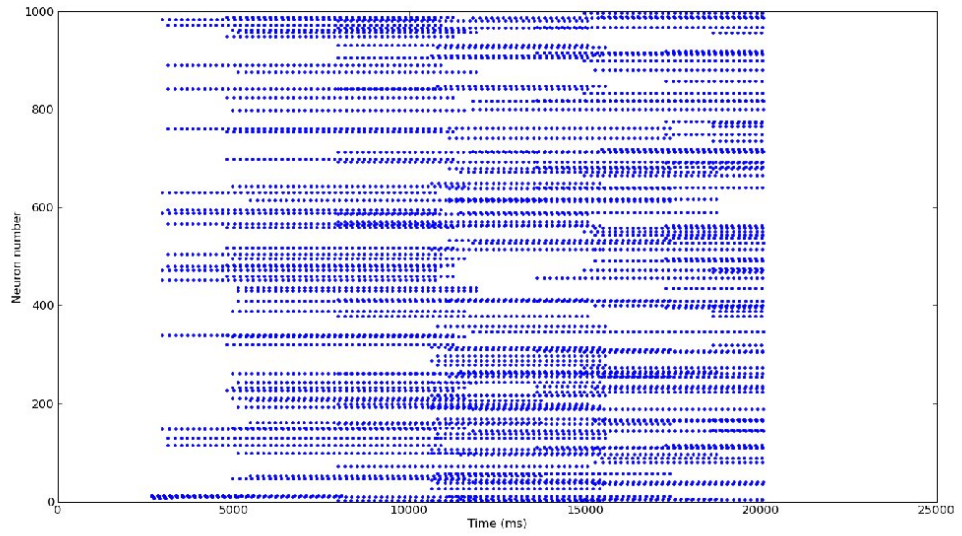


FIG. 3 – La simulation

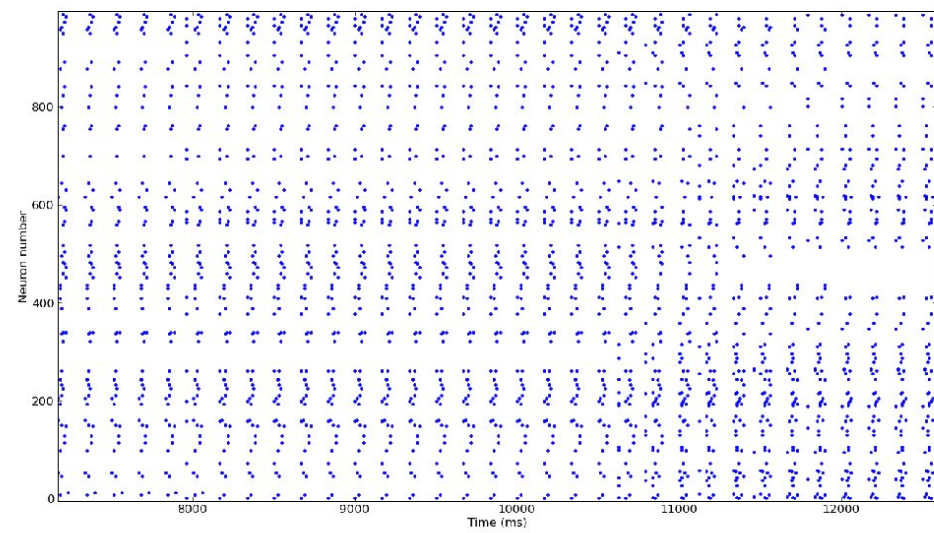


FIG. 4 – Un premier zoom

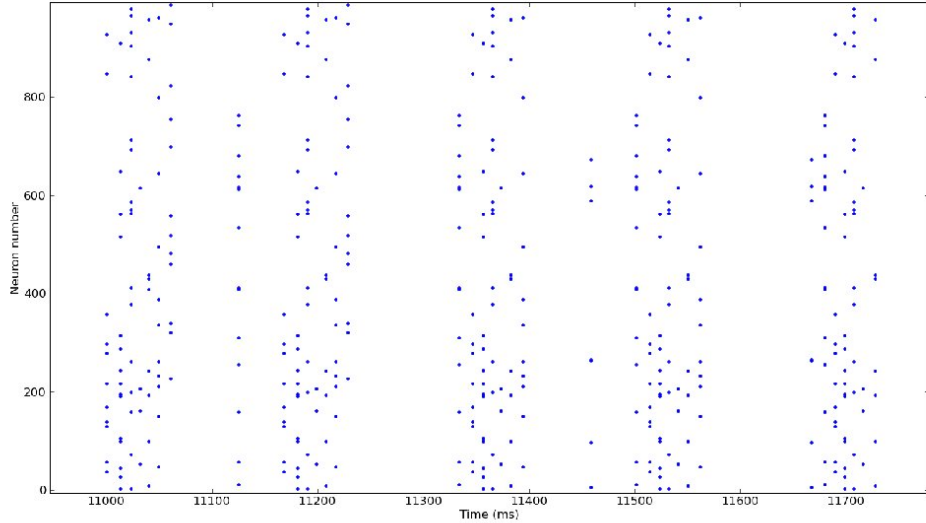


FIG. 5 – Encore plus zoomé

cette période à la fin de laquelle les neurones 7 à 13 tirent chacun à leur tour et que les 7 sous-cycles créés par l'inhibition sont présents dans le potentiel de chaque neurone (la figure 11 montre les potentiels membranaires de neurones du réseau avec une entrée sur le 10<sup>e</sup> neurone à la fin du 2<sup>e</sup> cycle affiché, on voit bien les 7 inhibition qui créent 7 moments du sommet de l'oscillation ou le neurone peut tirer).

La troisième remarque qui est développée plus tard est que l'entrée d'une nouvelle donnée ne peut pas avoir lieu n'importe quand (fait souligné dans l'article) on a fait ici le choix de ne faire rentrer des données qu'au minimum de l'oscillation.

Le premier graphe est trop grand pour qu'on ne puisse voir grand chose si ce n'est que lorsqu'un neurone est excité il tire régulièrement jusqu'à ce qu'il s'arrête.

Le deuxième graphe permet déjà de voir les motifs groupés légèrement décalés dans le temps qui se répètent mais se décalent dans sur le sommet de l'oscillation au fur et à mesure que de nouvelles données rentrent jusqu'à disparaître (par exemple à 8 s, une nouvelle donnée arrive, les 7 anciens motifs s'activent chacun leur tour quand l'oscillation s'approche de son maximum puis au cycle suivant la nouvelle vient se mettre au 1<sup>er</sup> sous-cycle les autres se décalent et le 7<sup>e</sup> ancienne motif qui était composée d'un unique neurone (en bas) disparaît). Le phénomène se répète à 10,5 s à 11 s et à 11,5 s.

Le troisième graphe permet de voir encore plus précisément ce qui se passe,

on voit entre autre ce qui se passe si 2 motifs partagent 1 même neurone, en l'occurrence celui qui arrive juste après 11100 ms et le 4<sup>e</sup> motif déjà présent qui partagent 1 neurone vers le 400<sup>e</sup>. Le neurone en question disparaît du 4<sup>e</sup> motif qui devient le 5<sup>e</sup> et reste présent dans le motif qui devient le 1<sup>er</sup> de la série.

Le réseau de neurone remplit donc exactement son rôle, mémorisant les 7 dernière entrées qu'il a reçu, tant que les motifs ne se recourent pas (mais plus la population est grande, plus ce risque est faible).

### 3.2 Autres résultats

Tout d'abord, nous avons observé une corrélation entre la valeur de  $A_{ADP}$  et le nombre de mémoires possiblement stockables :

AADP	nombre de mémoires stockables
5.0*mV	0
5.1*mV	1
5.8*mV	1
5.9*mV	2
6.6*mV	2
6.7*mV	3
7.4*mV	3
7.5*mV	4
8.2*mV	4
8.3*mV	5
8.9*mV	5
9.0*mV	6
9.5*mV	6
9.6*mV	7
10.0*mV	7

Par la suite, les résultats sont obtenues avec comme valeur d'AADP 9mV car ceci permet d'économiser du temps de calcul du fait que l'on peut prendre le pas d'intégration normal de Brian.

De plus, nous avons observé que le temps d'entrées de l'information à stocker est crucial. En effet, si on ne rentre pas dans le creux, le modèle commence à faire n'importe quoi. Ainsi, en prenant une entrée de l'information toutes les 166ms au lieu de toutes les 167ms, nous avons pu observer que le modèle ne marchait plus. En effet, la présentation de ces deux séries de spikes le montre (figure 6 et 7).

La figure 6 correspond à une entrée toutes les 166 ms, on peut remarquer que les spikes se synchronisent et que la 14<sup>e</sup> entrée n'est pas mémorisée (cf figure 8). Alors que le graphe de droite qui correspond à une entrée toutes les 167ms est ce que l'on souhaite et attend de notre modélisation (aux phénomènes anormaux près soulignés au préalable qui ont lieux tant que le réseau ne contient pas 7 motifs activés).

Nous avons également essayé d'insérer du bruit ( $\sigma = 0.01\text{mV}$ , cf figure 9) afin de voir si cela renforcerait, comme dans certains autres mécanismes,

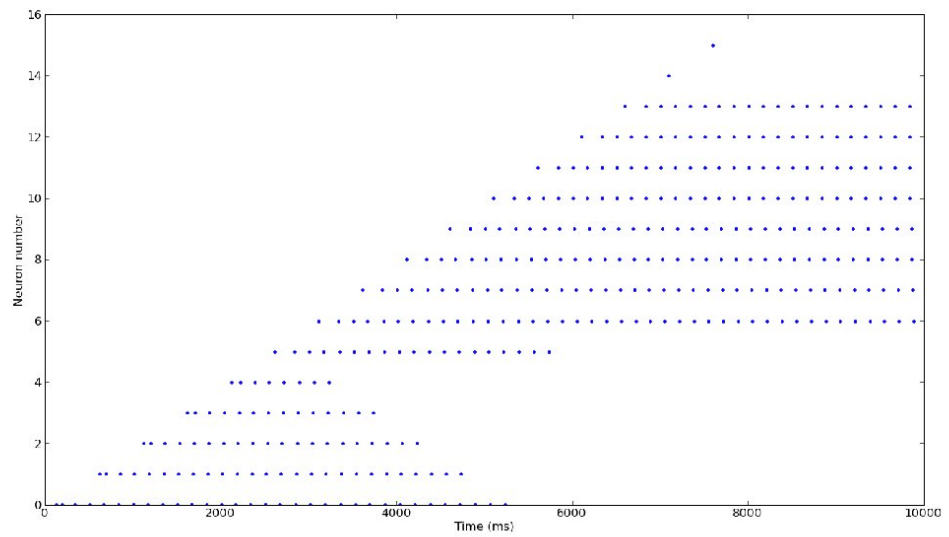


FIG. 6 – Les entrées se décalent en finissent proche du maximum, au milieu des motifs

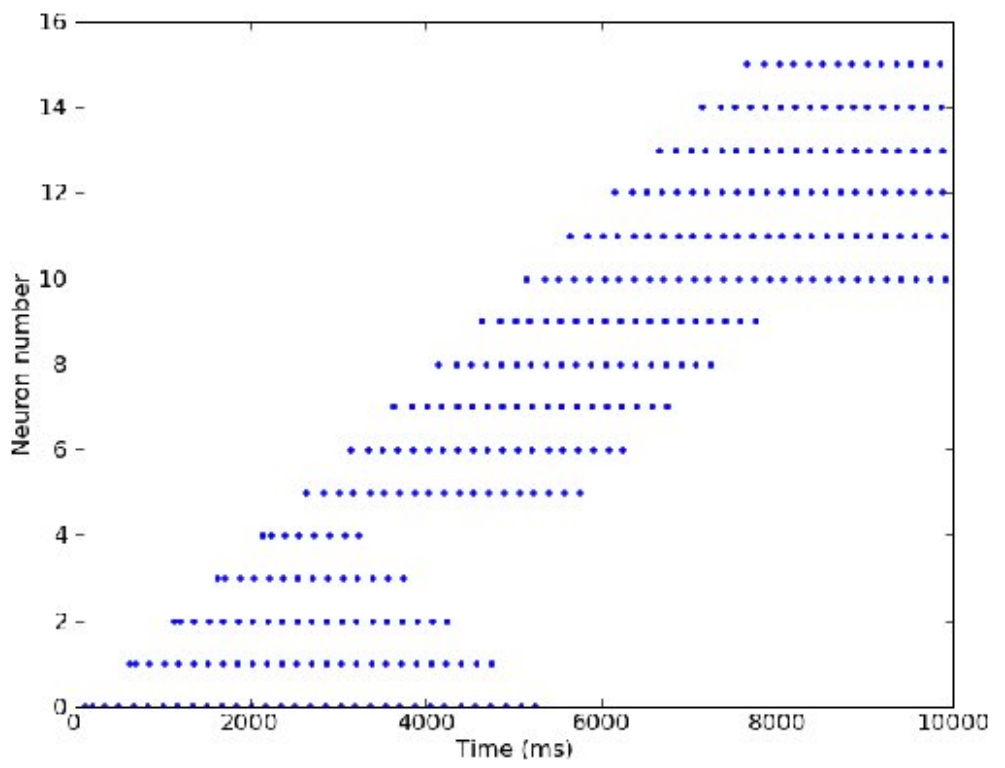


FIG. 7 – Entrées dans le creux

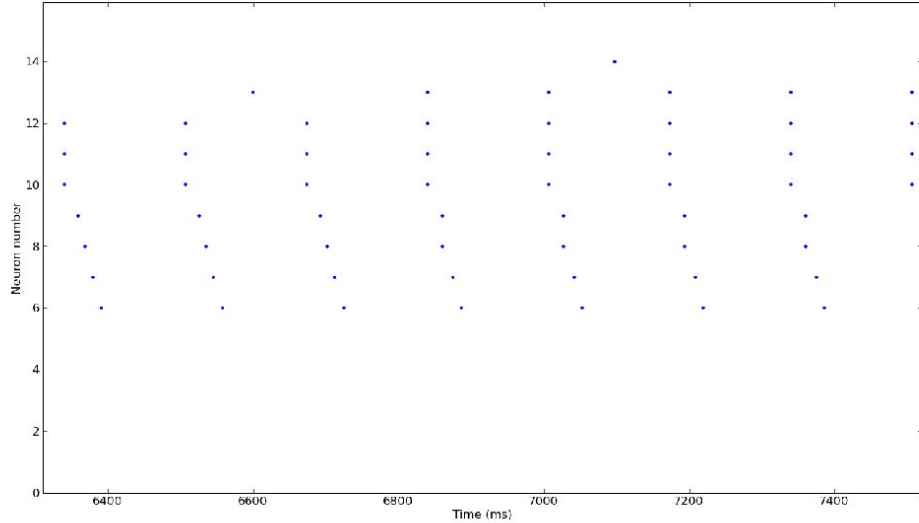


FIG. 8 – Zoom pour les arrivées toutes les 166ms

la précision. Le résultat est catastrophique. Même un bruit aussi faible fait complètement échouer le modèle, qui est bien trop sensible pour pouvoir supporter du bruit. En effet un neurone activé au sous-cycle  $n$ , au sous-cycle  $n-1$  du cycle suivant a quasiment atteint son seuil, un petit peu de bruit (ou une intégration numérique pas assez précise) le fait passer au dessus de son seuil et il tire au sous-cycle précédent... De la même façon, on peut voir le résultats ci-dessous.

Sur le graphique 10 qui correspond à l'introduction du bruit, on observe que les mémoires se superposent (par exemple le cyan et le rouge du bas) et que certaines ne disparaissent pas alors qu'elles le devraient (la rouge du bas) comme on peut le comparer avec le graphique de droite qui lui est le graphique obtenu pour une simulation sans bruit. Cependant la figure 9 montre bien à quel point le bruit est minime.

## 4 Discussion et perspective

Pour ce projet, nous n'avons eu le temps de ne regarder que le nombre de mémoires stockables en fonction de la valeur de  $A_{ADP}$  (au dixième près), mais il serait envisageable, d'une part, d'augmenter la précision de ces résultats et, d'autre part, de regarder l'influence d'autres paramètres. En effet les paramètres donnés pas l'article donnent un réseau qui fonctionne, alors que pour la plupart les changer un petit peu rend le réseau complètement déficient et pas juste pas

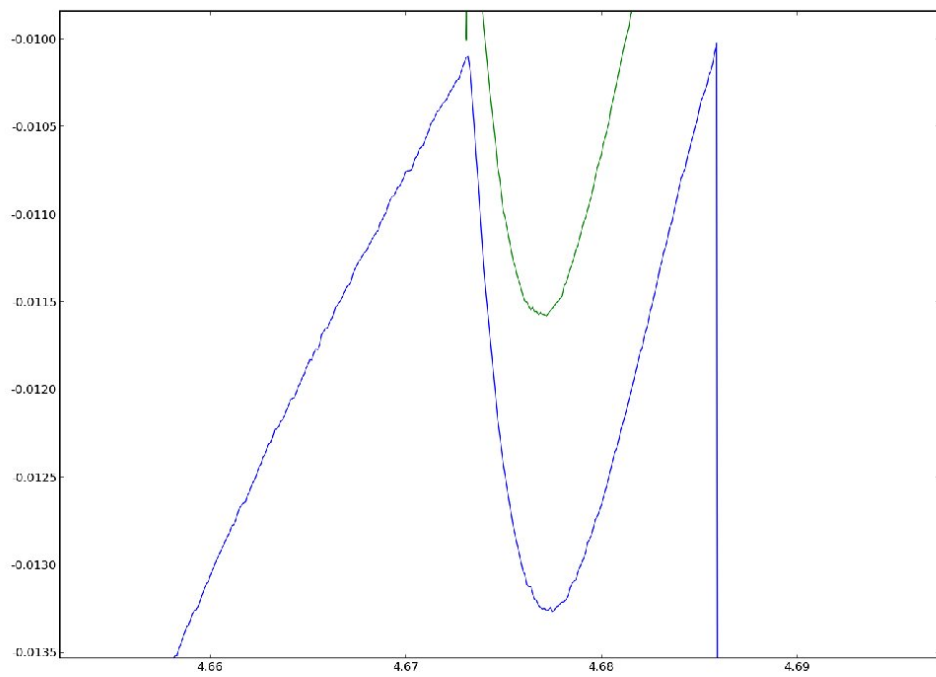


FIG. 9 – Le bruit correspond à la légère brisure le long de la courbe

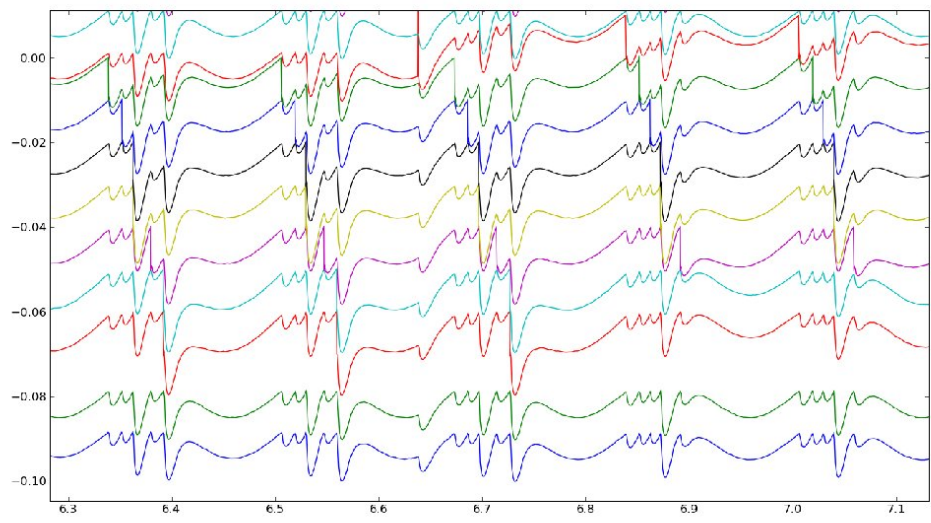


FIG. 10 – Potentiel de 11 neurones du réseau bruité

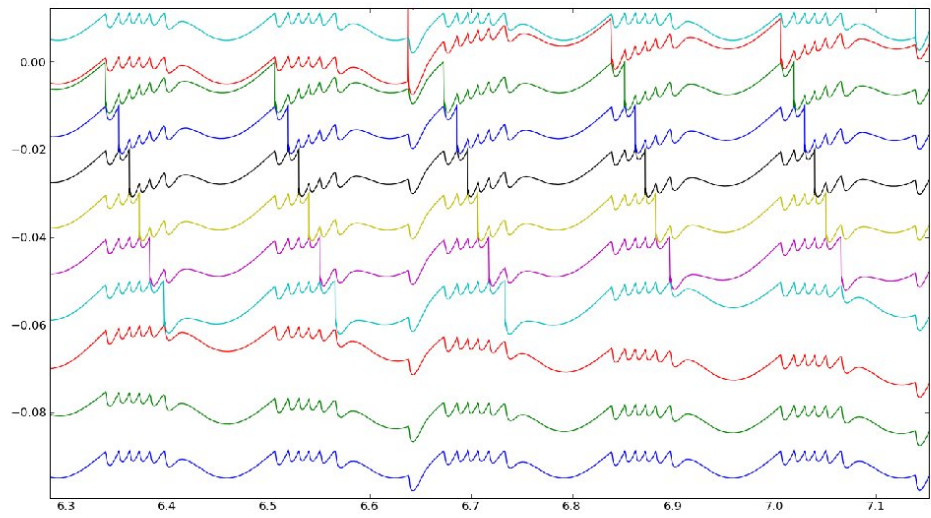


FIG. 11 – Potentiel de 11 neurones du réseau en fonctionnement normal

aussi efficace.

De plus il serait intéressant de comprendre comment palier à l'extrême sensibilité au bruit du réseau qui le rend en réalité inapte à représenter la réalité car un neurone in-vivo est en permanence soumis à du bruit.

## A neurone ADP solitaire (figure 2)

```
from brian import *

#Constante du modele
Vr = -60*mV
V0 = -60*mV
Vs = -50*mV
B = 5*mV
f = 6*Hz
AADP = 10*mV
tauADP = 200*ms
tau = 0.2*ms
w = 35*mV

#Equation du modele
##Neurone ADP et oscillation
eqs = '''
dv/dt = (-v + Vosc + V0 + VADP)/tau : mV
Vosc = B*sin(2*pi*f*t) : mV
dVADP/dt = (xADP - VADP)/tauADP : mV
dxADP/dt = -xADP/tauADP : mV
'''

resetADP = '''
VADP = 0
v = Vr
xADP = e*AADP
'''

#Les neurones
##qui retiennent l'information
G1 = NeuronGroup(N = 1,model = eqs,reset = resetADP ,threshold = Vs)

#Initialisation
##Entrees d'initialisation
spiketimes = [(0,7./(4*f))]
Ge = SpikeGeneratorGroup(1,spiketimes)

##Connection entrees d'initialisations
Ci = Connection(Ge,G1,'v')
Ci[0,0] = w
```

```
#On lance la simulation
##Conditions initiales
G1.v=[V0]
G1.VADP=[0*mV]
G1.xADP=[0*mV]

##Enregistrement des donnees
Mv = StateMonitor(G1,'v',record=True)
Ms = SpikeMonitor(G1)

##Simulation
run(1000*ms)

#Figures

figure(1)
plot (Mv.times,Mv[0])

show()
```

## B Code de la simulation à 1000 neurones

```
from brian import *

defaultclock.dt = 0.02*ms

#Parametres du modele
##Nombre de neurones

#Constante du modele
Vr = -60*mV
V0 = -60*mV
Vs = -50*mV
B = 5*mV
f = 6*Hz
AADP = 10*mV
tauADP = 200*ms
Ainh = -4*mV
tauinh = 5*ms
tau = 0.2*ms

w = 35*mV

#Equation du modele

##Neurone ADP et oscillation
eqs = '''
dv/dt = (-v + Vosc + Vinh + V0 + VADP)/tau : mV
Vosc = B*sin(2*pi*f*t) : mV
dVinh/dt = (xinh - Vinh)/tauinh : mV
dxinh/dt = -xinh/tauinh : mV
dVADP/dt = (xADP - VADP)/tauADP : mV
dxADP/dt = -xADP/tauADP : mV
'''

resetADP = '''
VADP = 0
v = Vr
xADP = e*AADP
'''

eqs_inh = '''
dx/dt = x/tau : 1
'''
```

```

#Les neurones

G1 = NeuronGroup(N = 1000,model = eqs,reset = resetADP ,threshold = Vs)

#inhibition
Ginh = NeuronGroup(N = 1, model = eqs_inh, reset = 0,
threshold = 1, refractory = tauinh/2)

##Entrees d'initialisation
nb_entrees = 14
spiketimes = []
for i in range(nb_entrees) :
    spiketimes = spiketimes + [(i,3./(4*f)+i/f)]

Gi = SpikeGeneratorGroup(nb_entrees,spiketimes)

#Connection entrees d'initialisations
Ci = Connection(Gi,G1,'v')
for i in range(nb_entrees) :
    Ci[i,i] = w

#Entrees
nb_ent = 20
duree = 100
discr_ent = randint(0,duree,nb_ent)
entrees = []
for i in range(nb_ent) :
    entrees = entrees+[(i,(71./(4*f))+((1/f)*discr_ent[i]))]
Ge = SpikeGeneratorGroup(nb_ent,entrees)

Ce = Connection (Ge,G1,'v',weight = w, sparseness = 0.01)

#Structure des connections

##Connection inhibitrice
Cinh1 = Connection(G1,Ginh,'x')
Cinh1.connect_full(G1,Ginh,1.1)
Cinh2 = Connection(Ginh,G1,'xinh')
Cinh2.connect_full(Ginh,G1,Ainh*e)

#On lance la simulation
#conditions initiales
G1.v=[V0]
G1.VADP=[0*mV]
G1.xADP=[0*mV]

```

```
#temps de simulation

run(16/f)
#Minh = StateMonitor(Ginh,'x',record=True)
#Mv = StateMonitor(G1,'v',record=True)
Ms = SpikeMonitor(G1)
run((duree+5)/f)

#Figures

'''
figure(1)
for i in range (20) :
    plot (Mv.times,Mv[i]+i*0.01)
'''

figure(2)
raster_plot()

'''
figure(3)
plot(Minh.times,Minh[0])
'''

show()
```