TP6 et TP7
Expériences et Neurones

1 Introduction : Expérience

Dans ce TP, nous considèrerons une expérience classique d'électrophysiologie dans laquelle on mesure la réponse d'un neurone à une stimulation électrique. La figure 1 montre le principe de cette expérience.

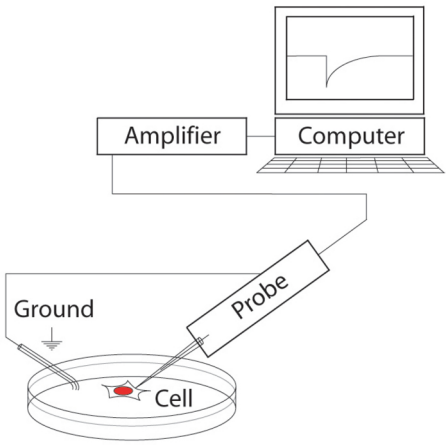

FIGURE 1 – Schéma du dispositif expérimental.

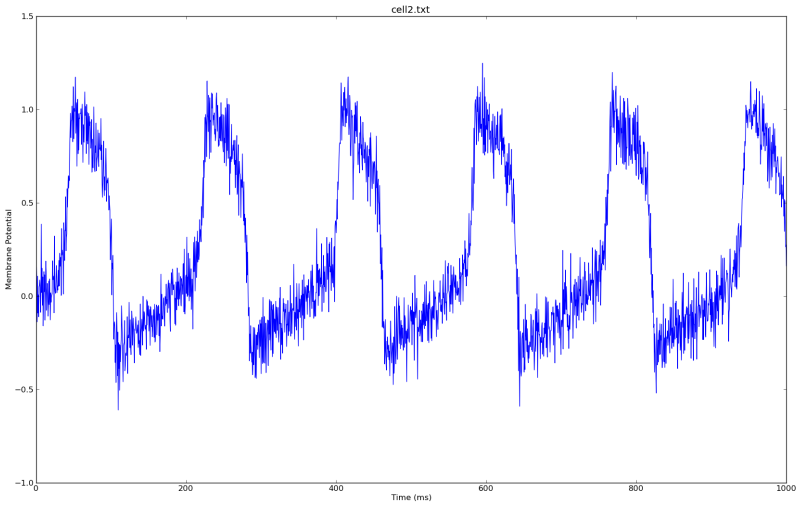

FIGURE 2 – Graphe des données expérimentales fournies dans le fichier *cell2.txt*.

Les mesures du potentiel électrique de la cellule sont sauvegardées dans un fichier texte ayant la structure suivante :

```
1: ExperimentalData-November
2: 3094
3: 0.0 0.0
4: 0.05 0.10
5: 0.10 0.12
6: ⋮
```

Ligne 1 : Commentaire, sans espace, maximum 40 charactères.
Ligne 2 : Nombres de points enregistrés.
Ligne 3 et suivantes : Un couple t_i v_i par ligne, t_i est le temps (en ms) et v_i la valeur enregistrée (en mV).

2 Travail TP6

1. La première étape est de simuler ces données à l'aide du modèle de FitzHugh-Nagumo (voir section A.1), ainsi que de les afficher. Vous considèrerez un maximum de 4096 points, tel que prévu dans la structure *trace* suivante :

```
typedef struct{
    char comment[40];
    int nbpts;
    float time[4096];
    float value[4096];
} trace;
```

   (a) Écrire la fonction *simuTrace(int tmax, float dt, float params[], trace* uneTrace)*. Cette fonction va intégrer les équations du modèle (voir section A.2). Elle remplit la structure *uneTrace*. Pour correspondre aux données expérimentales recueillies, on prendra $t_{max} = 1000$ms et $dt = 0.5$ms.

   (b) Écrire la fonction *printTrace(trace uneTrace)*, qui va afficher à l'écran le contenu de la structure de la manière suivante :

```
CommentaireDeLaTrace
t=0.0   v=0.0000
t=0.5   v=0.0000
t=1.0   v=0.0001
t=1.5   v=0.0003
...
```

   Afficher les données simulées à l'aide de *printTrace(trace uneTrace)*.

2. Vous allez ensuite comparer simulation et expérimentation. Vous utiliserez pour cela la fonction *expeTrace(trace* uneTrace)* qui remplit la structure *uneTrace* avec des données réelles issues d'une expérimentation. Cette fonction vous est fournie pour le TP6 (à récupérer depuis la plateforme Moodle).

   (a) Écrire la fonction *errorTrace(trace uneTrace1, trace uneTrace2)* qui calcule l'erreur RMS entre deux jeux de données (voir section A.3). Elle renvoie l'erreur sous la forme d'un réel.

(b) Comparer les données expérimentales avec les simulations pour les jeux de paramètres P_a et P_b (voir annexe A.1).

(c) En vous basant sur les valeurs de l'erreur RMS, conclure sur le jeu de paramètres le plus adapté pour simuler cette experience.

3 Travail TP7

3. Écrire les fonctions *saveTraceBin(char* fileTrace, trace uneTrace)* qui enregistre la trace *uneTrace* dans un fichier structuré *fileTrace* et *readTraceBin(char* fileTrace, trace* uneTrace)* qui lit la trace *uneTrace* depuis un fichier structuré *fileTrace*. Tester qu'un fichier généré peut bien être relu ensuite par le programme.
4. Dans la suite, on veut lire les résultats de l'expérience enregistrés dans un fichier texte. Vous allez donc remplacer la fonction donnée au TP6 *expeTrace()* par une fonction *readTraceTxt()*que vous allez écrire.
   (a) Écrire la fonction *readTraceTxt(char* fichier, trace* uneTrace)*, qui va lire le contenu du fichier dont le nom est passé en argument et remplir correctement la structure *uneTrace*.
   (b) En utilisant les fonctions que vous venez de programmer, lisez et affichez les données présentes dans le fichier *cell2.txt* (à récupérer depuis la plateforme Moodle).
   (c) Écrire la fonction *saveTraceTxt(char* fileTrace, trace uneTrace)* qui enregistre la trace *uneTrace* dans un fichier texte *fileTrace*, tel que celui contenant les données expérimentales.
   (d) Tracer à l'aide d'un tableur les données expérimentales et celles simulées en utilisant les fichiers textes générés.
5. Question Bonus : En utilisant des allocations dynamiques et la nouvelle version de la structure *trace*, réécrire les fonctions des questions 1 et 4 pour prendre en compte des traces dont la longueur n'est plus limitée à 4096.

```
typedef struct{
    char comment[40];
    int nbpts;
    float *time;
    float *value;
} trace;
```

4 Travail à rendre

Un seul compte-rendu est à rendre pour les 2 TPs.

Chaque binome doit déposer sur le moodle une archive (.zip) contenant les fichiers avec le code commenté (.c) et un compte-rendu décrivant les tests effectués, les limites de votre programme et les problèmes éventuellement rencontrés. La date limite de dépot est date du TP + 7 jours.

A Annexes

Vous trouverez dans cette section tous les éléments nécessaires à la réalisation du TP.

A.1 Modèle de FitzHugh-Nagumo

Ces deux équations forment le modèle de FitzHugh-Nagumo :

$$\frac{dV}{dt} = (\alpha - V)(V-1)V - w \tag{1}$$

$$\frac{dw}{dt} = \epsilon(\beta V - \gamma w - \delta) \tag{2}$$

Elles simulent la valeur du potentiel électrique d'un neurone (V) en fonction du temps et des paramètres : α, δ, ϵ. w est une variable d'inactivation sans correspondance physique, nécessaire au calcul.

Pour simuler l'expérience décrite dans la section précédente on considère deux jeux de paramètres possibles :

$P_a = \{\alpha = 0.1, \beta = 0.5, \gamma = 1.0, \delta = 0.05, \epsilon = 0.01\}$
$P_b = \{\alpha = 0.1, \beta = 0.75, \gamma = 1.0, \delta = 0.05, \epsilon = 0.3\}$

A.2 Algorithmes d'intégration des équations différentielles.

Cet algorithme simple permet de calculer l'évolution dans le temps d'une équation différentielle.

Algorithm 1 Euler Hybride

Require: t_{max}, dt, $\alpha, \beta, \gamma, \delta, \epsilon$
  tableau Time, Trace
  entier i=0
  reel v=0, w=0, dv=0, dw=0, t=0
  while $t < t_{max}$ **do**
    dv = $(\alpha - v)(v-1)v - w$
    dw = $\epsilon(\beta v - \gamma w - \delta)$
    v += dv * dt
    w += dw * dt
    Time[i]=t
    Trace[i]=v
    i++
    t += dt
  end while

A.3 Mesure d'erreur RMS [1]

Soit x la mesure, $\hat{x}$ la simulation et n le nombre d'éléments :

$$E_{rms} = \sqrt{\frac{\sum_{i=0}^{n}(x_i - \hat{x}_i)^2}{n}} \tag{3}$$

C'est la moyenne quadratique des erreurs entre x et $\hat{x}$.

1. RMS : Root Mean Square