Modélisation d'un processus biologique simplifié sur une page web : Sujet 1

Lola Denet - Université de Bordeaux - Master 1 de bio-informatique 2019-2020

17 Avril 2020

Résumé

La bio-informatique est une discipline peu connue du grand public mais dont l'importance est capitale, notamment dans le domaine de la recherche. En plein essor, elle permet de mettre la technologie au service de la recherche en biologie dans un but d'optimisation. Cet article a pour objectif de présenter une des applications de la bio-informatique : la modélisation d'un processus biologique. Le modèle a été réalisé à l'aide d'une combinaison de plusieurs langages de programmations (HTML, CSS et JavaScript) de sorte qu'il soit accessible depuis une page web. Il est chargé de représenter, de façon très simplifiée, le mécanisme de liaison d'un neurotransmetteur à son récepteur.

Domaines liés à cet article : biologie, bio-informatique, modélisation, neurosciences, programmation web.

Introduction

Cet article est rédigé dans le cadre de la réalisation d'un projet de bio-informatique ayant pour but la modélisation d'un processus biologique simplifié accessible depuis une page web. Notre sujet consiste en une configuration particulière attribuée parmi un total de dix modèles possibles. Dans cette configuration, les récepteurs sont alignés horizontalement sur la limite de l'espace post-synaptique. Les neurotransmetteurs sont positionnés dans l'espace présynaptique et traversent l'espace inter-synaptique à vitesse constante afin de se fixer sur un récepteur. Ils forment ensemble, un récepteur ouvert mobile. Les méthodes de programmation web sont utilisées pour rendre le modèle visuellement agréable et interactif. Nous allons, tout d'abord, présenter les méthodes et outils utilisés pour la réalisation de ce projet avant d'en exposer brièvement les résultats.

1 Matériel et méthodes

1.1 Création de la page web

Afin que le modèle soit accessible depuis un navigateur et/ou en ligne, il a été nécessaire de réaliser un premier fichier au format `.html`. Comme son extension l'indique, il est rédigé en langage *HyperText Markup Language* (HTML). C'est un langage utilisant un système de balises pour être interprété par un navigateur. Il s'utilise généralement en combinaison avec le langage *Cascading Style Sheet* (CSS) pour la mise en forme et avec le langage JavaScript pour la programmation. Une fois les trois fichiers réalisés et liés, le `.html` peut être ouvert dans un navigateur (figure 1). D'autre part, il pourra être mis en ligne à condition de respecter la réglementation et les normes établies par le *World Wide Web Consortium* (W3C) [1]. Un outil en ligne [2] existe pour s'assurer du respect de ces normes de manière automatique.

Notre fichier `index.html` contient les balises nécessaires à l'affichage et l'identification de la page. Celles contenues dans le `<head></head>` permettent d'identifier la page mais aussi de lier ce fichier à celui de mise en forme à l'aide de la balise `<link rel="stylesheet" href="style.css">`. Dans le corps du fichier (`<body></body>`), nous avons créé un lien hypertexte grâce à la balise `<a href="article.pdf" target="blank">Texte à cliquer</a>`permettant d'afficher cet article au format `.pdf` dans un nouvel onglet du navigateur de sorte que l'utilisateur puisse avoir tous les renseignements sur l'utilisation de ce modèle. Ensuite, nous avons utiliser les balises `<form></form>` et `<input></input>` pour permettre à l'utilisateur de saisir la vitesse des neurotransmetteurs et la durée de simulation dans une zone de saisie contenant une valeur par défaut (`value="0.5"` pour la vitesse et `value="120"` pour la durée). L'utilisateur peut aussi débuter, stopper ou réinitialiser la simulation par des boutons à cliquer. Chaque `input` possède un identifiant permettant d'accéder à ses informations depuis le programme. L'option `onclick=""` du bouton permet d'appeler une ou plusieurs fonctions du script qui est associé au fichier `index.html` dès que l'utilisateur clique dessus. La balise `<canvas></canvas>` permet de créer un canevas, c'est-à-dire un espace où il est possible de dessiner des formes. La balise `<script src="main.js"></script>` est chargée de lier ce fichier à celui qui contient le programme en langage JavaScript. Quant au `footer`, il contient du texte affiché en bas de page.

1. `https://www.w3.org`
2. `https://validator.w3.org`

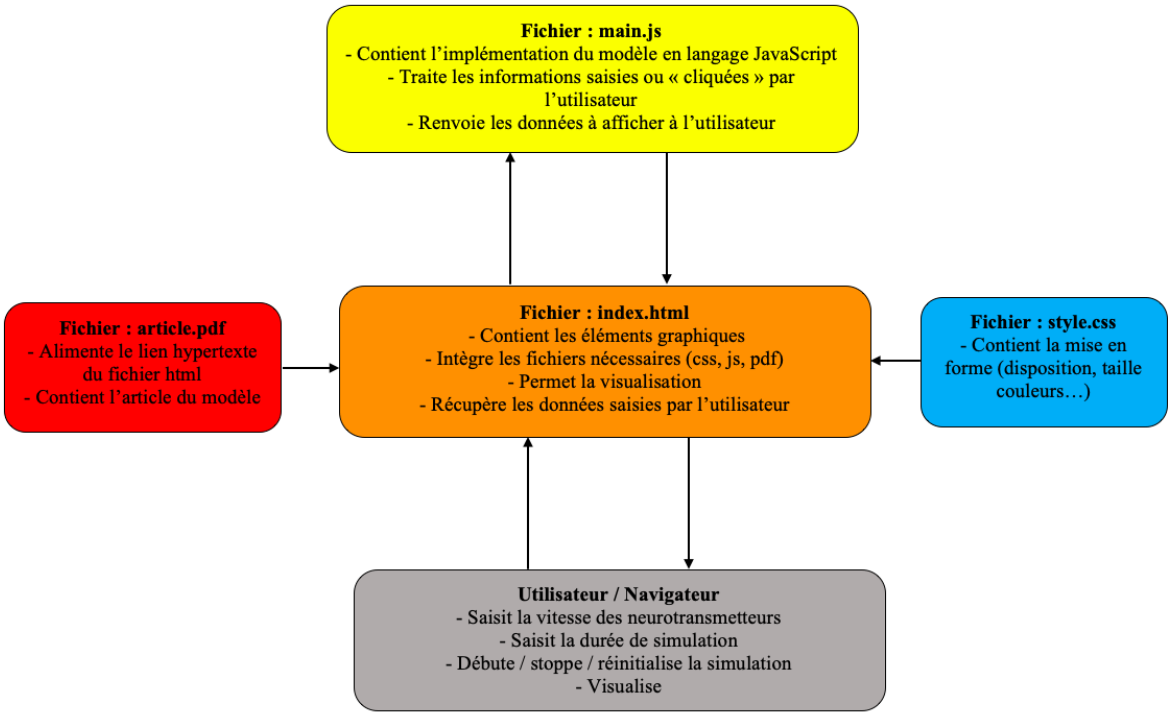

FIGURE 1 – Architecture globale du programme

Cette figure représente les interactions entre les différents fichiers utilisés pour fournir notre modèle fonctionnel et accessible depuis une page web. Les flèches représentent le sens des interactions entre les différents fichiers et/ou l'utilisateur.

Le fichier `style.css` contient la mise en forme de la page. Chaque élément est traité soit par son nom de balise soit par son nom de classe. Ainsi, entre accolades, il est possible de spécifier les options de mise en forme souhaitées (taille, couleur, disposition, typage, etc.)

1.2 Implémentation du modèle

Le fichier `main.js` contient toute l'implémentation du modèle en JavaScript. Nous avons créé une classe par type d'objet (neurotransmetteurs et récepteurs). Chaque classe contient des fonctions qui lui sont applicables : `dessiner()`, `afficher()` et `detecter()`. La première permet de dessiner les objets immobiles sur le canevas. La seconde actualise l'affichage en tenant compte de la vitesse des objets en ajoutant cette vitesse à la position précédente de l'objet. Elle fonctionne comme si plusieurs tableaux étaient superposés. A chaque "mouvement", un nouveau tableau vient remplacer le précédent avec les objets dont les coordonnées ont été actualisées. La dernière fonction permet de détecter la rencontre entre un neurotransmetteur et un récepteur. Elle est chargée de comparer les coordonnées et d'appliquer une modification de forme, de couleur et de vitesse au récepteur lorsqu'une rencontre est détectée. Ces classes sont essentielles à la manipulation des objets.

Le programme principal du fichier peut se diviser en quatre parties distinctes :

Création du canevas : elle se compose de quatre lignes permettant d'utiliser l'objet `canvas` créé dans le fichier `index.html` de façon à l'adapter à la taille de la fenêtre. Les dimensions ainsi recueillies sont stockées dans deux variables.

Création des récepteurs : cette partie permet de créer les récepteurs selon la trame du constructeur de la classe `Recepteur`. Elle initialise un tableau où seront stockés les objets. Leur nombre et leurs positions sont aléatoires. Pour cela, nous avons eu besoin d'autres fonctions : `random()`, `positionner()` et `random_x()`. La première est appelée pour définir un nombre aléatoire de récepteur entre une valeur minimale et maximale (exclue). La seconde est appelée pour créer les valeurs de x (coordonnée horizontale) possibles pour que les récepteurs ne se superposent pas. La troisième est chargée de mélanger une copie du tableau qui contient ces valeurs de x de sorte qu'elles soient attribuées dans le désordre. Enfin, chaque récepteur est créé et stocké dans un tableau avec des coordonnées horizontale et verticale, une vitesse, une couleur et une taille.

Création des neurotransmetteurs : cette partie permet de créer les récepteurs selon la trame du constructeur de la classe `NeuroT`. Elle initialise un tableau où seront stockés les objets. Leur nombre et leurs positions sont aléatoires. Pour cela, nous avons eu besoin de la fonction `random()`. Elle est appelée pour définir un nombre aléatoire de neurotransmetteurs entre une valeur minimale et maximale (exclue). Il a été nécessaire d'utiliser des instructions conditionnelles afin de garantir le bon positionnement des neurotransmetteurs. En effet, ces blocs d'instructions permettent de placer les neurotransmetteurs face aux récepteurs et lorsque la position en première ligne est occupée, les molécules sont placées en seconde ligne puis si nécessaire en troisième ligne. Ainsi, elles ne se chevauchent pas. Ensuite, chaque neurotransmetteur est créé et stocké dans un tableau avec des coordonnées horizontale et verticale, une vitesse, une couleur et une taille. Les coordonnées de `x` sont aussi stoquées dans un dictionnaire et traitée par le constructeur afin d'empêcher le chevauchement. Une fois sorti du bloc itérateur, nous appelons la fonction `ordonner()` de sorte que les neurotransmetteurs soient triés dans leur conteneur. De cette façon, le départ de la molécule se fait d'abord par la première ligne lorsque plusieurs molécules ont la même coordonnée horizontale.

Appels de fonctions : cette partie ne comporte qu'un seul appel de fonction (`initialiser()`). Cette fonction est la dernière à être appelée lors de l'appel du programme par la balise `<script></script>` car les autres appels de fonction se trouvent l'intérieur de cette fonction ou sont déclenchés par les boutons utilisateurs sur la page web. La fonction `initialiser()` permet l'affichage de départ du canevas et des objets. Tout d'abord, elle dessine le canevas, ensuite la limite pré-synaptique et la limite post-synaptique (traits noirs horizontaux). Enfin, elle itère sur les tableaux contenant les objets d'intérêt de sorte que : pour chaque objet contenu dans le tableau, la fonction `dessiner()` est appelée.

Les fonctions `reset()`, `timer()`, `resultats()` et `simuler()` sont appelées directement par l'utilisateur en cliquant sur les boutons de la page.

La première permet de réactualiser la page web et ainsi revenir à l'affichage de départ en générant de nouveaux objets avec des emplacements de nouveau aléatoires. Elle est appelée lorsque l'utilisateur clique sur le bouton `Reset`.

La seconde conditionne la durée de la simulation. Elle est appelée lorsque l'utilisateur clique sur le bouton `Débuter la simulation` de sorte à lancer un minuteur d'une durée totale égale à deux minutes. Ainsi la simulation prend fin à la fin du temps imparti et appelle la fonction `resultats()`. Le temps imparti est défini par défaut à deux minutes mais l'utilisateur peut choisir de le modifier avant de cliquer sur le bouton.

La fonction `resultats()` se déclenche également lorsque l'utilisateur clique sur le bouton `Stopper la simulation`. Elle compte le nombre de récepteurs ouverts en fin de simulation (nombre de récepteurs ayant une largeur de 60 contre 40 pour les récepteurs fermés) et affiche une alerte contenant le nombre de neurotransmetteurs, celui de récepteurs fermés et ouverts. L'utilisateur peut donc stopper la simulation à tout moment ou attendre la fin de la durée définie et obtenir les résultats.

La dernière fonction est chargée de l'animation. Elle est déclenchée par le bouton `Débuter la simulation`. Elle appelle la fonction `initialiser()` puis récupère la vitesse saisie par l'utilisateur. Ensuite, des blocs itérateurs permettent de parcourir les tableaux d'objets et pour chaque objet, les fonctions `dessiner()` et `afficher()` sont appelées. Pour les neurotransmetteurs, un intervalle aléatoire est défini à chaque itération de sorte que la vitesse de chaque objet soit appliquée selon cet intervalle. Par exemple, le premier neurotransmetteur démarre au bout de 10 secondes à la vitesse fixée par l'utilisateur puis l'intervalle est actualisé aléatoirement de sorte que tous les neurotransmetteurs ne quittent pas l'espace pré-synaptique en même temps. La fonction `detecter()` est appelée pour les récepteurs de façon à détecter, à chaque itération, s'il y a une liaison neurotransmetteur-récepteur. La dernière ligne de la fonction `simuler()` lui permet de s'exécuter de manière infinie (du moins, jusqu'à la fin du temps imparti). Cela conditionne la fluidité de l'animation.

2 Résultats

Lorsque le fichier `index.html` est ouvert dans le navigateur, nous voyons la page web de notre modèle s'afficher, comme dans la figure A.1. Le modèle est également accessible en ligne [3]. La page contient : un titre ; un lien hypertexte dirigeant vers cet article dans un nouvel onglet ; deux formulaires où l'utilisateur peut changer la vitesse des neurotransmetteurs et la durée de la simulation, dont les valeurs par défaut sont affichées dans les cases (respectivement 0.5 et 120) ; trois boutons cliquables pour débuter, stopper et réinitialiser la simulation ; le canevas contenant les objets d'intérêt et permettant la simulation ; un pied de page.

Dans la figure 2 nous observons une capture d'écran de la page en cours de simulation.

Les cercles oranges (taille=15) représentent les neurotransmetteurs se déplaçant verticalement vers

3. `https://lola-denet.github.io/projet_web/index.html`

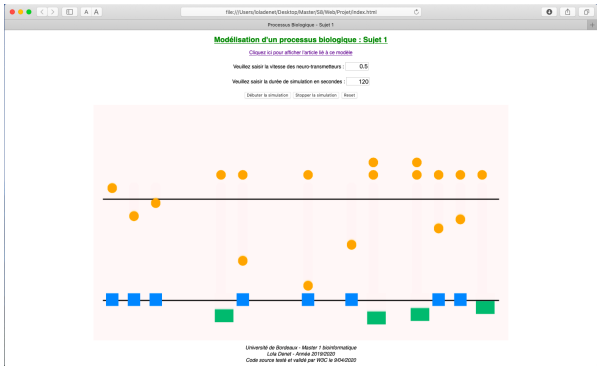

FIGURE 2 – Aperçu d'une simulation

le bas à vitesse constante et intervalles de départ aléatoires. Leurs positions dans l'espace pré-synaptique (dans l'alignement des récepteurs) et leurs nombres (entre 20 et 30 inclus) sont également aléatoires.

Les lignes noires délimitent les espaces : l'espace pré-synaptique au dessus de la première limite, l'espace inter-synaptique entre les deux limites et l'espace post-synaptique en dessous de la dernière limite.

Les récepteurs fermés sont représentés par des carrés bleus immobiles (taille=40x40). Leurs positions sont aléatoires sur la limite post-synaptique. Leur nombre est aléatoirement défini entre 12 et 18 inclus.

Lorsque un neurotransmetteurs occupe la même position qu'un récepteur, ces derniers se transforment, à l'écran, en un rectangle vert (taille=60x40) qui poursuit le mouvement du neurotransmetteur à la même vitesse quittant ainsi l'espace post-synaptique.

La simulation prend automatiquement fin au bout de deux minutes affichant une alerte contenant les résultats comme dans la figure A.2. Cette alerte s'affiche de la même façon avec les mêmes informations si nous cliquons sur le bouton chargé de stopper la simulation.

Lorsque cette alerte est fermée, la page s'actualise avec une nouvelle page de simulation générant un nouveau nombre d'objet aléatoire à de nouvelles positions également aléatoires. Cela se produit également lorsque nous cliquons sur le bouton chargé de réinitialiser la simulation.

Nous avons effectué des simulations de test afin d'évaluer le modèle. Les données de ces simulations sont disponibles sous forme de tableaux dans les figures B.1,B.2 et B.3. Nous avons modélisés ces résultats sous formes de graphiques dans les figures 3, C.1, et C.2.

Avant de définir les intervalles de départ comme aléatoires, nous l'avions fixé à cinq secondes. Cependant, lorsque nous réduisions la vitesse des objets, il pouvait arriver que certains neurotransmetteurs n'aient pas le temps de former une liaison avec un récepteur fermé comme dans la figure C.1.

En définissant un intervalle de départ aléatoire, chaque simulation permet la liaison entre les neurotransmetteurs et tous les récepteurs comme dans la figure 3.

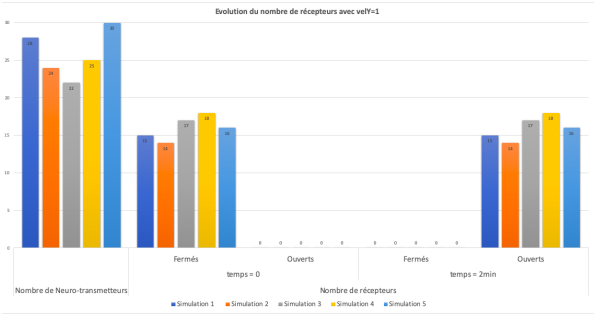

FIGURE 3 – Évolution du nombre de récepteurs à vitesse constante et intervalle aléatoire

Ici, il n'y a aucun récepteur ouvert en début de simulation et aucun récepteur fermé à la fin des simulations. Le nombre de récepteurs ouverts en fin de simulation correspond au nombre de récepteurs fermés du début de simulation. Nous voyons également qu'à chaque simulation, le nombre de neurotransmetteurs et de récepteurs sont différents et donc bien aléatoires dans l'intervalle fixé.

Nous avons fait plusieurs essais en faisant varier la vitesse des neurotransmetteurs afin de vérifier que la première liaison se forme bien dans un délai inférieur à une minute.

Nous pouvons observer sur la figure C.2 que plus la vitesse est réduite, plus le délai d'ouverture du premier récepteur est augmenté. Nous constatons également que la vitesse peut être réduite jusqu'à une valeur de 0.1 pour respecter le délai annoncé précédemment.

Conclusion

Nous avons montré qu'il était possible de rendre accessible, depuis une page web, la modélisation d'un processus biologique simplifié. Notre modèle est conforme aux attentes que nous nous étions fixées puisqu'il est fonctionnel et qu'il permet la visualisation simplifiée du processus de liaison neurotransmetteur/récepteur. De plus, il présente des fonctionnalités permettant à l'utilisateur d'interagir avec celui-ci en modifiant certains paramètres. Par ailleurs, il pourrait être intéressant d'améliorer ce modèle de sorte que la simulation s'arrête lorsque tous les récepteurs sont ouverts et d'afficher à l'utilisateur la durée de la simulation dans la fenêtre de résultats. Enfin, nous pourrions réfléchir à modifier le traitement des résultats de sorte que l'utilisateur puisse les récupérer directement dans un fichier de sorties afin de pouvoir directement les stocker et les manipuler depuis un outil externe à la page web (création de graphiques, calculs statistiques, etc.).

Appendices

A Page web

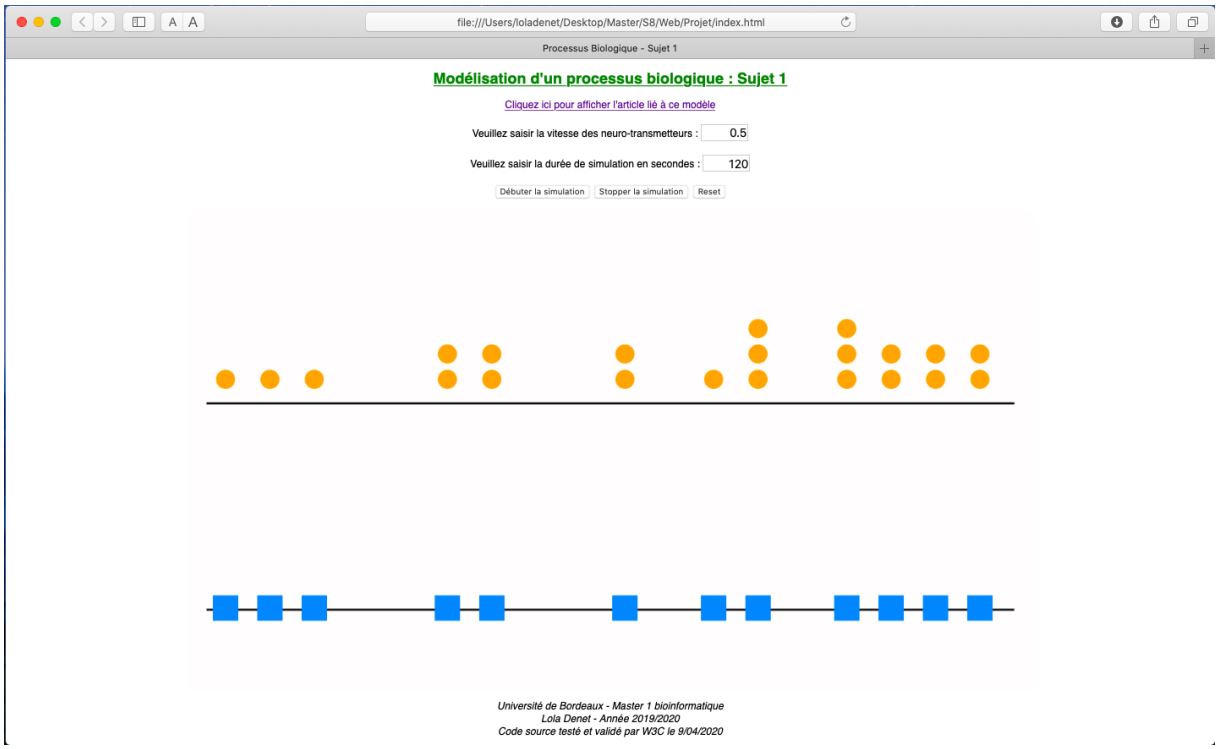

FIGURE A.1 – Aperçu de départ de la page web

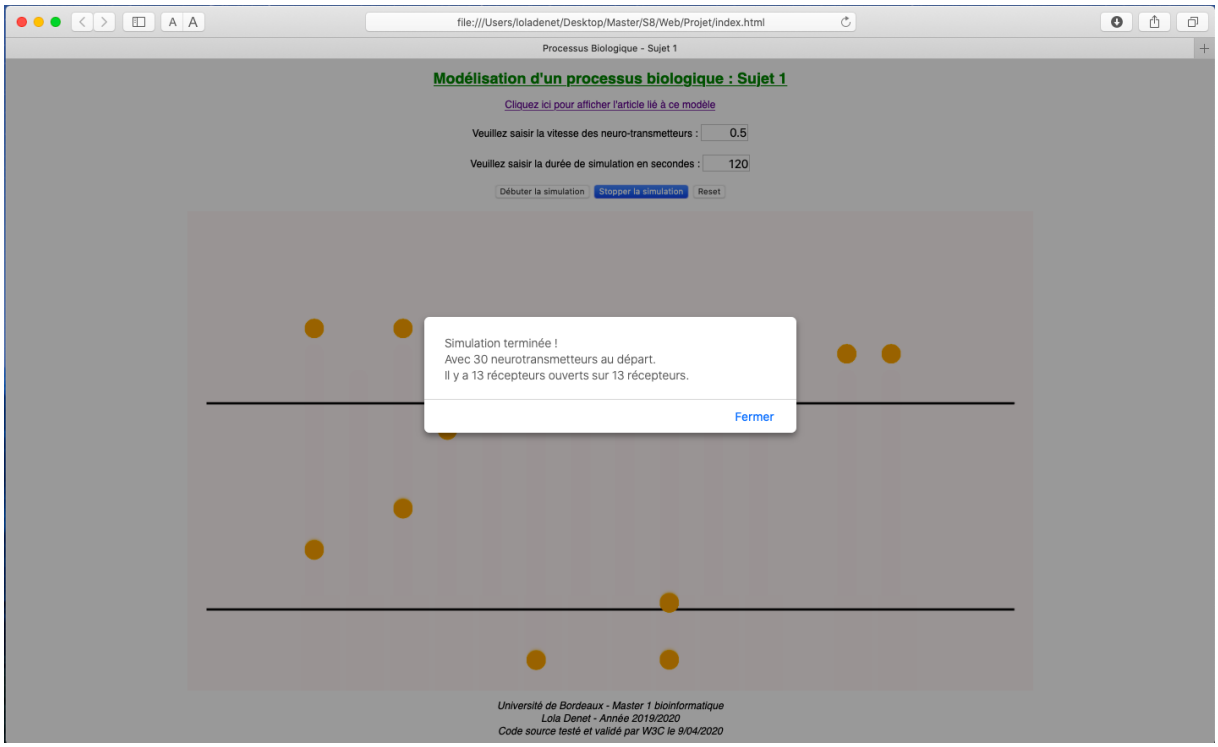

FIGURE A.2 – Aperçu de la fin d'une simulation

B Résultats des simulations : Tableaux de données

Evolution du nombre de récepteurs avec velY=1					
	Nombre de Neuro-transmetteurs	Nombre de récepteurs			
		temps = 0		temps = 2min	
		Fermés	Ouverts	Fermés	Ouverts
Simulation 1	28	15	0	0	15
Simulation 2	24	14	0	0	14
Simulation 3	22	17	0	0	17
Simulation 4	25	18	0	0	18
Simulation 5	30	16	0	0	16

FIGURE B.1 – Évolution du nombre de récepteurs à vitesse constante et intervalle aléatoire

Ce tableau est un exemple réalisé lors des tests. Il est lié a la représentation graphique de la figure 3. Ici, cinq simulations ont été effectuées avec une vitesse des neurotransmetteurs égale à 1 et un intervalle aléatoire. Nous pouvons constater que leur nombre et celui des récepteurs fermés sont aléatoires et compris dans l'intervalle fixé. En début de simulation aucun récepteur n'est ouvert et ils le sont tous une fois qu'elle est terminée. L'intervalle de départ des neurotransmetteurs est aléatoire. Le même type de résultats a été obtenu pour une vitesse égale à : 1, 0.75, 0.5, 0.25 et 0.1.

Evolution du nombre de récepteurs avec velY=0.1 et intervalle=5sec					
	Nombre de Neuro-transmetteurs	Nombre de récepteurs			
		temps = 0		temps = 2min	
		Fermés	Ouverts	Fermés	Ouverts
Simulation 1	25	18	0	6	12
Simulation 2	28	16	0	3	13
Simulation 3	21	16	0	3	13
Simulation 4	23	17	0	4	13
Simulation 5	22	18	0	5	13

FIGURE B.2 – Évolution du nombre de récepteurs à vitesse et intervalle constants

Ce tableau correspond au graphique de la figure C.1. Il a été réalisé lors de tests avec une vitesse 0.1 et un intervalle constant de 5 secondes entre chaque départ de neurotransmetteur. Nous pouvons constater qu'après les 2 minutes de simulation, certains récepteurs sont restés fermés. A l'inverse, à vitesse égale mais avec un intervalle de départ entre les neurotransmetteurs aléatoire, tous les récepteurs sont ouverts en fin de simulation.

Délai d'ouverture du premier récepteur en fonction de la vitesse avec un intervalle aléatoire	
Vitesse	**Temps(sec)**
1	5
0.75	7
0.5	10
0.25	22
0.1	55

FIGURE B.3 – Délai d'apparition du premier récepteur ouvert en fonction de la vitesse

Ce tableau correspond au graphique de la figure C.2. Il traduit, pour chaque test de vitesse, le délai d'ouverture du premier récepteur. Ainsi, nous constatons que la vitesse minimale acceptable est 0.1 pour obtenir une première liaison dans un délai inférieur à 1 minute.

C Résultats des simulations : Graphiques

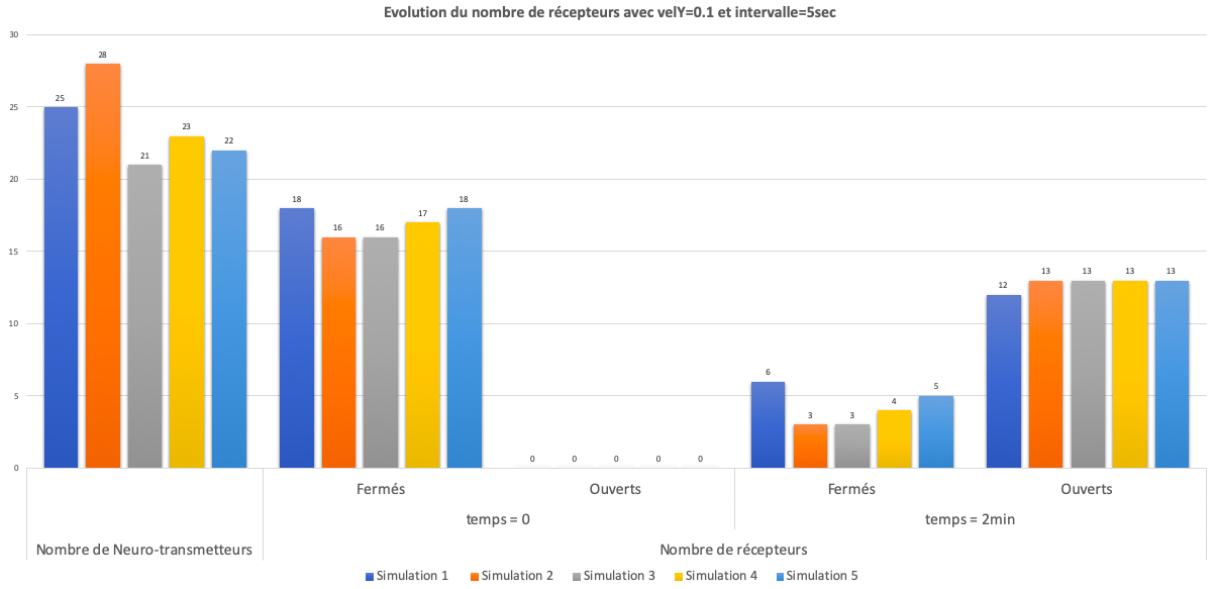

FIGURE C.1 – Évolution du nombre de récepteurs à vitesse et intervalle constants

Le graphique représente les données de cinq simulations (une couleur par simulation). L'histogramme de la partie gauche représente le nombre de neurotransmetteurs pour chaque simulation. Le second histogramme représente le nombre de récepteurs fermés et ouverts au début de chaque simulation (à T=0) et le dernier à la fin de chaque simulation (à T=2 minutes). Nous pouvons constater que certains récepteurs sont restés fermés en fin de simulation avec ces paramètres-ci. En moyenne, seules 13 liaisons ont le temps de se former quel que soit le nombre de récepteurs fermés au départ.

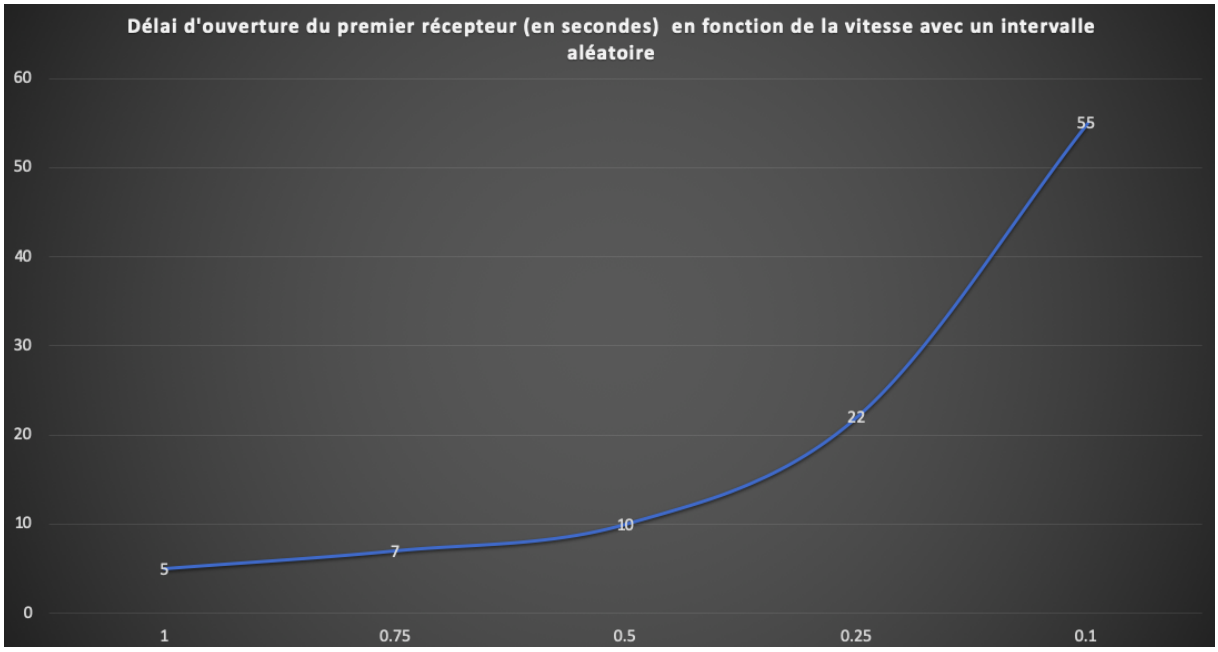

FIGURE C.2 – Délai d'apparition du premier récepteur ouvert en fonction de la vitesse

Cette courbe représente le délai d'obtention de la première liaison entre un neurotransmetteur et un récepteur fermé. La vitesse des neurotransmetteurs est représentée en abscisse. Elle varie de 1 à 0.1. Le temps, de 0 à 60 secondes, est en ordonnée.