THÈSE de DOCTORAT

de l'UNIVERSITÉ TOULOUSE CAPITOLE

Présentée et soutenue par

Monsieur Florian COGONI

Le 18 novembre 2024

Approche participative transdisciplinaire pour la modélisation multi-agents appliquées à la biologie cellulaire

École doctorale : **Mathématiques, Informatique et Télécommunications de Toulouse**
Spécialité : **Informatique et Télécommunications**
Unité de recherche : **IRIT : Institut de Recherche en Informatique de Toulouse**

Thèse dirigée par *Monsieur Sylvain CUSSAT-BLANC et Madame Valérie LOBJOIS*

Composition du jury
Rapporteur : M. Christophe LE PAGE
Rapporteur : M. Vincent RODIN
Directeur de thèse : M. Sylvain CUSSAT-BLANC
Co-directrice de thèse : Mme Valérie LOBJOIS

Résumé

La modélisation est un outil puissant et versatile pour décrire et comprendre les phénomènes qui régissent notre monde. Les mathématiques et plus récemment l'informatique ont permis l'émergence de modèles et de représentations qui permettent une plus grande compréhension de ces mécanismes. Ce niveau d'abstraction permis par les outils numériques est appelé *in silico*.

La biologie s'intéresse à des objets complexes faisant intervenir plusieurs phénomènes qui peuvent interagir à plusieurs échelles. C'est pourquoi cette science se repose, entre autres, sur des représentations et des modèles pour simplifier la réalité et extraire les mécanismes principaux permettant d'expliquer ce qui est observé. Les progrès techniques dans le domaine des sciences computationnelles ont permis à la biologie de bénéficier de nouveaux outils pour approfondir leur compréhension du vivant au travers des modèles *in silico*. Cependant, ces outils requièrent des compétences en mathématiques et/ou informatique qui peuvent freiner les biologistes à les utiliser ou compliquer leur compréhension du modèle dans le cadre de collaborations avec des modélisateurs.

Les travaux présentés dans cette thèse s'intéressent au développement d'une méthodologie et d'une plateforme au service de cette dernière pour faciliter la collaboration entre biologistes et modélisateurs dans le cadre du développement de modèles multi-agents de biologie cellulaire.

Dans un premier temps, nous avons développé une méthodologie participative pour impliquer les biologistes dans le processus de modélisation en facilitant le dialogue et la compréhension du modèle se reposant sur des outils visuels comme des diagrammes UML. Cette méthodologie a pour objectif de réduire le temps de développement et la frustration en maximisant l'intercompréhension, le dialogue et l'implication de tous les collaborateurs.

Dans un second temps, pour appuyer cette méthodologie, nous avons développé une plateforme, nommée ISiCell, permettant de générer le code du modèle à la volée à partir des différents diagrammes. La plateforme permet ensuite de visualiser et d'analyser des simulations. Le but de cette plateforme est de faciliter et d'accélérer la mise en place de la méthodologie tout en offrant un outil versatile pour développer des modèles multi-agents de biologie cellulaire.

Enfin, nous avons éprouver notre méthodologie et notre plateforme en développant différents modèles issus de la littérature, d'anciennes ou de nouvelles collaboration avec des biologistes. Cela nous a permis, à la fois, d'enrichir et de raffiner ISiCell et la méthodologie qu'elle sert.

Ces travaux ont été réalisés au sein de l'équipe Réelle Expression Vie Artificielle (REVA) de l'Institut de Recherche en Informatique de Toulouse (IRIT UMR 5505) et ont été financés par une bourse de doctorat de la région Occitanie et de l'Université Toulouse I Capitole dans le cadre du projet OnkoOptim.

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Au cours des dernières années, la démocratisation de l'informatique a permis l'émergence de nombreux outils numériques adressés à la communauté scientifique. En effet, l'augmentation de la puissance de calcul a permis d'offrir toujours plus d'applications à l'informatique pour d'autres domaines de recherche que ce soit en physique, en chimie, en géologie, en sociologie ou dans le cas qui nous intéresse plus particulièrement en sciences du vivant et de la santé. Les travaux de ce projet de thèse s'inscrivent, effectivement, dans le cadre de collaborations interdisciplinaires entre biologistes et informaticiens.

L'informatique et la biologie ont donné naissance à des disciplines à la rencontre des deux domaines témoignant de la fécondité de l'interdisciplinarité. Ainsi, la recherche scientifique a pu s'enrichir des apports de la biologie computationnelle, de la vie artificielle, des algorithmes bio-inspirés, de la biologie des systèmes ou de la bio-informatique qui représentent tous des disciplines mêlant informatique et biologie à des degrés divers. Ces nouvelles disciplines font interagir différents profils de chercheurs avec des domaines d'expertise ne se recoupant pas mais a également permis l'émergence de profils à l'intersection de l'informatique et de la biologie.

Cette interdisciplinarité a permis l'émergence de nombreux modèles informatiques qui offrent aux biologistes de nouveaux outils pour mieux comprendre les phénomènes qu'ils étudient. En effet, l'usage croissant de simulations numériques de biologie permet aux biologistes de confronter leurs connaissances et leurs a priori à un système numérique basé sur ces derniers pour confirmer/infirmer/raffiner des hypothèses. Certains modèles suffisament robustes peuvent même servir d'outils de prédiction.

Le contexte de la modélisation de biologie cellulaire se prête assez particulièrement au paradigme de la Modélisation Multi-Agents, qui constituent l'approche de modélisation des travaux présentés dans ce manuscrit. En effet, cette branche de la biologie s'intéresse au fonctionnement des cellules au sein de leur

environnement et cette approche de modélisation permet de simuler des ensembles de cellules indépendantes dont le comportement dépend de leurs conditions internes et de leur environnement direct. On permet ainsi aux biologistes de tester des hypothèses sur le comportement des cellules au sein de leur environnement pour expliquer les phénomènes plus complexes observés à l'échelle d'une population.

C'est dans le contexte de collaborations entre modélisateurs et biologistes pour développer des Modèles Multi-Agents de biologie cellulaire que les travaux de cette thèse s'inscrivent.

1.2 Problématique

Comme énoncé précédemment, les travaux présentés dans ce manuscrit s'inscrivent dans le cadre de collaborations entre biologistes et informaticiens pour modéliser de la biologie cellulaire. Ces coopérations nécessitent que chaque participant comprennent au maximum les aspects biologiques et algorithmiques du modèle qui est développé. En effet, l'intercompréhension entre scientifiques appartenant à des domaines d'expertise différent est un élément clef à la fécondité de ces collaborations interdisciplinaires.

De plus, le développement de modèle est généralement incrémental et itératif car les hypothèses et les mécanismes biologiques sont implémentées au fur et à mesure du développement et de la complexification du modèle. Il est donc nécessaire que le dialogue entre informaticiens et biologistes soit le plus efficace possible afin de s'assurer que les attentes des biologistes soient correctement prises en compte et que les contraintes liées au modèle soient entendues également. Ce raffinement itératif du modèle est d'autant plus long que les différents collaborateurs rencontrent des difficultés à se comprendre.

C'est pourquoi la problématique de ce travail de thèse est la suivante : **Comment faciliter et améliorer le dialogue, l'interaction et l'intercompréhension entre modélisateurs et biologistes pour réaliser des Modèles Multi-Agents de biologie cellulaire ?**

1.3 Structure du manuscrit

Le chapitre 2 constitue l'état de l'art de ce manuscrit. Nous commençons dans ce chapitre par introduire la modélisation par des définitions et une rétrospective de l'utilisation de modèles en sciences pour amener à la modélisation dans le cadre spécifique de la biologie. Dans un second temps, nous y abordons les différents niveaux d'abstraction *in vivo, in vitro et in silico* et nous attardons sur les apports de ces derniers aux sciences du vivant. Nous abordons ensuite différentes approches *in silico* appliquées à la modélisation du vivant pour introduire, par la suite, les Modèles Multi-Agents, qui constituent le paradigme de modélisation adopté dans ces travaux, leur application en biologie

et les plateformes permettant la modélisation de processus biologiques. Enfin, nous positionnons nos travaux dans le contexte de cet état de l'art.

Le chapitre 3 développe la méthodologie que nous avons constituée pour le développement participatif de Modèles Multi-Agents basés cellules au travers d'un modèle fil rouge. Il commence par introduire le concept de modélisation participative et plus précisément dans le contexte de la biologie. Nous détaillons ensuite la mise en place et le déroulement de notre méthodologie jusqu'à l'obtention d'un premier prototype fonctionnel et validé qualitativement par les biologistes. La fin du chapitre aborde le devenir du modèle après le développement du premier prototype satisfaisant et se termine par une conclusion préliminaire sur l'intérêt de notre approche.

Le chapitre 4 s'intéresse à l'interface d'ISiCell, la plateforme que nous avons développé pour appuyer notre méthodologie. Nous y détaillons le fonctionnement et l'intérêt de ses trois composantes principales que sont ISiCell Builder, ISiCell Viewer et les outils d'exploration, d'analyse et d'optimisation que sont ISiCell Explorer et ISiCell Workbench. Ce chapitre ce conclue par une prise de recul de la plateforme vis-à-vis de la méthodologie.

Le chapitre 5 détaille les échanges d'informations entre les différentes entités de la plateforme et les choix d'architecture de nos modèles. ISiCell dispose d'une architecture client/serveur et ce chapitre s'intéresse aux échanges entre le client et le serveur de la plateforme. Il explique, dans un premier temps, la génération de code côté client et la structure des fichiers transmis au serveur permettant la génération du modèle. Il s'intéresse, dans un second temps, à décrire la structure de notre métamodèle et comment cette standardisation de l'architecture de modèles facilite la génération et l'injection de code. Les deux sections suivantes s'attardent sur les fonctionnements de l'outil de visualisation et des outils d'analyse Python des simulations. Le chapitre se termine sur les perspectives d'amélioration de la plateforme.

Le chapitre 6 présente les différents modèles développés avec la plateforme. Il présente d'abord trois modèles reproduits de la littérature, la crypte intestinale, l'inflammation musculaire et le modèle d'injection de Lymphocytes T Cytotoxiques. La seconde partie du chapitre concerne les modèles issus de collaborations avec des biologistes en application directe de notre méthodologie avec le modèle de sphéroïde et surtout le modèle de neurogenèse. Il se conclue sur une prise de hauteur sur l'intérêt et l'efficacité de notre méthodologie.

Le chapitre 7 tire les conclusions des travaux développés dans ce manuscrit et proposent des perspectives sur les courts et moyen termes de poursuite de ces recherches. Il se termine sur un bilan plus général qui replace les résultats de cette thèse dans leur contexte scientifique plus global.

Chapitre 2

Modélisation et biologie

Sommaire

Dans ce chapitre, nous allons introduire le concept de modélisation, plus particulièrement dans le contexte de la biologie, et établir un état de l'art ainsi qu'une rétrospective des différentes approches existantes. Nous nous attarderons plus particulièrement sur le cas des modèles multi-agents utilisés dans le cadre de la modélisation de biologie cellulaire et des plateformes permettant de développer ces modèles.

2.1 Introduction à la modélisation

Dans cette section, nous allons introduire et définir les différents concepts attachés à la modélisation de manière générale. Nous détaillerons ensuite les intérêts de l'emploi de modèles et enfin nous nous intéresserons plus spécifiquement au cas de la biologie. Cette section permettra de développer une rétrospective des différents aspects abordés.

2.1.1 Définition et rétrospective de la modélisation

Qu'est-ce qu'un modèle ?

Dans le cadre d'une démarche scientifique, on peut qualifier de modèle un concept ou un objet issu d'un processus de simplification et d'abstraction de la réalité. Cette définition permet d'inclure de manière assez large tout un ensemble de représentations, de techniques et d'outils dans la définition de modèle, des répliques à plus petites échelles de bâtiments ou véhicules, aux cartes géologiques ou topographiques en passant par des simulations statistiques ou informatiques.

L'*Internet Encyclopedia of Philosophy* (IEP) définit, par exemple, un modèle comme une représentation d'un objet, d'un comportement ou d'un système que l'on veut comprendre [1]. Ainsi le modèle peut se définir par analogie avec le phénomène ou l'objet réel qu'il est censé représenté.

La philosophe Daniela Bailer-Jones définissait un modèle comme une description interprétative qui facilite l'accès à ce même phénomène [Bailer-Jones, 2009]. Cette facilitation résulte souvent d'une simplification et d'une réduction des mécaniques et phénomènes étudiés, ce qui induit que les modèles ne constituent qu'une description partielle du sujet étudié. En effet, un modèle est le résultat d'un processus d'essentialisation d'un sujet d'étude afin de dégager certaines caractéristiques dont on veut approfondir la compréhension et la connaissance.

Cela peut se traduire par la conception de modèles réduits pour prototyper des productions plus grandes que ce soit dans les domaines de l'ingénierie [Westine et al., 2012] ou de l'architecture [Picaut and Simon, 2001, Thanachareonkit et al., 2005, Lirola et al., 2017] ou pour reproduire en laboratoire des phénomènes à plus petites échelles dans le cas des sciences expérimentales comme la géophysique [Hubbert, 1937, Penlou et al., 2023a, Penlou et al., 2023b]. Ce procédé permet d'un côté de réduire les coûts de l'étude et d'obtenir des prototypes dont on connaît et contrôle la grande majorité des paramètres. De la même manière, des modèles computationnels offrent encore plus de libertés et réduisent d'autant plus les coûts sur le sujet d'étude au prix d'une abstraction plus élevée car le prototype n'a pas de réalité matérielle et aucune mesure physique ne peut être obtenue à partir de ce genre de prototypes. De son côté, la réalisation de cartes topographiques [Smith, 2005], géologiques [Maltman, 2012, Lisle, 2020] ou routières [Stanojevic et al., 2018, Kim et al., 2023] permettent d'accéder facilement à des informations et caractéristiques sur des zones géographiquement

1. Définition de modèle de l'IEP : https://iep.utm.edu/models/

distantes sans avoir besoin de les obtenir ou mesurer sur place. Ces exemples permettent d'entrevoir un large panel des produits et méthodes de modélisation.

La modélisation regroupe donc tout un ensemble de techniques permettant d'abstraire, simplifier, réduire et essentialiser un objet ou phénomène étudié afin de faciliter l'accès à des informations et caractéristiques recherchées. Par extension, un modèle est le résultat d'un processus de modélisation et constitue une description partielle du sujet étudié et un modélisateur désigne la personne qui participe à la production d'un modèle. D'un autre côté, une simulation désigne la réalisation d'une expérience sur un modèle avec des paramètres fixes et définis. Une expérience peut désigner la réalisation d'un protocole expérimental, l'exécution du code correspondant à un modèle computationnel ou tout autre procédé permettant d'éprouver le modèle dans des conditions contrôlées et d'en obtenir des résultats quantitatifs ou qualitatifs. Si le modèle est stochastique, c'est-à-dire que dans des conditions similaires les résultats peuvent être différents, alors il est souvent nécessaire d'itérer plusieurs fois l'expérience. Chacune de ces itérations constituent une simulation à part entière.

Pour résumé, voici une liste de définitions des différents termes abordés dans cette section et qui seront employés tout au long de ce manuscrit :

- **Modèle :** Description partielle d'un sujet étudié issus d'un processus de simplification, d'abstraction, de réduction et/ou d'essentialisation.
- **Modélisation :** Ensemble des procédés permettant l'obtention d'un modèle.
- **Modélisateur :** Personne réalisant la conception et la production d'un modèle.
- **Expérience :** Protocole permettant d'éprouver un modèle dans des conditions contrôlées.
- **Simulation :** Itération de l'exécution d'une expérience.

Dans la section suivante, nous remettrons la modélisation dans son contexte en philosophie des sciences.

Modélisation et réductionnisme

Dans un contexte scientifique, la position philosophique réductionniste se traduit par une volonté de dégager de grands principes généraux permettant de décrire, d'expliquer et d'inclure le plus de phénomènes possibles afin que les sujets étudiés puissent être considérés comme des cas particuliers de théories plus fondamentales. Dans son ouvrage *The structure of science* [Nagel, 1961], le philosophe Ernest Nagel établit les bases du réductionnisme scientifique. Ainsi il énonce qu'une théorie A se réduit à une théorie B si, précisément, toutes les lois de A peuvent être déduites des lois de B. Ainsi un des exemples les plus utilisés pour illustrer la réduction de Nagel est le suivant : les lois galiléennes de la chute des corps terrestres et celle des lois keplériennes du mouvement des planètes peuvent être réduites à un cas particulier de la théorie newtonienne de la gravitation qui elle-même peut se réduire à la théorie de la relativité générale d'Einstein. Cet exemple permet de bien illustrer les prétentions du réductionnismes qui sont de chercher les lois fondamentales de l'univers permettant d'en

déduire toutes les autres.

La modélisation s'inscrit dans une démarche scientifique du réductionnisme. En effet, un modèle tend à simplifier et réduire les phénomènes qu'il cherche à décrire à un ensemble de règles plus générales. Le modélisateur cherche à comprendre son sujet d'étude en le décomposant en une somme de phénomènes ou de parties, dont il a déjà une connaissance et une compréhension en amont, afin de reproduire les caractéristiques qui l'intéressent.

Sahotra Sarkar distingue trois catégories de réductionnisme [Sarkar, 1992] :

— Le réductionnisme théorique qui s'illustre particulièrement dans les travaux de Nagel et Schaffner [Nagel, 1949, Schaffner, 1967] qui définissent le réductionnisme comme une relation entre théories, comme dans l'exemple précédemment illustré.

— Le réductionnisme explicatif qui statue qu'un phénomène peut-être expliqué par ses parties constitutives et leur organisation au sein de l'objet étudié et s'illustre notamment dans les travaux de Wimsatt et Kim. [Wimsatt, 1976, Kim, 2008]

— Le réductionnisme constitutif ou ontologique qui conçoit que tout objet peut se réduire à un ensemble unique d'entités et de propriétés fondamentales restreintes et matérielles et que toutes propriétés d'un objet est la résultante de propriétés à une plus petite échelle. Cette catégorie de réductionnisme est notamment développée dans les travaux de Davidson, Fodor et Rosenberg. [Davidson, 1970, Fodor, 1980, Rosenberg, 1978, Rosenberg, 1985]

Les deux premières catégories sont qualifiées d'épistémologiques, dans le sens où, elles se reposent sur des représentations du monde et s'appliquent donc à des concepts, des lois scientifiques ou des théories. Tandis que le réductionnisme constitutif s'intéresse à ce qui est/existe et se portera donc sur des objets et des propriétés. Ces catégories ne sont pas mutuellement exclusives mais illustrent les différentes approches du réductionnisme dans le domaine scientifique. Ainsi la modélisation selon qu'elle usera d'outils mathématiques, computationnels et plus généralement de représentations issus de notre perception de notre environnement tendra à s'inscrire dans une approche épistémologique du réductionnisme. D'un autre côté, la reproduction physique de l'objet que l'on étudie afin d'obtenir des propriétés intrinsèques similaires tiendra plus d'une approche ontologique.

Il est bon de rappeler que l'approche réductionniste tend à être biaisée vers une explication physicaliste du monde et de l'ensemble des phénomènes que l'on observe et à proposer l'unification des sciences dans la physique [Oppenheim and Putnam, 1958], supposément plus fondamentale. Cette position est notamment critiquée dans le domaine des sciences naturelles [Ulanowicz, 2001, Brigandt and Love, 2008, Mazzocchi et al., 2011], et des sciences humaines et sociales [Sayer, 2010, Verschuren, 2001]. En effet, le réductionnisme dans sa forme la plus radicale arrive difficilement à prendre en compte les mécanismes issus d'interactions à plus grande échelles (équilibre écologique, effets de groupes, etc.).

Dans la section suivante, nous esquisserons une brève rétrospective de la modélisation dans le domaine des sciences.

Une histoire de la modélisation

La définition large de modèle que nous avons énoncées précédemment intègre un nombre conséquent de concepts et d'objets de sorte qu'il est est assez complexe de trouver un point de départ historique à cette notion. De plus, elle est intimement liée à la notion de science en elle-même qu'elle nous semble indissociable de la mise en place du formalisme scientifique. Ainsi si l'on devait trouver une ébauche de la notion de modèle scientifique historique, on pourrait faire, a minima, remonter l'émergence du concept aux penseurs grecques de l'Antiquité. On peut ainsi penser aux théories atomistes de Démocrite [Berryman, 2004] qui postulait que la matière était composée d'entités dénombrables et insécables, aux modèles astronomiques d'Hipparque [Duke, 2002] décrivant les mouvements des astres ou aux classifications du vivant d'Aristote [Lloyd, 1961]. Ainsi, dès les premières traces écrites de l'Antiquité, on peut trouver des représentations, des théories et des classifications qui démontrent d'une volonté de faire émerger les grands principes gouvernant les phénomènes observables et constituent ainsi des premières tentatives sourcées de modélisation explicative.

Le Moyen-Âge est souvent perçu comme une période d'obscurantisme vis-à-vis de la production scientifique occidentale, on observe tout de même cependant des progrès et de évolutions dans les représentations et théories qui gouvernent le monde. On peut notamment penser aux traités de médecine et notamment d'anatomie issus du monde musulman avec des médecins comme Yuhanna ibn Masawaih ou Ibn Sina [Obaidullah, 2007] ou aux travaux de Roger Bacon en optique [Lindberg, 1997].

L'époque moderne, quant à elle, est marquée par de grands progrès dans les domaines de la cartographie [Turnbull, 1996] et de la navigation dans le cadre des grandes découvertes mais également en astronomie, en chimie et en physique. Cette époque voit émerger les prémices de la science moderne et de sa méthodologie. C'est à cette période qu'a lieu la révolution copernicienne en astronomie remettant en question le modèle géocentrique pour une représentation héliocentrique du mouvement des astres [Kuhn, 1992] complétée, entre autres, par les travaux de Johannes Kepler [Gingerich, 1973, Koyré, 2013]. Les bases de la mécanique classique sont posées un peu plus tard par Isaac Newton [Dijksterhuis et al., 1959] faisant suite aux travaux sur les mouvements des corps de René Descartes [Blackwell, 1966] ou de Galilée [Chow, 2013]. La fin de la période signe aussi la fin des théories arisotélicienne de l'alchimie et l'avènement de la chimie moderne [Thomson, 2023], avec les travaux d'Antoine Lavoisier [Holmes, 1985, Donovan, 1988] et de Claude Louis Berthollet [Lemay and Oesper, 1946, Grand, 1975, Grapí and Izquierdo, 1997], et l'apparition, entre autres, des premières nomenclatures des éléments chimiques et l'énonciation de la loi de la conservation de la masse. Ainsi, l'époque moderne correspond à une période de grand renouvellement scientifique qui voit émerger de nouvelles représentations du monde et des lois qui le gouvernent et posent ainsi les premiers jalons de la science moderne et donc des premiers modèles scientifiques modernes.

Le XIXème siècle voit s'accélérer le développement des sciences et donc la révision des représentations et le développement de nouveaux modèles dans tous

les domaines. On peut penser aux travaux de Thomas Young [Young, 1802] et Augustin Fresnel [Fresnel, 1868] en optique qui ont permis la mise en avant de la conception ondulatoire de la lumière, l'électromagnétisme de Maxwell [Sengupta and Sarkar, 2003] qui unifie les anciens modèles de l'électricité et du magnétisme, la découverte de la plupart des éléments chimiques et leur classification par les travaux de Dmitri Ivanovitch Mendeleïev [Mendeleev, 2013], la théorie de l'évolution de Charles Darwin [Darwin, 1859, Darwin, 1872a, Darwin, 1872b] qui renverse les conceptions figistes du vivant.

Enfin, le siècle dernier a poursuivi cette accélération et permis la structuration de la modélisation contemporaine. Ainsi, la physique s'est enrichie, sur cette période, de la théorie de la relativité d'Albert Einstein [Einstein, 1905, Einstein, 1915a, Einstein, 1915b, Einstein, 2003] et de la physique quantique avec les travaux de Max Planck [Planck, 1901, Planck, 1914, Badino, 2015]. De même, c'est au cours de ce siècle que la génétique prend son essor notamment avec la découverte de la structure de l'ADN par James Watson, Maurice Wilkins et Francis Crick [Watson and Crick, 1953]. L'émergence de l'informatique pendant la deuxième moitié du siècle ouvre la porte à la modélisation et la simulation numérique et l'augmentation exponentielle de la puissance de calcul permet de développer de nouveaux outils théoriques et de prédiction. En effet, l'apparition des méthodes computationnelles permettent à l'ensemble des sciences de profiter d'un nouveau niveau d'abstraction et de simuler rapidement et à moindre coût *in silico* qui était inenvisageable auparavant. Le domaine se développe rapidement à partir de la seconde guerre mondiale notamment durant le projet Manhattan avec les simulations de détonations nucléaires avec des méthodes de Monte Carlo [Benov, 2016].

L'ensemble de la suite de ce manuscrit s'intéresse principalement aux modèles issus des méthodes numériques appliquées à la biologie que nous approfondissons dans les sections suivantes. Dans la section suivante, nous abordons l'intérêt et les avantages d'employer des modèles dans un cadre scientifique.

2.1.2 Intérêts de la modélisation

Dans la section précédente, nous avons défini de manière très large la modélisation puis replacé le concept de modèle dans son contexte philosophique puis historique. Dans cette section, nous nous intéressons aux objectifs de la modélisation, à ses prétentions et aux attentes liées à son emploi.

Prétentions de la modélisation

Dans son article de 1976, Georges Box énonçait que «*tous les modèles sont faux*» [Box, 1976] et complétait en 1978 que cependant «*certains sont utiles*» [Box, 1979]. Au travers de cet aphorisme, Box exprime le fait qu'un modèle ne peut fondamentalement être juste ou vrai car par définition il ne constitue qu'une description partielle du réel et ne peut donc retranscrire toute la complexité de ce qu'il cherche à reproduire. Cependant, cela ne signifie pas que l'usage de

modèle est vain mais qu'il ne faut pas attendre d'un modèle qu'il soit pertinent en dehors d'un domaine pour lequel il a été développé ou éprouvé.

Ainsi, chaque modèle se conscrit à un domaine de validité au sein duquel il peut procurer des informations sur le phénomène ou l'objet qu'il tente de décrire. Ce domaine peut s'étendre en fonction des besoins pour, par exemple, déterminer des cas limites ou éprouver des cas théoriques qui ne sont pas ou ne peuvent pas être observés. Le modèle peut également se complexifier pour prendre en compte de nouveaux cas et répondre à plus de questions et ainsi étendre son espace de pertinence.

La prétention centrale d'un modèle est avant tout de répondre à une ou plusieurs problématiques. Ainsi, il est nécessaire durant la conception et lors des simulations de se demander ce que l'on veut obtenir comme informations du modèle et si celui-ci est pertinent vis-à-vis des besoins et des attentes. Dans leur article [Oh and Oh, 2011], Phil Seok Oh et al. proposent quatre objectifs de la modélisation : décrire, expliquer, prédire et communiquer. Dans l'étude de Daniela Bailer-Jones sur les réflexions des scientifiques sur les modèles [Bailer-Jones, 2002], une des caractéristiques des modèles qui ressort est qu'ils sont construits par des mécanismes d'omission et de simplification afin de capturer l'essence du phénomène étudier.

De notre côté, nous distinguerons donc trois grands objectifs de la modélisation abordés dans les sections suivantes : la simplification, l'exploration et la prédiction.

Un outil de simplification

Par simplification, nous entendons la capacité d'un modèle à décrire des mécanismes et des caractéristiques qui permettent l'explication de plusieurs observations par des processus de généralisation et d'abstraction. Ainsi un modèle peut servir à dégager un ensemble de caractéristiques, de règles et de lois permettant de décrire et d'expliquer tout un ensemble d'observations et de situations similaires. Cela peut concerner des systèmes de classification, des lois mathématiques ou des modèles réduits.

L'usage de systèmes de classification que ce soit en chimie ou en taxonomie permet de dégager des caractéristiques principales communes à des ensembles spécifiques. Cela permet de déduire des propriétés d'un individu, d'un objet ou d'un phénomène à partir de son appartenance à une catégorie à laquelle il a été assigné. On peut ainsi raisonner sur de plus grandes échelles sans prendre en compte chacun des éléments ou individus de l'ensemble.

Les lois mathématiques permettent d'abstraire et quantifier des mécanismes observables et d'en déduire des propriétés et des dynamiques. Cela peut être utilisé pour mettre en évidence l'effet des différents paramètres d'un système sur le phénomène ou l'objet étudié ou comme outil de déduction de caractéristiques à partir de propriétés déjà connues. Cela simplifie donc l'obtention d'informations sur l'objet ou le phénomène étudié à partir d'un nombre restreint de caractéristiques directement observables.

Enfin, le cas des modèles réduits permet de disposer d'un objet moins coûteux et plus facilement manipulable possédant un ensemble de caractéristiques communes avec l'objet ou le phénomène étudié. Ainsi les propriétés que l'on pourra mettre en évidence sur la version réduite pour être attribuer par extension au sujet d'étude. Plusieurs versions simplifiées peuvent être développées en fonction des propriétés et des mécanismes que l'on veut mettre en évidence ou plus spécifiquement étudié.

Un outil de d'exploration

Par exploration, nous entendons la capacité d'un modèle à pouvoir simuler plusieurs situations différentes afin de tester des hypothèses. Cela peut permettre de s'intéresser à des cas limites pour déterminer le domaine de validité du modèle en comparant à des données déjà existantes, de mettre en évidence des situations particulières qui n'ont jamais été observées ou encore déterminer des cas optimaux pour répondre à une problématique particulière.

Ainsi, la détermination des cas limites permet de mettre en évidence des situations qui sont soit théoriquement impossibles, soit nécessiteuses de la prise en compte de nouveaux mécanismes ou encore pour vérifier la validité du modèle sur l'ensemble du domaine étudié. L'intérêt est d'approfondir les connaissances sur le fonctionnement du modèle et donc potentiellement du sujet étudié et sur les améliorations à possiblement apporter pour prendre en compte plus de situations.

L'exploration du modèle peut amener à la découverte de comportements particuliers non triviaux. De cette manière, on peut amener le modèle à déceler des situations qui n'ont peut-être jamais été observées car aucune observation n'a été faite dans les conditions décrites par le modèle ou que ces conditions sont impossibles à obtenir en dehors du modèle par des contraintes qui n'avaient pas été prises en compte auparavant.

Enfin, dans un objectif d'optimisation, l'exploration d'un modèle peut permettre de trouver les conditions optimales à la résolution d'un problème. La modélisation peut ainsi permettre de répondre de manière efficace à des besoins dans les contraintes qui lui sont imposées.

Un outil de prédiction

Par prédiction, nous entendons la capacité d'un modèle à donner des résultats cohérents et pertinents sur une situation dont on ne dispose pas de données. Cela peut concerner des événements futurs qui n'ont pas encore eu lieu, des prototypes avant la production/réalisation d'un objet, d'une expérience ou d'un projet, ou des situations trop complexes ou coûteuses à observer in situ. Tous ces modèles nécessitent d'être suffisament précis et calibrés pour que leur prédiction soient les plus justes et pertinentes.

Prédire des événements futurs peut être un usage de certains modèles. Ainsi, on peut penser aux modèles météorologiques qui font des prédictions à l'échelle de quelques jours ou aux lois astronomiques qui permettent de prévoir la position

des astres à l'échelle de milliers d'années. Ces modèles permettent d'anticiper et de se projeter des cas futures et de prévoir plusieurs scénarios afin de s'adapter en fonction des situations.

Développer un prototype pour un projet, un produit ou une expérience permet de disposer d'un modèle qui permettra de répondre à tout un ensemble de problématiques avant la réalisation. Ainsi les prototypes successifs permettent de raffiner le modèle jusqu'à en obtenir un qui satisfasse les différentes attentes et contraintes avant l'obtention d'un produit ou d'un protocole définitif qui sera employé en situations réelles. Le modèle devra avoir été pertinent dans ses prédictions des différentes situations auxquels le produit final sera soumis pour répondre aux attentes.

Enfin, un modèle peut permettre d'obtenir des informations sur des situations qu'on ne peut pas observer/vérifier directement ou qui sont trop complexes à obtenir. Cela peut concerner la modélisation de phénomènes qui sont spatialement trop éloignés pour pouvoir les observer directement, des événements passés ou dont l'accès est quasiment impossible à aucun outil de mesure à cause des conditions extrêmes nécessaires à l'apparition du phénomène recherché. On peut penser au fonctionnement internes des astres, aux hypothèses archéologiques et/ou historiques ou aux modèles de tectonique géologique.

2.1.3 Modélisation et biologie

Dans le cadre de ce manuscrit, nous nous intéressons plus spécifiquement à l'emploi de la modélisation dans le cadre de la biologie. Ainsi dans cette section nous allons rapidement introduire les différents enjeux de la modélisation en biologie au travers d'exemples de modèles utilisés dans ce domaine.

La biologie englobe un ensemble de processus complexes s'étudiant de l'échelle moléculaire jusqu'à l'étude de l'ensemble des écosystèmes au niveau planétaire. Ainsi, on peut distinguer tout un ensemble de sous-domaines de la biologie avec leurs propres échelles de travail et leurs objets d'étude. De manière non-exhaustive et de l'échelle la plus petite à la plus grande, on peut distinguer :

- La biologie moléculaire, qui s'intéresse au fonctionnement interne des cellules et aux réactions physico-chimiques qui s'y opèrent.
- La biologie cellulaire, qui opère à l'échelle des organites présents dans les cellules.
- La microbiologie, qui étudie des populations de micro-organismes et à leurs interactions avec leur environnement.
- La physiologie, qui opère à l'échelle d'un tissu ou d'un organe.
- L'anatomie, qui étudie la structure des organismes vivants.
- L'écologie, qui s'intéresse à des ensembles d'organismes vivants, à leur environnement et à leurs interactions.

Chacun de ces domaines disposent d'un ensemble de représentations et de modèles selon les sujets étudiés. On peut ainsi penser aux équations de réactions chimiques qui permettent de décrire les interactions qui s'opèrent à l'échelle moléculaire, aux modèles de respiration et de photosynthèse cellulaire qui décrivent des interactions entre une cellule et son environnement, aux représentations du

système vasculaire qui illustre le fonctionnement de la circulation sanguine au sein d'un organisme vivant ou encore aux réseaux trophiques qui mettent en évidence les interactions entre différentes espèces au sein d'un écosystème.

Les enjeux de la modélisation en biologie sont liés à la complexité des phénomènes étudiés qui peuvent être soumis à des variabilités intra- et inter-individuelles non négligeables. Ainsi, les principales attentes de la modélisation en biologie sont la mise en évidence des mécanismes principaux à l'oeuvre dans les phénomènes étudiés dans un but de simplification et de compréhension, l'infirmation ou la confirmation d'hypothèses grâce à l'abstraction permise par la modélisation et enfin la possibilité d'anticiper, provoquer ou prévenir des situations à partir de conditions initiales données.

Dans la section suivante, nous allons nous intéresser plus spécifiquement aux échelles microscopiques de modélisation en nous intéressant à plusieurs niveaux d'abstractions utilisés de manière courante en biologie.

2.2 Modélisation en biologie

Dans cette section, nous allons nous intéresser à trois niveaux d'abstraction classiquement utilisés en biologie (voir Figure 2.1), en commençant par les cas les plus concrets pour aller jusqu'au considérations les plus abstraites.

Chacune de ces familles de modèles sont complémentaires les une des autres et possèdent chacune leurs avantages et inconvénients. Il est en effet plus simple d'explorer avec un modèle plus abstrait et de confronter ensuite les résultats obtenus avec des expériences plus proches de situations cliniques ou naturelles.

2.2.1 Modèles *in vivo*

Les modèles *in vivo* désignent l'ensemble des modèles en biologie dont les données sont issues d'expériences au sein d'organismes vivants. Cela prend en compte l'acquisition de données expérimentales sur des animaux, des plantes ou chez l'Homme, les essais cliniques sont aussi une forme d'expérimentation *in vivo*, le modèle étant le fonctionnement du corps humain. Les modèles *in vivo* sont ceux qui se rapprochent le plus des conditions réelles des phénomènes et mécanismes étudiés chez les organismes vivants et sont donc les plus représentatifs de la situation physiologique. En revanche, l'utilisation d'êtres vivants, en plus de poser une question éthique, est de facto plus coûteuse, plus soumise aux bruits statistiques notamment à cause des variations inter-individuelles au sein d'une même espèce et donc aussi plus difficilement réplicables.

La pharmacologie, qui s'intéresse aux effets de certaines molécules sur les organismes vivants et notamment celui des humains, est un exemple de domaine d'utilisation des modèles *in vivo*. En effet, afin de mettre en évidence l'effet d'un médicament sur un être humain celui-ci doit être testé chez des patients afin de constater l'ensemble des conséquences observables de l'inoculation. Cela permet de confirmer et d'infirmer les hypothèses sur les effets et prévus de la molécule

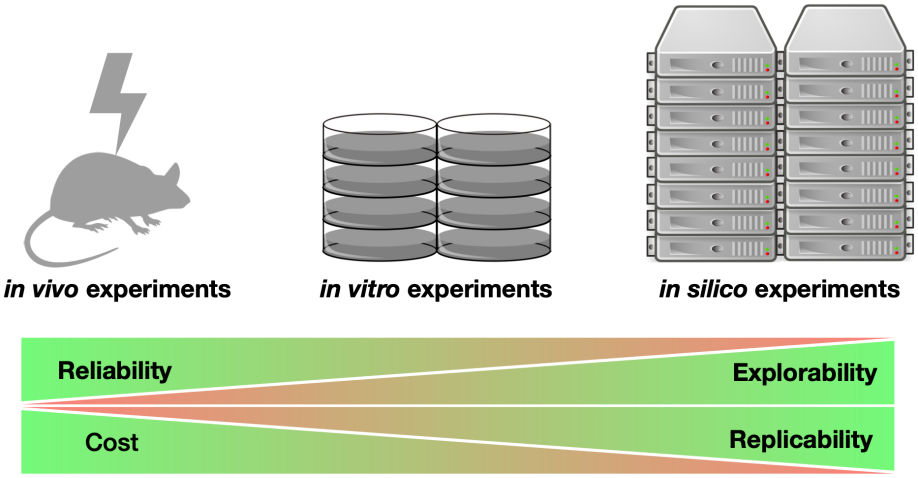


FIGURE 2.1 – **Schéma récapitulatif des trois niveaux d'abstraction en modélisation du vivant.**
De la gauche vers la droite, les modèles sont placés des moins abstraits au plus abstraits, des plus fiables au moins fiables, des plus coûteux au moins coûteux, des moins explorables au plus explorables, des moins réplicables au plus réplicables. Ainsi les expériences *in vitro* représentent un entre deux entre les expériences *in vivo*, qui sont plus fiables mais plus coûteuses et complexes à explorer et répliquer, et les modèles *in silico* qui sont moins coûteux, plus facile à explorer et répliquer mais moins fiables.
Source : [Cussat-Blanc, 2020]

sur l'organisme cible. Dans cet exemple, le modèle est celui de la métabolisation de la molécule dans un corps humain.

Les expériences *in vivo* que ce soit en pharmacologie ou dans tout autre domaine de la biologie s'effectuent principalement sur des animaux non-humains et des plantes. Ainsi pour des raisons de coûts et d'éthique, il est plus simple de mener des études préliminaires à un médicament sur d'autres espèces proches que directement sur des humains. Les animaux les plus utilisés pour cette tâche sont les souris [Albers and Sonsalla, 1995, Assini et al., 2009, van den Buuse, 2010] à cause de leur proximité avec l'humain et leur capacité à se reproduire rapidement. Parmi les modèles de souris certaines sont clonées pour limiter la variabilité interindividuelle [Wakayama et al., 1999, Tamashiro et al., 2003, Kishigami et al., 2006], d'autres sont aussi génétiquement modifiées pour avoir des caractéristiques plus proches des être humains [Cryan and Mombereau, 2004, Davey and MacLean, 2006, Gurumurthy et al., 2020]. L'objectif est d'avoir un modèle aussi proche d'un être humain possible en grande quantité pour un coup réduit afin d'augmenter le volume de données en multipliant les expériences. De manière générale, quand les résultats expérimentation *in vivo* ne peuvent

être obtenu sur des patients humains, d'autres mammifères, notamment des primates, sont généralement utilisés pour garder une proximité avec le sujet cible.

Le poisson-zèbre est un autre animal fréquemment utilisé en tant que modèle *in vivo* pour la pharmacologie [Goldsmith, 2004, van Wijk et al., 2016, MacRae and Peterson, 2023]. En effet, il partage 70% de son génome avec l'être humain [Howe et al., 2013] et se reproduit encore plus vite que les souris, ce qui en fait une alternative intéressante à l'usage de mammifères.

2.2.2 Modèles *in vitro*

Les modèles *in vitro* désignent l'ensemble des modèles en biologie dont les données sont issues d'expériences sur des cellules en dehors d'un organisme vivant ou de leur environnement naturel dans des conditions de laboratoires contrôlées. Cela prend en compte l'étude de colonies de bactéries ou de tissus en boîte de Pétri ou encore des expériences sur des cultures de plantes. Les modèles *in vitro* sont certes plus éloignées de cas clinique chez des organismes vivants mais permettent un plus grand contrôle des conditions expérimentales et sont moins coûteuses, ce qui permet une plus grande liberté d'exploration des protocoles et une plus grande facilité à reproduire les expériences.

Les modèles *in vitro* permettent d'isoler une population de cellules et de les étudier dans un environnement contrôlé et donc de mettre en évidence des caractéristiques et des mécanismes propres sans interaction avec le reste d'un organisme ou d'un environnement naturel. Ceci permet un niveau d'abstraction plus élevé et de s'intéresser aux partie d'un système indépendemment de sa globalité. On dispose ainsi de plus de libertés sur les conditions expérimentales et on peut donc éprouver plus facilement plus de situations.

La modélisation *in vitro* est utilisée dans une grande variété de cas différent pour étudier différents tissus, molécules ou organismes microscopiques. De manière non-exhaustive, on peut penser aux modèles de lésions cérébrales traumatiques [Kumaria and Tolias, 2008, Morrison III et al., 2011], de la barrière hémato-encéphalique [Garberg et al., 2005, Wilhelm et al., 2011], de sphéroïdes pour simuler des tumeurs cancéreuses [Hamilton, 1998, KUNZ-SCHUGHART et al., 1998, Pinto et al., 2020], de croissance bactérienne [O'grady and Pennington, 1966, Chirife et al., 1983, Feldberg et al., 1988, Kincaid and Lavoie, 1989, Cerenius et al., 2010, Mariani et al., 2014], de synthèse d'ARN [Travers et al., 1970, Krieg and Melton, 1987, Beckert and Masquida, 2011] ou d'ADN [Smith et al., 1970, Weymouth and Loeb, 1978], de cellules de foie pour la toxicologie [Guillouzo, 1998, Soldatow et al., 2013, Zeilinger et al., 2016] ou encore de plantules [HUSSEY, 1976, Wardle et al., 2004, Roy et al., 1983, Fuentes et al., 2005]. Tous ces exemples montrent que la modélisation *in vitro* concerne un grand nombre d'applications différentes recoupant plusieurs sous-domaines de la biologie.

Il existe aussi des modèles qui sont à mi-chemin entre les modèles *in vitro* et *in vivo*. On peut penser aux modèles *ex vivo* [Fletcher et al., 2006, Merbah et al., 2011, McLean et al., 2018] qui consistent entre autres à prélever des tis-

sus depuis un organisme vivant et de les cultiver ensuite en laboratoire ou les modèles *in situ* comme les vaches canulées [Hristov and Broderick, 1996, Harvatine and Allen, 2006] ou les souris à chambre [Lehr et al., 1993, Schreiter et al., 2011] qui permettent d'observer et d'expérimenter directement à l'intérieur d'un organisme vivant. Ces modèles ont l'avantage de faciliter l'acquisition et l'exécution de protocole par rapport aux expériences *in vivo* en restant proches de conditions naturelles.

2.2.3 Modèles *in silico*

Les modèles *in silico* désignent l'ensemble des modèles qui se reposent sur l'emploi de méthodes informatiques pour simuler leur sujet d'étude. Cela prend en compte un ensemble assez large d'approches computationnelles qui peuvent être déterministes ou stochastiques, discrètes ou continues. Les modèles *in silico* sont les plus éloignés de la réalité biologique et leur pertinence peut en ce sens être remise en doute. Cependant ils sont les solutions les moins coûteuses pour disposer d'outils d'exploration d'hypothèses grâce à leur capacité intrinsèque à pouvoir offrir une grande liberté et un contrôle absolu sur les conditions expérimentales. Le partage du code de ces modèles leur permet d'être parfaitement réplicables.

La modélisation numérique permet une grande variété d'abstractions. En effet, en fonction des mécanismes que l'on veut étudier, il est possible de ne pas prendre en compte la spatialisation, de discrétiser des valeurs continues ou inversement, de ne pas modéliser les interactions avec l'environnement ou encore en agglomérant plusieurs . Toutes ces abstractions répondront à un compromis entre complexité, explicabilité, performances et réalismes du modèle. Ainsi plus un modèle est complexe et réaliste plus il aura tendance à être computationnellement coûteux et plus difficilement explicable, car il sera plus dur de dégager des mécanismes principaux à l'oeuvre à cause du nombre d'hypothèses prises en compte. A l'inverse, un modèle plus abstrait est souvent moins exigeant en ressources de calcul et intègre moins d'hypothèses ce qui facilite l'explication et la compréhension.

Les modèles *in silico* permettent donc une grande liberté et permettent un travail théorique préliminaire aux développements de modèles *in vitro* ou *in vivo* plus complexes et surtout plus réalistes. Et inversement, le modèle *in silico* peut s'enrichir et se calibrer à partir de données obtenues *in vitro* ou *in vivo*. Ainsi l'approche *in silico* est complémentaire des deux approches précédemment présentées et permet donc d'élargir l'arsenal d'outil disponibles pour modéliser et donc comprendre et appréhender le vivant.

La modélisation *in silico* recoupe le terme de vie artificielle. En effet, ce domaine s'intéresse à reproduire des systèmes ou des organismes vivants de manière artificielle que ce soit au travers de robots, de procédés physico-chimiques comme en biologie synthétique ou à l'utilisation de programmes informatiques. Ces derniers intègrent les modèles *in silico* utilisés pour étudier des phénomènes biologiques. Ainsi on peut considérer que cette famille de modèles s'inscrit aussi dans le domaine plus large de vie artificielle.

De la même manière, les modèles *in silico* s'inscrivent dans le domaine de la bio-informatique et de la biologie computationnelle. Ces deux domaines intègrent des méthodes issus de l'informatique pour traiter et analyser des données issues d'expériences biologiques mais aussi le développement, l'utilisation et l'étude de modèles *in silico* pour éprouver des hypothèses et construire des représentations plus fines de la réalité.

Dans la section suivante, nous approfondissons l'utilisation de solutions de modélisation numérique dans le cadre de la biologie au travers d'une brève rétrospective des modèles *in silico* développés et d'une présentation non exhaustive des différentes approches employées.

2.2.4 Apport des modèles *in silico*

Les premières mentions du terme *in silico* dans la littérature scientifique remontent à la fin des années 1980. On attribue la paternité du terme à Christopher Langton à l'occasion de la première conférence sur la vie artificielle [Langton, 1987], terme dont la paternité est aussi attribuée à Langton sur la même période [Langton, 1986]. On peut faire remonter l'origine de la vie artificielle et de la modélisation *in silico* à la théorisation des automates par John von Neumann [Neumann, 1948] à la fin des années 1940. Il définit un automate comme toute machine dont le comportement progresse logiquement d'étape en étape en combinant des informations provenant de son environnement et de sa propre programmation et énonce l'hypothèse que les organismes vivants peuvent se réduire à de tels automates. Au cours des années 1950, les premiers modèles d'automates tentant de reproduire des organismes vivants sont développés, notamment le modèle de plantes artificielles capables de se répliquer d'Edward Moore [Moore, 1956].

La biologie, en plus de profiter des nouveaux outils issus du développement de l'informatique, va permettre d'enrichir réciproquement les méthodes computationnelles au travers, notamment, des algorithmes bio-inspirés. Ainsi, les premières réalisations de réseaux de neurones artificielles aboutissent à l'implémentation du perceptron par le psychologue américain Rosenblatt à la fin des années 1950. [Rosenblatt, 1957, Rosenblatt, 1958] De la même manière, les travaux de Nils Barricelli dans les années 1950 et 1960 sur la vie artificielle et l'évolution ont permis l'émergence de programmes inspirés du vivant comme les premiers algorithmes génétiques [Barricelli et al., 1954, Barricelli, 1962, Fogel, 2006]. Les systèmes de Lindenmayer (L-système) est un autre exemple d'outil théorique informatique issus de l'observation du vivant. En effet, les L-systèmes correspondent à une grammaire formelle inventé à la fin des années 1960 par le biologiste hongrois Aristid Lindenmayer [Lindenmayer, 1968a, Lindenmayer, 1968b] permettant, notamment, de produire des figures fractales observées chez certaines plantes.

La bio-informatique et la biologie computationnelle [Baxevanis et al., 2020] émergent dans les années 1970 comme de nouveaux champs de recherche pluridisciplinaire à la frontière entre la biologie et l'informatique. On doit la paternité de la bio-informatique aux néerlandais Paulien Hogeweg et Ben Hesper au cours

d'un article de 1970 [Hesper and Hogeweg, 1970] où ils définissent ce nouveau champ comme l'étude des processus d'information dans les systèmes biotiques. La bio-informatique s'illustre particulièrement dans le domaine de la génétique, en effet, le développement des méthodes computationnelles a permis de faciliter l'analyse et le séquençage de l'ADN [Horner et al., 2010, Woo et al., 2010, Pevsner, 2015] et l'étude des protéines et de leur structure [Edwards and Cottage, 2003, Gu and Bourne, 2009, Gáspári, 2020].

Au cours des années 2000, la biologie des systèmes (ou biologie intégrative) s'est intéressée à intégrer les différentes échelles de la biologie pour analyser les organismes vivants comme un système complexe où chaque partie interagit avec l'ensemble aux différentes échelles. Ainsi cette approche holiste prend à contrepied les approches traditionnellement plus réductionnistes des autres champs d'étude de la biologie et de ses modèles. La biologie intégrative se repose aussi sur des techniques computationnelles pour étudier des systèmes vivants de l'échelle moléculaire à l'échelle macroscopique [Kitano, 2002, Bruggeman and Westerhoff, 2007, Chuang et al., 2010, Wanjek, 2011].

Ainsi, le développement de l'informatique a permis l'émergence de champs interdisciplinaires à l'interface entre la biologie et les sciences computationnelles. Ces deux domaines se sont, de cette manière, mutuellement enrichis et ont permis le développement de modèles *in silico* au travers de collaborations entre biologistes et informaticiens.

2.3 Différentes approches de modélisation

Dans cette section, nous allons présenter différentes approches de modélisation du vivant qui co-existent actuellement. Chacune d'elles possède ses avantages et inconvénients et leur pertinence dépendra principalement de ce que l'on cherche à obtenir du modèle et des données à disposition pour le calibrer. Nous allons nous intéresser de manière non-exhaustive à différentes approches de modélisation *in silico*, notamment illustrées par la Figure 2.2. Cette figure illustre la cohérence entre l'approche de modélisation et l'échelle biologique du phénomène étudié. Il faut, cependant, garder à l'esprit qu'il est possible de réaliser des modèles hybrides opérant sur plusieurs échelles [Apeke et al., 2017, Apeke et al., 2022].

Dans cette section, nous nous intéresserons d'abord aux modèles à base d'apprentissage machine, puis aux approches basées sur l'utilisation d'équation différentielle, pour terminer cette section avec les automates cellulaires.

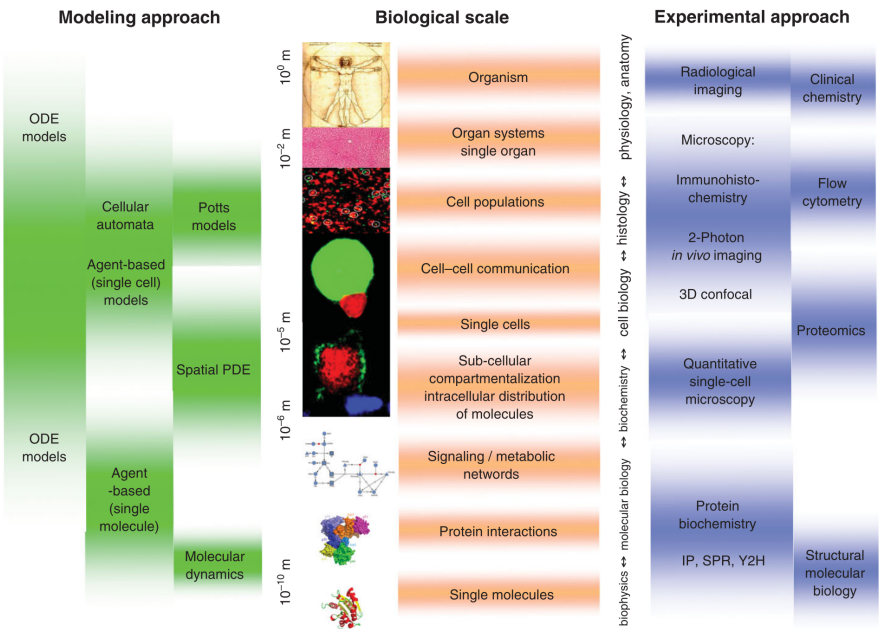


FIGURE 2.2 – **Schéma récapitulatif des différentes approches de modélisation *in silico* en concordance avec les échelles biologiques et les approches expérimentales correspondantes.**
La biologie peut s'étudier de l'échelle de la molécule jusqu'à celle d'organismes vivants dans leur entièreté. Chaque échelle nécessite donc des approches expérimentales différentes pour les observer et les étudier, cela se traduit par le développement d'approches de modélisation *in silico* aussi différentes pour s'adapter aux spécificités de chacune de ces échelles.
Source : [Meier-Schellersheim et al., 2009]

2.3.1 Modèles à base d'apprentissage machine

L'apprentissage machine constitue un sous-domaine de l'intelligence artificielle (voir Figure 2.3) constitué d'un ensemble d'approches et de techniques consistant en l'emploi d'algorithmes capables d'apprendre à partir de jeux de données et de généraliser ses connaissances à de nouveau jeux de données sans instruction explicite. Cette approche permet d'obtenir une grande variété de modèles prédictifs entraînés sur des jeux de données aux formats variés pour des applications toute aussi large. Depuis plusieurs décennies, les modèles à base d'apprentissage machine se sont illustrés dans les domaines de la médecine, de la biologie et de la pharmacologie [Tarca et al., 2007, Jones, 2019, Walsh et al., 2021].

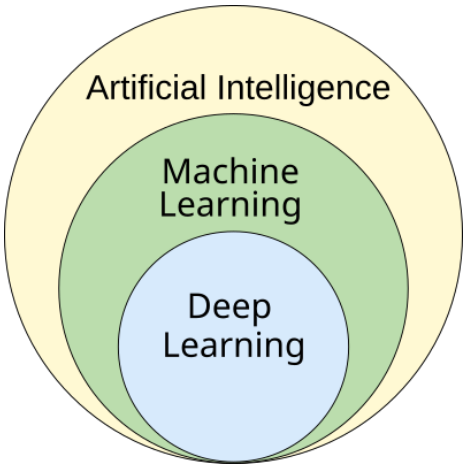


FIGURE 2.3 – **Schéma d'inclusion de l'apprentissage machine**
L'apprentissage machine (machine learning en anglais) est un sous-domaine de l'intelligence artificielle (artificial intelligence en anglais).
Source : [Sindhu et al., 2020]

Réseaux de Neurones Artificiels

Parmi les méthodes d'apprentissage machine, la famille la plus connue est probablement celle des Réseaux de Neurones Artificiels (RNA). Cette méthode s'inspire du fonctionnement des neurones chez les animaux ; un RNA est constitué d'un ensemble de neurones artificiels, disposant d'une entrée et d'une sortie, connectés les uns aux autres par des synapses possédant chacune un poids. Ainsi un RNA peut accepter plusieurs entrées et fournir plusieurs sorties (voir Figure 2.4). Les RNA ont été entraînés en biologie, en pharmacologie et en médecine pour un large ensemble d'applications [Patel and Goyal, 2007, Livingstone, 2008, Suzuki, 2011, Darsey et al., 2015], notamment pour de la classification de cellules à partir d'image de microscopie [Kusumoto and Yuasa, 2019]. Ces modèles fonctionnent cependant souvent comme des boîtes noires, ce qui les rend difficilement interprétable.

Arbres de décision

Les arbres de décision constituent une autre famille populaire de méthodes d'apprentissages machine. Leur fonctionnement se base sur un système de décision ou chaque nœud de l'arbre constitue un choix basé généralement sur un seuil par rapport à une variable d'entrée donnée et qui détermine en sortie une catégorisation ou une valeur. La Figure 2.5 donne un exemple de représentation d'un arbre de décision. Les modèles à base d'apprentissage machine issus d'une approche par arbres de décisions ont permis l'émergence de modèles utilisés dans de nombreux contextes variés [Shazadi, 2021] notamment pour de

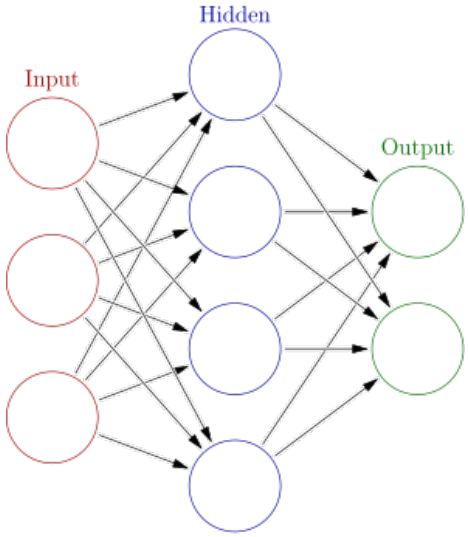


FIGURE 2.4 – **Schéma d'un exemple de réseau de neurones artificiels.** Les neurones sont organisés en 3 couches (entrée, cachée, sortie), chaque flèche correspond à une synapse entre la sortie et l'entrée d'un neurone. Source : utilisateur wikimedia Rayhem sous la licence Creative Commons Attribution-Share Alike 3.0 Unported

la classification en génomique ou en oncologie [Che et al., 2011, Ghiasi and Zendehboudi, 2021]. Ces modèles permettent d'obtenir des arbres de décisions compréhensibles après entraînement, ce qui en fait des modèles explicatifs et prédictifs intéressants. Cependant, ils sont sujet à la suradaptation qui peut résulter en des arbres parfois trop complexes ou trop spécifiques aux données d'entraînement.

Programmation génétique

Une troisième famille de modèles à base d'apprentissage est celle de la programmation génétique. Cette technique consiste en l'utilisation d'algorithmes génétiques pour optimiser des programmes à répondre à un objectif donné. Ainsi par sélection naturelle et itérations successives, le programme obtenu en sortie devient un modèle qui peut être explicatif et/ou prédictif en fonction des besoins. Une manière de représenter un modèle issu d'un processus de programmation génétique est présenté dans la Figure 2.6. La programmation génétique est employé dans plusieurs domaines de la biologie dont la recherche sur le cancer [Worzel et al., 2009], l'étude de cellules de mammifères [Dray et al., 2021] ou en biologie cellulaire pour étudier les structures complexes dues à la chimiotaxie [Bai et al., 2008]. Les algorithmes génétiques peuvent être plus gourmands en ressource que d'autres algorithmes d'apprentissage car ils nécessitent de tra-

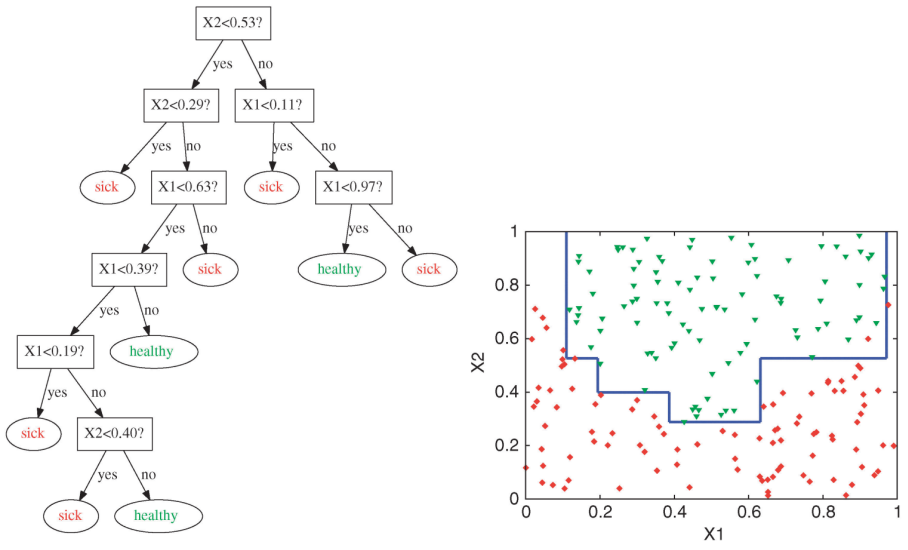


FIGURE 2.5 – **Schéma d'un arbre de décision permettant la classification entre des patients sains ou malade à partir de deux variables continues.**

Source : [Geurts et al., 2009]

vailler avec une population de solutions suffisament large pour ne pas converger trop rapidement vers un optimum local.

Réseaux bayésiens

Enfin une dernière famille d'approche *in silico* par apprentissage est celle des réseaux bayésiens. Ces derniers reposent sur une représentation graphique et probabiliste d'un phénomène étudié par rapport à des connaissances pré-établies (voir Figure 2.7). Ils permettent de mettre en évidence des relations causales stochastiques entre les variables du système étudié en se basant sur les formules de probabilités conditionnelles issues du théorème de Bayes [Thomas, 1765]. Ainsi par itérations successives, le modèle peut faire évoluer les lois de probabilités reliant les variables entre elles ou même modifier les relations causales si celles-ci ne sont pas pré-déterminées. Les réseaux bayésiens sont utilisés en biologie computationnelle pour des applications diverses [Needham et al., 2007, Ziebarth et al., 2013] notamment pour décrire des réseaux de régulation génique [Zou and Conzen, 2005, Liu et al., 2016, Xing et al., 2017]. Ces modèles peuvent être cependant très coûteux à entraîner lorsque le nombre de variables est élevé ou que la structure n'est pas pré-déterminée en amont.

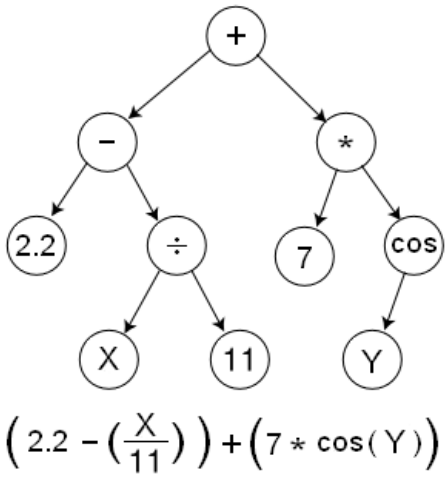


FIGURE 2.6 – **Schéma d'un programme obtenu par un algorithme génétique sous forme d'arbre.**
Chaque nœud correspond à un opérateur et les feuilles à des variables ou des constantes.

Limites des modèles à base d'apprentissage

Tous ces modèles à base d'apprentissage machine permettent d'obtenir des résultats pertinents en offrant une large gamme d'approches différentes, chacune pouvant répondre à un besoin spécifique. Cependant, toutes ces méthodes nécessitent d'avoir un volume de données suffisant pour entraîner des modèles qui pourront répondre aux objectifs attendus, ce qui peut représenter un frein aux collaborations sur des modèles où l'acquisition de de données est coûteuse.

De plus, les modèles résultants de ces modèles émanent du résultat d'un entraînement qui ne permet pas la collaboration entre biologistes et informaticiens pour obtenir le prototype. Ainsi, dans le cadre d'une coopération, les biologistes ne peuvent être partis prenants que dans la constitution du jeu de données d'entraînement et dans la validation du modèle issu de la phase d'entraînement.

2.3.2 Systèmes d'équations différentielles

Les équations différentielles sont un outil mathématique permettant de décrire des relations entre différents paramètres d'un système. Les solutions de ces équations permettent de dégager des lois mathématiques décrivant les dynamiques du système étudié. Dans la suite de cette section, nous intéresserons aux équations différentielles déterministes puis à des équations intégrant de la stochasticité pour se rapprocher des processus naturels soumis à des mécanismes aléatoires.

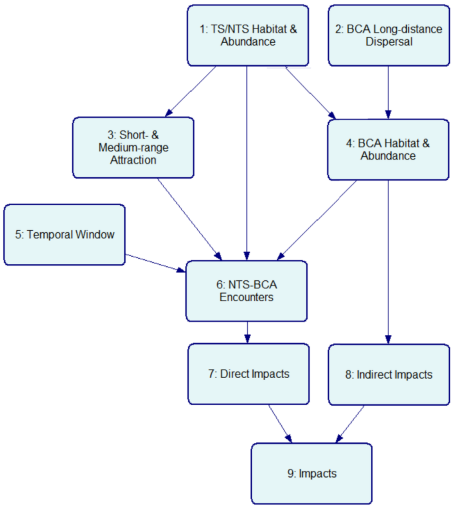


FIGURE 2.7 – **Graphe de la structure d'un réseau bayésien utilisé pour évaluer l'impact d'agents de biocontrôle sur des espèces non-ciblées.** Chaque flèche correspond à un lien causal d'une propriété à une autre. Ainsi l'impact d'agents de biocontrôle (BCA pour Biological Control Agent en anglais) dépend de la fenêtre d'exposition à la substance, de sa capacité de dispersion et de l'habitat et de l'abondance des espèces ciblées ou non (TS/NTS pour Targeted Species/Non-Targeted Species en anglais). On peut ainsi décrire la chaîne causale construite sur les a priori des experts. Chaque dépendance est pondérée pour correspondre aux besoins du modèle. Source : [Meurisse et al., 2022]

Equations Différentielles Ordinaires et Différentielles

Les Equations Différentielles Ordinaires et les Equations Différentielles Partielles (EDO/EDP) constituent un outil largement employé pour décrire des dynamiques continues au sein d'un système. Elles peuvent s'appliquer à des cas assez simples de croissance exponentielle de tumeurs cancéreuses [VP, 1956, Dethlefsen et al., 1968] ou de proies prédateurs avec deux espèces comme dans les équations de prédation de Lotka-Volterra [Lotka, 1925, Volterra, 1926] (voir équation 2.1 et Figure 2.8).

$$\begin{cases} \dfrac{dx}{dt}(t) &= (\alpha - \beta y(t))x(t) \\ \dfrac{dy}{dt}(t) &= (\delta x(t) - \gamma)y(t) \end{cases} \tag{2.1}$$

où
— t est le temps ;
— x(t) est l'effectif des proies en fonction du temps ;

— y(t) est l'effectif des prédateurs en fonction du temps ;
— α est le taux de reproduction intrinsèque des proies ;
— β est le taux de mortalité des proies dû aux prédateurs rencontrés ;
— δ est le taux de reproduction des prédateurs ;
— γ est le taux de mortalité intrinsèque des prédateurs.

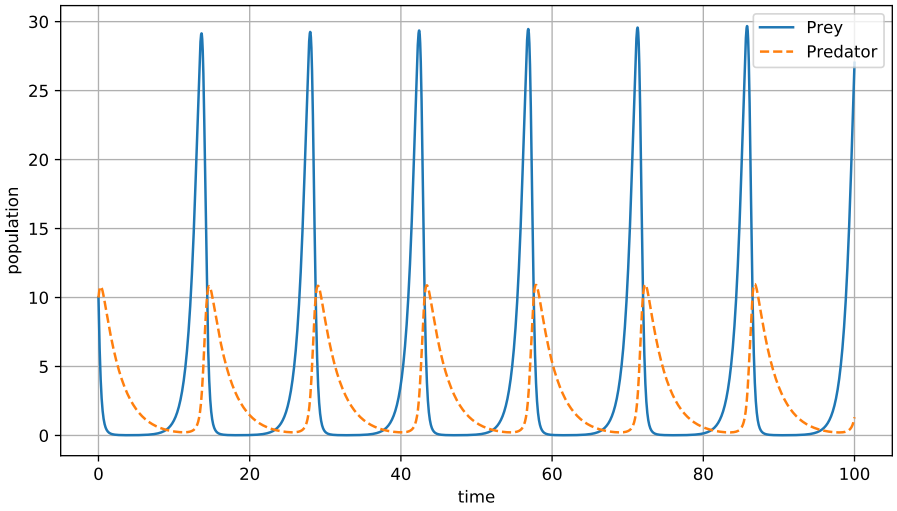


FIGURE 2.8 – **Exemple de dynamiques issues des équations de Lotka-Volterra**
Avec : x0 = 10 ; y0 = 10 ; $\alpha = 1.1$; $\beta = 0.4$; $\delta = 0.1$; $\gamma = 0.4$.
Source : utilisateurs wikimedia Ian Alexander (paramètres, version PNG) Krishnavedala (vectorisation) sous la licence Creative Commons Attribution-Share Alike 4.0 International

Ainsi ce système d'EDO décrit l'évolution des populations de proies et de prédateurs en fonction l'une de l'autre. On emploie aussi les systèmes d'équations différentielles à des situations plus complexes comme des réseaux biochimiques [Breitling et al., 2008, Zhou et al., 2011] ou des dynamiques de réponses inflammatoires [Caudill and Lynch, 2018, Zhang et al., 2019].

Le large emploi des équations différentielles en biologie s'expliquent, en partie, par leur capacité à pouvoir décrire des phénomènes à toutes les échelles du vivant. De plus, leur aspect déterministe permet d'en faire des outils de prédictions pour obtenir des trajectoire d'évolution d'un système à partir de ses conditions initiales. Enfin, cette technique de modélisation permet d'agréger un ensemble de mécanismes co-opérants et de les réduire à un système dynamique dépendant d'un ensemble de conditions initiales et de variables.

Equation Différentielles Stochastiques

Pour prendre en compte le caractère stochastique de la plupart des phénomènes biologiques, un autre type d'équations différentielles est aussi utilisés. Il s'agit des EDS (pour Equations Différentielles Stochastiques) qui permettent d'introduire de la stochasticité aux équations par l'introduction de bruit. Ainsi, au détriment de la perte du caractère totalement déterministe des EDO et EDP, les EDS autorisent de caractériser la part d'aléatoire observée dans la quasi totalité des systèmes biologiques. Elles peuvent, entre autre, être employé pour modéliser des dynamiques de population, de réactions chimiques ou épidémiques [Cresson and Sonner, 2018, Browning et al., 2020].

Si l'on reprend le système d'EDO de Lotka-Volterra (voir équation 2.1), on peut ajouter du bruit à ses équations sous la forme de mouvements brownien [Uhlenbeck and Ornstein, 1930, Wang and Uhlenbeck, 1945, Mörters and Peres, 2010]. Dans son article, Arató [Arató, 2003] propose une extension très simple du modèle proie-prédateur de Lotka-Volterra :

$$\begin{cases} \dfrac{dx}{dt}(t) &= (\alpha - \beta y(t))x(t) + \sigma_1 \frac{dw_1}{dt}(t) \\ \dfrac{dy}{dt}(t) &= (\delta x(t) - \gamma)y(t) + \sigma_2 \frac{dw_2}{dt}(t) \end{cases} \tag{2.2}$$

où
- t est le temps ;
- x(t) est l'effectif des proies en fonction du temps ;
- y(t) est l'effectif des prédateurs en fonction du temps ;
- α est le taux de reproduction intrinsèque des proies ;
- β est le taux de mortalité des proies dû aux prédateurs rencontrés ;
- δ est le taux de reproduction des prédateurs ;
- γ est le taux de mortalité intrinsèque des prédateurs ;
- $\sigma_{1,2}$ sont les constantes pour chaque espèce permettant de quantifier ;
- $w_{1,2}$ sont les bruits brownien des deux espèces.

Ce système d'EDS permet d'ajouter du bruit quantifiable aux équations déterministes de Lotka-Volterra et ainsi de mieux reproduire les dynamiques observées dans la nature. Il existe plusieurs manières d'intégrer de la stochasticité aux modèles de proies/prédateurs et ont donné lieu à plusieurs études sur le sujet [Bahar and Mao, 2004, Bao et al., 2011, Vadillo, 2019, Clenet et al., 2023].

Limites des modèles à base d'ED

Les systèmes d'équations différentiels sont un outil puissant pour décrire des dynamiques au sein d'un système biologique ou écologique. En effet, ils permettent de rendre compte de caractéristiques globales du système et sur leur évolution.

Cependant tous ces systèmes d'équations différentielles ne permettent pas de décrire des mécaniques individuelles que ce soit au niveau des molécules au cours d'une réaction chimique, des cellules à l'intérieur d'un tissu ou d'organismes

vivants au sein d'un écosystème. Elles n'offrent ainsi aucune possibilité de mettre en évidence des caractéristiques émergentes au sein d'un système biologique et donc de tester des hypothèses sur des mécaniques intrinsèques dépendant des éléments constitutifs de l'ensemble étudié.

De plus, l'utilisation d'équations différentielles ne facilitent pas les interactions avec des biologistes. En effet, ces équations constituent une abstraction trop éloignée de ce que les biologistes peuvent observer et rend difficile l'explicabilité des résultats.

2.3.3 Approches ascendantes

Dans cette section, nous aborderons des approches de modélisation dont le fonctionnement est décrit depuis ses éléments atomiques ou depuis sa plus petite échelle. Ses approches permettent de définir des comportements simples à la plus petite des échelles du système et d'observer l'émergence de comportements plus complexes aux plus grandes échelles du système. Ces approches sont connus sous le terme d'approches ascendantes ou *bottom-up* en anglais.

Nous aborderons d'abord des modèles mathématiques de réaction diffusion, puis nous nous intéresserons aux modèles à événements discrets et nous terminerons cette section avec les automates cellulaires.

Systèmes de Réaction-Diffusion

Un Système de Réaction-Diffusion (SRD) est un modèle mathématique qui décrit l'évolution des concentrations d'une ou plusieurs substances spatialement distribuées et soumises à deux processus : un processus de réactions chimiques locales, dans lequel les différentes substances se transforment, et un processus de diffusion qui provoque une répartition de ces substances dans l'espace [Roques, 2013]. Ces systèmes permettent de décrire localement l'évolution d'une concentration et d'observer à plus grande échelle l'émergence de motifs et de comportements plus complexes.

Un des premiers emplois de ces systèmes a été les modèles de morphogenèse chez les êtres vivants et notamment de l'apparition des motifs comme les rayures du Tetraodon (voir Figure 2.9). Alan Turing en étudiant ses motifs par le biais de SRD [Turing, 1952] a donné son nom à ses motifs. Ces modèles ont, par la suite, continué d'être étudiés [Dewar et al., 2010, Maini et al., 2012, Diambra et al., 2015, Vittadello et al., 2021]. Ces motifs sont issus d'EDP spatialisées de la forme suivante :

$$\frac{\partial q}{\partial t} = D \triangle q + R(q) \tag{2.3}$$

où

- q correspond à la concentration d'une substance à une position et un temps fixé ;
- D est une matrice de diffusion ;
- R représente les réactions locales.

Ainsi cette équation permet de décrire à l'échelle locale l'évolution de la concentration d'une substance dans le temps. L'équation correspond à la somme de deux termes : $D \triangle q$ qui correspond à la diffusion de la substance et $R(q)$ qui correspond aux évolutions de concentrations dues aux réactions locales.

FIGURE 2.9 – **Motif de Turing naturel sur un Tetraodon.**
Les motifs complexes présents sur les Tetraodon peuvent être reproduits par des SRD.
Source : utilisateur wikimedia Chiswick Chap sous la licence Creative Commons Attribution-Share Alike 3.0 Unported

Les SRD ont aussi été employée pour modéliser d'autres contextes biologiques. En effet, ils ont été utilisés dans des simulations de chimiotaxie chez des bactéries [Lebiedz and Maurer, 2004], pour étudier la répartition du calcium dans les poils du verticille d'algues unicellulaires [Goodwin and Trainor, 1985], dans des modèles pour la formation des motifs squelettiques dans les membres des vertébrés [Zhu et al., 2009] et de polarité au sein des cellules qui migrent [Otsuji et al., 2007].

Simulation à Evénements Discrets

La simulation à événements discrets (SED) modélise le fonctionnement d'un système comme une séquence d'événements dans le temps. Chaque événement se produit à un moment donné et marque un changement d'état dans le système. Ces modèles permettent de décrire le comportement du système en fonction des occurrences des événements et permet ainsi d'observer des dynamiques émergentes du système en fonction de la séquence étudiée.

Les SED ont été notamment employé dans le cadre de l'aide à la décision en milieu hospitalier pour modéliser l'attribution d'organes en fonction des demandeurs et des donneurs. Ainsi les travaux de Schechter et al. [Shechter et al., 2005]

décrivent un modèle à événements discrets simulant la gestion de l'attribution de foie entre demandeurs et donneurs (voir Figure 2.10).

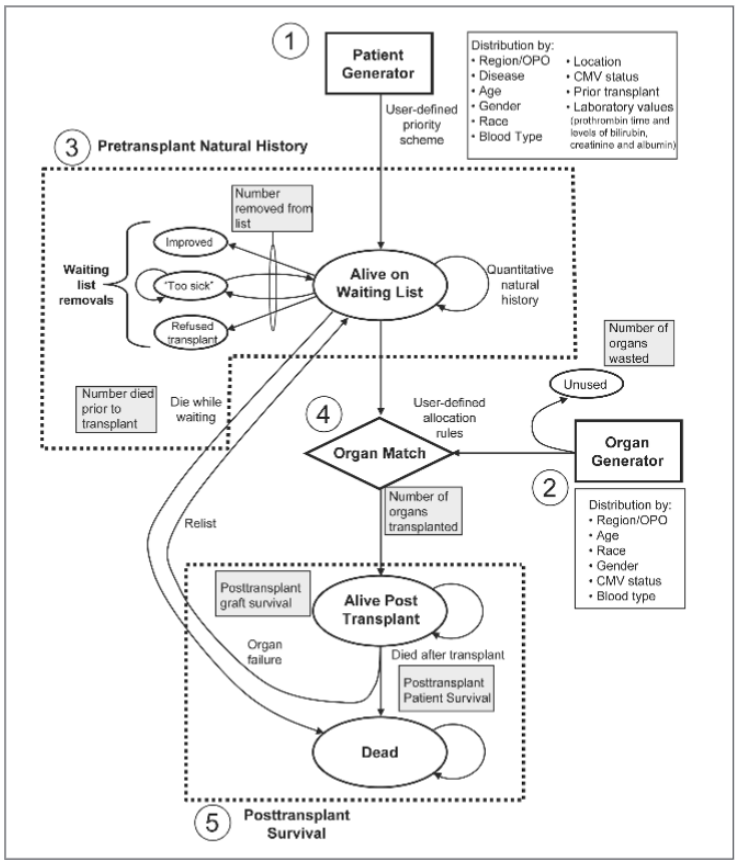


FIGURE 2.10 – **Graphe récapitulatif du modèle à événements discrets du modèle de gestion de dons du foie.**
Le système est décrit par les listes de patients dans différents états et d'organes disponibles. Il permet de dégager plusieurs caractéristiques du système au cours du temps, comme le nombre de patients décédés, d'organes transplantés ou perdus.
Les générateurs de patients et d'organes produisent aléatoirement selon des lois de Poisson respectivement des événements générant nouveaux patients et de nouveaux organes avec des caractéristiques aléatoires. Les patients en attente peuvent quitter le système s'ils obtiennent un événement d'amélioration, de mort ou de refus ou d'acceptation de transplantation. Les organes disponibles peuvent quitter le système s'il reçoit un événement d'inutilité ou s'il y a une transplantation acceptée.
Source : [Shechter et al., 2005]

Ces modèles ont aussi été utilisés pour modéliser les liaisons de protéines de l'ADN [Ghosh et al., 2007], de production de protéines à partir de l'ADN [Yeol et al., 2005] et de malaria [McKenzie et al., 1998].

Automates cellulaires

Les automates cellulaires sont une approche de modélisation à spatialisation discrète apparu très tôt dans l'histoire de l'informatique et de la vie artificielle. Il s'agit d'automates comme définis dans la section 2.2.4, possédant la particularité d'évoluer au sein d'une grille discrète permettant la spatialisation des différents automates. A chaque pas de simulation, chacun des automates se comportera selon les prescriptions de sa machine à états qui pourra dépendre de son voisinage direct.

Depuis leur théorisation au début des années 1950 par les travaux de John von Neumann, de Stanislaw Ulam et de Nils Barricelli [von Neumann, 1951, Sarkar, 2000, Martinez et al., 2013], les automates cellulaires ont été intimement liés au domaine de la vie artificielle. Parmi les modèles de vie artificielle, le plus connu est très certainement le jeu de la vie de John Conway [Gardner, 1970, Bays, 2010] (voir Figure 2.11) dont les règles simples cachent une grandes variétés de comportements et de propriétés émergents [Rendell, 2011]. Plusieurs variantes de ce modèle sont aussi apparues dans les décennies suivantes [Schulman and Seiden, 1978, Fujita et al., 2015, Aguilera-Venegas et al., 2019] suite à l'engouement que ce modèle a apporté au champ des automates cellulaires. Un des exemples les plus récents, est Lenia, un modèle à base d'automates cellulaires, créé par Bert Wang-Chak Chan, fonctionnant selon un principe similaire au jeu de la vie mais avec un temps, un espace et des états continus [Chan, 2018, Chan, 2020, Jain et al., 2023].

Les automates cellulaires ont aussi permis la réalisation de modèles de biologie notamment capables de reproduire ou décrire les motifs complexes des coquilles de mollusques marins [Rucker, 2005, Zawidzki, 2011], le fonctionnement des stomates de plantes [Okayama and Murase, 2002, Peak et al., 2004, Peak et al., 2023], le comportement des fibroblastes [Bouligand, 1986] ou encore le fonctionnement des chromatophores de certains céphalopodes [Manukyan et al., 2017, Fofonjka and Milinkovitch, 2021].

Ces modèles ont d'intéressants par rapports aux approches précédentes qu'ils permettent de décrire des comportements complexes à partir d'éléments subséquents constitutifs du système. Ainsi, cette approche permet de mettre en évidence des phénomènes émergents au sein d'un système depuis ces parts. Cette caractéristique est pertinente dans le cadre de la modélisation de systèmes biologiques qui constituent souvent des ensembles d'éléments plus élémentaires (cellules au sein d'un tissus, espèces vivantes au sein d'un écosystème, etc.).

Dans la section suivante, nous aborderons les modèles multi-agents qui sont une quatrième approche de modélisation similaires aux automates cellulaires. Il s'agit de l'approche de modélisation employée dans la suite de ce manuscrit.

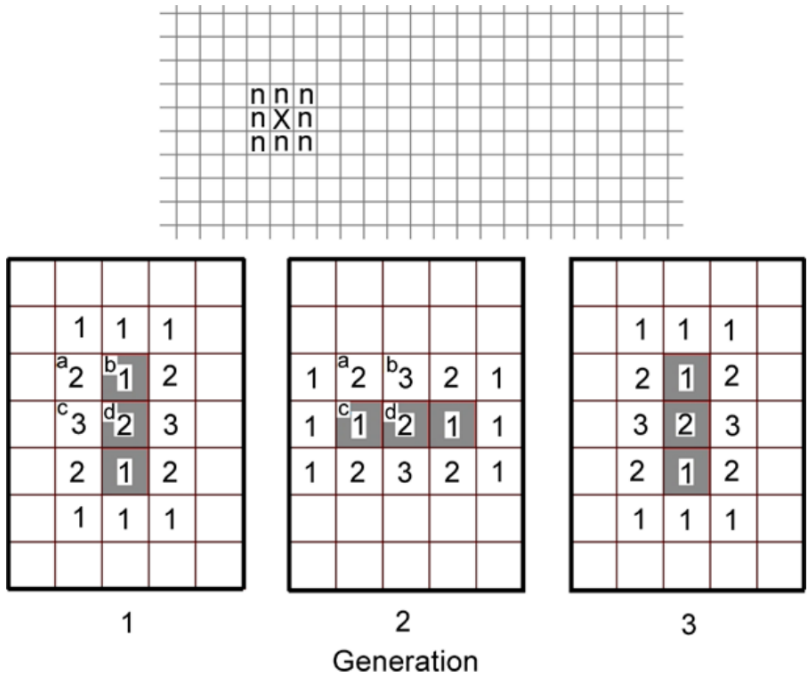


FIGURE 2.11 – **Succession de trois itérations du jeu de la vie à partir d'une configuration initiale**

En haut : Chaque cellule d'une grille a 8 voisins. Les cellules contenant "n" sont voisines de la cellule contenant le "X". Toute cellule de la grille peut être "morte" ou "vivante". **En bas** : La configuration initiale est à gauche. Les cellules grisées sont vivantes ; toutes les autres sont mortes. Le nombre à l'intérieur de chaque cellule donne la quantité de voisins vivants pour cette cellule. Les générations se succèdent selon les règles suivantes : "Les cellules vivantes ayant exactement 2 ou 3 voisins vivants restent vivantes (sinon elles meurent) ; les cellules mortes ayant exactement 3 voisins vivants reviennent à la vie (sinon elles restent mortes)". Pour le passage de la génération 1 à 2, la cellule "a" est morte et comme elle n'a pas exactement 3 voisins vivants, elle reste morte. La cellule "b" est vivante, mais elle a besoin d'exactement 2 ou 3 voisins vivants pour rester en vie ; comme elle n'en a qu'un, elle meurt. La cellule "c" est morte ; comme elle a exactement 3 voisins vivants, elle revient à la vie. Et la cellule "d" a 2 voisins vivants ; elle restera donc en vie. Et ainsi de suite. La configuration se répète toutes les deux générations. De telles formes sont appelées oscillateurs.

Source : [Adamatzky, 2010]

2.4 Introduction aux Modèles Multi-Agents

Dans cette section, nous allons nous intéresser aux Modèles Multi-Agents (MMA) et plus particulièrement à leur utilisation en biologie. Nous en profiterons pour aborder les plateformes permettant la modélisation de MMA conçu pour des simulations de phénomènes biologiques.

2.4.1 Définition et rétrospective

Les modèles multi-agents sont une famille de modèles qui se caractérisent par la description de phénomènes émergents au travers d'interactions entre des entités autonomes, nommées agents, au sein d'un environnement partagé. Ainsi chaque agent se comporte selon des règles prédéfinies en fonction de leurs conditions internes et de leur environnement direct. La parenté du terme agent n'est pas vraiment connue mais une des mentions les plus anciennes d'agents intelligents comme on l'entend dans le cadre des MMA est imputée à Allan Newel dans son article de 1982 *The knowledge level* [Newell, 1982]. Une autre hypothèse donne la paternité du terme agent à John Holland et John Miller suite à leur article de 1991 [Holland and Miller, 1991] reposant sur leurs présentations en conférences des années précédentes.

Les MMA tirent leur origine dans les automates de Von Neumann précédemment présentés dans les sections 2.2.4 et 2.3. En effet, on retrouve dans ces deux approches l'utilisation d'ensemble d'agents indépendants pour observer des comportements émergents à plus grande échelle. On remonte cependant l'origine des premiers modèles multi-agents aux années 1970 avec les travaux théoriques de l'économiste et théoricien du jeu Thomas Schelling sur son modèle de ségrégation [Schelling, 1971]. C'est au début des années 1980 qu'on peut retrouver une première formalisation des MMA avec les travaux de la biologiste néerlandaise Paulien Hogeweg sur la méthodologie de modélisation MIRROR (*Moving Interacting Reproducing and Retiring ORganisms* en anglais pour organismes en mouvement, en interaction, en reproduction et en retraite) [Hogeweg, 1978, Hogeweg and Hesper, 1979, Hogeweg and Hesper, 1985] qui amèneront entre autres à des modèles sur les structures sociales de colonies de bourdons [Hogeweg and Hesper, 1983]. C'est à ses travaux qu'on doit les notions de *ToDo* et *DoDom* qui régissent le comportement des agents et décrivent respectivement l'ensemble des actions possibles et leur priorité.

Au cours des années 1980, les MMA commencent à être employés pour appliquer et étudier des théories issues de la théorie des jeux sur la coopération, le conflit et les interactions interindividuelles en science politique et sociologie. On pensera notamment aux travaux de Robert Axelrod qui débute à la suite d'un tournoi qu'il organise entre plusieurs stratégies du dilemme du prisonnier au début des années 1980 [Axelrod and Hamilton, 1981, Axelrod, 1997] sous forme d'un MMA ou chaque agent s'affronte en suivant la stratégie qui lui a été implémentée. A la fin de la décennie, les premiers modèles de vie artificielle basée sur des MMA apparaissent notamment avec les travaux de Craig Reynolds sur les comportements d'agrégation chez les groupes d'animaux comme les bancs de

poissons, les troupeaux de mammifères ou les nuées d'oiseaux qui amèneront à ses modèles de boids (pour *bird-oids*, qui ressemble à un oiseau en anglais) en 1986 [Reynolds, 1987].

Les années 1990 voit la démocratisation des modèles multi-agents qui sont alors de plus en plus employés, notamment les sciences sociales. En simulation de comportements sociaux, un des modèles les plus connus développés sur la période est Sugarscape basé sur les règles de Joshua Epstein et Robert Axtell développées dans leur ouvrage *Growing Artificial Societies* [Epstein and Axtell, 1996]. Dans ce modèle, les agents sont des habitants évoluant au sein d'un environnement bi-dimensionnel discret contenant des quantités de ressources nommées sucre. A chaque pas de simulation, les habitants peuvent effectuer une grande variété d'actions comme se déplacer, échanger des informations avec un autre agent, se reproduire, mourir ou échanger des ressources.

Au cours des dernières décennies, les modèles multi-agents ont été utilisés pour simuler de nombreux phénomènes dans différents domaines comme les embouteillages au travers de simulations de réseaux automobiles [Nguyen et al., 2021] ou des modèles épidémiologiques notamment lors de la crise du Covid-19 [Wolfram, 2020, Cuevas, 2020, Silva et al., 2020, Kerr et al., 2021, Shamil et al., 2021]. Dans la section suivante, nous allons nous intéresser plus spécifiquement aux MMA appliqués à la biologie qui constituent avec les sciences sociales l'un des plus grands champs d'application de ces modèles.

2.4.2 MMA et biologie

Les modèles multi-agents disposent de plusieurs caractéristiques intéressantes pour simuler des phénomènes complexes impliquant diverses entités constituant un ensemble plus complexe à une échelle plus grande qui sont une caractéristique récurrente en biologie (espèces vivantes au sein d'un écosystème, cellules au sein d'un tissu, etc.). Cette approche permet d'aborder une problématique depuis la base vers une plus grande échelle (*bottom up* en anglais), ce qui permet d'étudier un ensemble depuis les éléments qui le constituent et ainsi de déduire de ses parties le fonctionnement d'un système. Ainsi pour étudier un tissu ou une colonie de bactéries, on pourra s'intéresser aux cellules constitutives de ces systèmes et tenter de retrouver les comportements qui permettent d'expliquer les phénomènes observés à l'échelle de la colonie ou du tissu. Cela permet ainsi d'obtenir des connaissances sur des entités que l'on ne peut pas facilement isoler pour les étudier indépendemment.

Les modèles multi-agents ont été employés pour simuler de nombreux phénomènes, organismes et systèmes biologiques au cours des dernières décennies. Ainsi, en écologie, on trouve de nombreux MMA utilisés pour modéliser et gérer des écosystèmes [McLane et al., 2011, Zhang and DeAngelis, 2020] utilisés notamment pour la gestion de forêts [Pérez and Dragicevic, 2010], d'espèces invasives [Ameden et al., 2009] ou de dynamique de populations de plantes aquatiques [Qi et al., 2009]. De la même manière, ils ont été largement employés pour des modèles allant des échelles de la biologie cellulaire à la physiologie (voir section 2.1.3), dans le cadre de MMA dont les agents sont des cellules [An et al.,

2009]. Parmi ces modèles, on compte des modèles de colonies de bactéries [Lardon et al., 2011], de bourgeons mammaires terminaux [Butner et al., 2016] ou de développement de tumeurs cancéreuses [Rejniak and Anderson, 2011, Wang et al., 2015, Metzcar et al., 2019].

Cette dernière famille de modèle est aussi appelée Modèles Basés Cellules (MBC). Parmi les MBC, on peut distinguer plusieurs approches pour représenter les agents cellulaires qui peuvent être discrètes ou continues. Au sein des approches discrètes, on peut compter les modèles à bases d'automates cellulaires notamment dans des modèles de migration cellulaire [Deutsch et al., 2021] ou de croissance de population de cellules au sein de systèmes microfluidique [Ballesteros Hernando et al., 2019]. Une autre approche discrète permettant de prendre en compte le volume des cellules par rapport aux automates cellulaires est celle des Modèles Cellulaires de Potts (MCP) [Scianna and Preziosi, 2013] aussi appelé modèle Glazier-Graner-Hogeweg [Balter et al., 2007], et a été utilisé pour modéliser également de la migration cellulaire [Scianna et al., 2012] ou des tumeurs cancéreuses [Szabó and Merks, 2013]. Au sein des approches continues, on distingue les modèles basés sur le centre qui sont les plus simples et approximent les cellules à des sphères ou des cercles à partir d'une position et d'un rayon. On peut ainsi utiliser des modèles de physique continus pour gérer les forces s'appliquant aux cellules en contact en fonction de leur distance centre à centre. Ces modèles ont été, entre autres, utilisés pour décrire aussi des croissances de tumeurs [Bull et al., 2020, Macnamara, 2021] ou des systèmes microbiologiques [Guo et al., 2008]. A l'autre bout du spectre, d'un point de vue de complexité computationnelle et de réalisme, existe les MBC dont les cellules agents disposent de membranes déformables [Van Liedekerke, 2019]. Malgré les capacités computationnelles actuelles, ces modèles limitent grandement le nombre de cellules pouvant être simulées et ont principalement été utilisés dans des simulations de cellules épithéliales [Fletcher et al., 2013, Fletcher et al., 2014].

La grande versatilité de cette approche pour modéliser différents systèmes biologiques multi-cellulaires a permis le développement de nombreux modèles mais également à une certaine formalisation de l'approche. De ce cadre ont émergé de nombreux outils et plateformes permettant de faciliter et d'accélérer le processus d'obtention d'un modèle fonctionnel et pertinent. Dans la section suivante, nous allons présenter différents outils permettant d'offrir un cadre de développement pour des modèles agents basés cellules.

2.4.3 Plateformes de modélisation de biologie

La demande pour la réalisation de modèles multi-agents a ouvert la porte à la création de nombreux outils et plateformes de modélisation dédiés aux modèles basés cellules. En effet, ces modèles partagent plusieurs caractéristiques communes qui permettent de généraliser leur architecture et leur fonctionnement. De ces similarités, ont pu émergé plusieurs environnement de travail spécifiquement créés pour simplifier et accélérer le développement de modèles basés cellules. De plus, l'utilisation de ces environnements permet de standardiser la

structure des modèles et de faciliter la reproductibilité.

Dans la suite de cette section, nous proposons une liste non exhaustive de ces plateformes ainsi que d'autres outils de modélisation utilisée en biologie.

NetLogo

NetLogo est un environnement de modélisation multi-agents [Tisue and Wilensky, 2004, Lytinen and Railsback, 2012] disposant de son propre langage. Cette plateforme est notamment utilisée à des fins pédagogiques pour introduire des néophytes à la modélisation multi-agents. Cette plateforme est très versatile et permet de réaliser des modèles agents de physique, chimie ou de biologie. En biologie, NetLogo a permis de développer des modèles de système immunitaire [Chiacchio et al., 2014, Shinde and Kurhekar, 2020], d'auto-organisation des cellules de poissons-zèbres [Dalle Nogare and Chitnis, 2020] ou de crypte intestinale [Bravo and Axelrod, 2013].

NetBioDyn

NetBioDyn est un logiciel permettant de développer des modèles multi-agents de biologie [Rivière et al., 2016, Ballet et al., 2017]. C'est un outil pédagogique dédié aux élèves du secondaires ou du supérieur. Il a été utilisé pour réaliser des modèles de systèmes agro-écologiques [Rodin et al., 2022].

CompuCell3D

CompuCell3D constitue une plateforme performante et open-source conçue pour simuler le comportement cellulaire et le développement des tissus dans des environnements tri-dimensionnels [Swat et al., 2012]. Sa polyvalence permet de modéliser divers processus cellulaires, notamment l'adhésion, la migration et la transduction des signaux. CompuCell3D prend notamment en charge la modélisation multi-échelle et fournit une interface de script Python, ce qui améliore son adaptabilité aux différents besoins de la recherche. CompuCell3D a été utilisé pour des modèles de croissance de tumeurs cancéreuses [Swat et al., 2009], de croissance de biofilms bactériens [Popławski et al., 2008] ou de migration cellulaire [Fortuna et al., 2020].

PhysiCell

PhysiCell est une plateforme de modélisation basée agents et open-source, conçue pour simuler des systèmes multicellulaires [Ghaffarizadeh et al., 2018]. Axée sur l'intégration de modèles cellulaires avec des représentations détaillées de la dynamique intracellulaire, PhysiCell propose une architecture modulaire permettant de simuler des processus biologiques tels que la prolifération et la mort cellulaires, ainsi que les réponses aux signaux micro-environnementaux. Cette plateforme s'est avérée pertinente pour étudier la complexité des systèmes multicellulaires. Elle a principalement été utilisée pour modéliser des tissus cancéreux [Ozik et al., 2018, Preen et al., 2019, Jalajakumari et al., 2022]. En outre,

PhysiCell a trouvé des applications dans divers domaines tels que le développement facial de la souris [Green et al., 2021] et la cicatrisation des tissus [Movilla et al., 2022].

BioDynaMo

BioDynaMo est un environnement C++ très performant qui tire parti de la parallélisation pour simuler de grandes quantités de cellules par simulation. Il est équipé de ses propres outils de visualisation intégrés. La modularité de la plateforme permet diverses applications, y compris l'expression génétique [Sayfullin et al., 2018]. BioDynaMo trouve notamment une utilité particulière dans le domaine des neurosciences, car il a été développé sur la base de CX3D [Zubler and Douglas, 2009].

CellDesigner

CellDesigner se distingue en tant que plateforme de modélisation graphique, mettant l'accent sur la biologie des systèmes en fournissant une interface facilement accessible pour la création de représentations visuelles de réseaux biologiques [Funahashi et al., 2003, Funahashi et al., 2008]. Prenant en charge le SBML (*Systems Biology Markup Language* en anglais) [Hucka et al., 2003], CellDesigner facilite la création de diagrammes de flux détaillés de différents systèmes biologiques (par exemple, la croissance épidermique [Oda et al., 2005] ou les interactions entre les macrophages [Oda et al., 2004]) et facilite l'échange de modèles entre différents outils de simulation. L'accent mis sur la représentation visuelle améliore l'accessibilité pour les chercheurs de toutes les disciplines.

Virtual Cell

Virtual Cell constitue une plateforme de modélisation et de simulation spécialement conçue pour l'analyse quantitative des systèmes de biologie cellulaire [Loew and Schaff, 2001]. Remarquable pour son aide à la création de modèles spatialement réalistes, Virtual Cell fournit des outils de simulation des processus de réaction-diffusion. Cette capacité est particulièrement avantageuse pour l'étude de la dynamique spatio-temporelle des voies de signalisation cellulaires, contribuant à une compréhension plus complète d'un large éventail de processus cellulaires [Slepchenko and Loew, 2010].

COPASI

COPASI (*COmplex PAthway SImulator* en anglais) apparaît comme un outil logiciel polyvalent pour la création et la simulation de modèles biochimiques [Hoops et al., 2006]. Prenant en charge la norme SBML, COPASI offre une interface facilement accessible pour la modélisation, la simulation et l'analyse d'un large éventail de systèmes biochimiques. Sa capacité d'adaptation fait de COPASI un atout précieux dans l'exploration de divers processus biologiques

tels que la thérapie du cancer [Esposito and Picchiami, 2021] ou la régulation du poids [Bauerstätter, 2022].

BioNetGen

BioNetGen relève les défis de la modélisation des systèmes biochimiques complexes grâce à son approche basée sur des règles. En utilisant des règles pour modéliser les interactions entre les molécules, BioNetGen offre une perspective unique sur la dynamique des réseaux biochimiques. Particulièrement utile pour l'étude des voies de transduction des signaux, BioNetGen améliore la compréhension des processus cellulaires complexes tels que la voie de signalisation du système immunitaire inné [An and Faeder, 2009].

Chaste

Cancer, Heart And Soft Tissue Environment (Chaste) est une bibliothèque C++ open source très performante pour la simulation informatique de modèles mathématiques développés pour la physiologie et la biologie [Mirams et al., 2013, Cooper et al., 2020]. Elle permet de produire des modèles basés cellules discrets ou continus bi- ou tri-dimensionnels. La bibliothèque prend en charge la résolution d'équations différentielles et l'utilisation de maillages 3D. Chaste a été utilisé pour des modèles de croissances de sphéroïdes, de cryptes entérites, de cœur [Corrias et al., 2012], de tissus gastro-intestinaux [Corrias et al., 2013] ou d'angiogenèse [Grogan et al., 2017].

Tellurium

Tellurium est un environnement de développement modulaire, open source et basé sur des librairies Python permettant de produire des modèles de biologie [Choi et al., 2018] et de les analyser. Il dispose de deux éditeurs de code similaires à Spyder IDE et une autre basé sur le système de notebooks de Jupyter [Medley et al., 2018]. Il a été utilisé pour étudier des dynamiques de réactions chimiques dans le cadre de simulation de biologie moléculaire et pour des modèles de métabolisme de bactérie ou de cellules souches.

Morpheus

Morpheus est un environnement de modélisation pour la simulation et l'intégration de modèles cellulaires avec des équations différentielles ordinaires et des systèmes de réaction-diffusion [Starruß et al., 2014]. Il dispose d'une interface graphique pour développer et analyser les modèles directement dans la plateforme. Il a été utilisé pour des modèles de migration de bactérie, de transdifférenciation et de formation de motifs dans le pancréas et de morphogenèse vasculaire.

Tissue Forge

Tissue Forge est un environnement open source pour la modélisation et la simulation de la physique, de la chimie et de la biologie basées sur les particules [Sego et al., 2023b, Sego et al., 2023a]. Il prend en charge des applications allant de la dynamique moléculaire classique aux modèles agents multicellulaires. Il a déjà été utilisé pour des modèles d'interactions moléculaires, de sphéroïdes cancéreux et de prolifération dans une crypte entérite.

2.5 Positionnement et objectifs des travaux

Nous avons proposé dans ce chapitre un aperçu et une rétrospective de la modélisation et plus spécifiquement de l'usage des modèles multi-agents dans le cadre de la modélisation de systèmes impliquant des populations de cellules. Nous avons terminé cet état de l'art par une présentation de différentes plateformes permettant de développer des MMA en biologie. Cependant, ces outils sont avant tout adressés à des modélisateurs ayant déjà une expérience en développement logiciel et limitent donc les interactions avec les biologistes qui ne sont pas accoutumés à dialoguer et travailler avec des modélisateurs (voir Table 2.1). Ainsi, ces plateformes n'offrent pas de solutions satisfaisantes pour permettre une collaboration efficiente au développement de modèles.

Nos travaux se concentrent sur le développement d'une méthodologie et d'une plateforme au service de cette dernière dans le cadre de la modélisation multi-agents de phénomènes biologiques impliquant plusieurs cellules, qui constitueront nos agents. Ainsi, les domaines d'application de nos modèles s'intéressent à des phénomènes étudiés aux échelles de la biologie cellulaire, de la microbiologie et de physiologie comme définis dans la section 2.1.3. Notre problématique peut donc se résumer ainsi :
Comment développer efficacement des modèles multi-agents de biologie en impliquant les biologistes dans le processus de modélisation ?
Pour répondre à cette problématique, nos objectifs sont les suivants :

- améliorer et stimuler l'implication des collaborateurs durant le développement de modèles,
- accélérer le développement de modèles,
- proposer un cadre méthodologique général,
- disposer d'une plateforme permettant d'appliquer notre méthodologie.

Pour répondre à notre problématique et atteindre nos objectifs, nous avons développé une méthodologie participative présentée dans le chapitre suivant ainsi qu'une plateforme qui permet une application efficiente de notre approche présentée dans les chapitres 4 et 5.

Plateforme	Référence	Compétences requises	Interface de développement	Interactivité des simulations
NetLogo	[Tisue and Wilensky, 2004, Lytinen and Railsback, 2012]	langage simplifié	Editeur de code	Modification de paramètres, visualisation intégrée de simulations
NetBioDyn	[Rivière et al., 2016, Ballet et al., 2017]	-	Assistant logiciel	Visualisation intégrée de simulations et de graphiques
CompuCell3D	[Swat et al., 2012]	Python	Editeur de code	-
PhysiCell	[Ghaffarizadeh et al., 2018]	C++	Assistant logiciel [Heiland et al., 2023]	Visualisation intégrée, modification de paramètres
BioDynaMo	https://www.biodynamo.org/	C++	Bibliothèque C++	Visualisation avec Paraview [Ayachit, 2015]
CellDesigner	[Funahashi et al., 2003, Funahashi et al., 2008]	SBML	Editeur de diagrammes	Modification de paramètres, visualisation intégrée de graphiques
Virtual Cell	[Loew and Schaff, 2001]	VCML, SBML	Assistant logiciel	Visualisation intégrée
COPASI	[Hoops et al., 2006]	SBML	Assistant logiciel	Visualisation intégrée de graphiques
BioNetGen	https://bionetgen.org/	BNGL, SBML	Editeur de diagrammes et de code	Visualisation intégrée de graphiques
Chaste	[Mirams et al., 2013, Cooper et al., 2020]	C++	Bibliothèque C++	Visualisation avec Paraview
Tellurium	[Choi et al., 2018]	SBML, Python	Editeur de code, notebooks	-
Morpheus	[Starruß et al., 2014]	SBML	Assistant logiciel	Visualisation intégrée
Tissue Forge	[Sego et al., 2023b, Sego et al., 2023a]	C++, Python	Bibliothèque C++ et Python	-

TABLE 2.1 – **Table récapitulative des différentes plateformes abordées**

Chapitre 3

Une méthodologie participative

Sommaire

Dans le chapitre précédent, nous avons fait l'état de l'art des différentes méthodes de modélisation dans le cadre de la biologie et plus particulièrement sur le cas des MMAs. Dans ce chapitre, nous allons détailler l'approche participative que nous avons mise en place pour améliorer les interactions entre biologistes et informaticiens dans le cadre de la modélisation de Modèles Multi-Agents (MMA). Cette méthodologie a fait l'objet d'une publication dans un conférence internationale [Cogoni et al., 2024]. Nous nous reposerons sur un modèle théorique de crypte intestinale qui servira de fil rouge pour illustrer les différents aspects de notre méthodologie.

3.1 Introduction à la modélisation participative

Dans cette section, nous allons introduire la notion de modélisation participative, avant de détailler la méthodologie que nous avons développée dans le cadre de cette thèse au travers de notre modèle théorique. Dans le chapitre suivant, nous nous intéresserons aux outils que nous avons développés et mis en place pour soutenir notre méthodologie.

3.1.1 Approche participative en modélisation

La modélisation participative est un paradigme qui implique des participants spécialisés dans différents domaines dans le processus de développement de modèles en mobilisant leurs connaissances à chaque étape. Cette philosophie de modélisation est couramment utilisée dans les domaines de la gestion des ressources environnementales [Hare et al., 2003, Castelletti and Soncini-Sessa, 2007, Basco-Carrera et al., 2017] mais, à notre connaissance, ne l'est pas encore dans le cas de la biologie cellulaire. Ce domaine fait pourtant intervenir et interagir des experts de domaines différents (biologistes, informaticiens, mathématiciens, chimistes, physiciens, etc.) qui n'ont pas toujours les clefs de compréhension nécessaires pour dialoguer efficacement avec les spécialistes des champs d'études dont ils ne sont pas issus.

L'objectif de l'approche participative est de rendre proactifs les différents collaborateurs afin de limiter la frustration et de rendre le développement plus efficace au travers du dialogue entre les différents spécialistes et les modélisateurs. Il existe différentes méthodes et outils et employés pour faciliter ces collaborations interdisciplinaires [Voinov et al., 2018]. Leur utilisation dépend principalement de la familiarité des différents acteurs avec ces outils et du phénomène que l'on veut modéliser. Les collaborations entre informaticiens et biologistes pouvant être compliquées dues, entre autres, au manque d'intercompréhension entre experts de domaines différents, il semble donc pertinent et intéressant d'appliquer les enseignements des approches participatives, développées dans d'autres domaines, à celui de la modélisation de biologie cellulaire.

Dans cette approche, les MMAs sont considérés comme un outil permettant

d'obtenir un modèle de manière collaborative. En effet, ils offrent la possibilité de gérer facilement un grand nombre d'agents spatialisés, autonomes et décentralisés, ce qui est particulièrement bien adapté aux systèmes socio-écologiques [Bousquet and Le Page, 2004, Dray et al., 2006] ainsi qu'à la modélisation des tissus biologiques [Thorne et al., 2007]. De plus, ils sont réputés pour être des modèles accessibles à des personnes qui ne sont pas familières avec l'informatique et la modélisation [Bodine et al., 2020, Blikstein and Wilensky, 2010].

La modélisation d'accompagnement ou ComMod [1] (pour *companion modelling* en anglais) est une approche qui s'inscrit dans le paradigme de la modélisation participative. Elle utilise des MMAs et des jeux de rôles pour faciliter les interactions entre différents acteurs et permettre des décisions collectives plus éclairées [Barreteau et al., 2003, Étienne, 2013]. Cette approche s'est principalement illustré dans les domaines de la gestion des ressources dans des contextes socio-écologique [Garcia et al., 2020, Hossard et al., 2022, Gurung et al., 2022, Jahel et al., 2023, Barbier et al., 2024].

3.1.2 Intérêt de l'approche participative en biologie

Dans le cadre de cette thèse, notre méthodologie s'est inscrite dans le paradigme de la modélisation participative. Comme mentionné précédemment, il n'y a pas, à notre connaissance, de ressources sur l'application de ce genre de méthodologie dans le cadre de la biologie cellulaire. Il nous semble, cependant, que la méthodologie participative peut faciliter les collaborations entre modélisateurs et biologistes. En effet, ce genre de modèles nécessite la collaboration entre des experts biologistes et des modélisateurs qui sont rarement expert des phénomènes à modéliser. Il faut donc que le savoir et les connaissances des biologistes soient transmis de la manière la plus efficace et juste aux modélisateurs afin que le modèle soit le plus fidèle à leurs attentes. De plus, en fonction de la complexité des phénomènes, le projet peut s'étendre sur des durées assez longues (mois, années), il est donc important de pouvoir minimiser la frustration afin que la collaboration puisse perdurer suffisamment longtemps pour obtenir des résultats satisfaisants.

Le fonctionnement typique d'un projet de modélisation est constitué de boucles itératives où chaque réunion est l'occasion de donner une nouvelle direction et de nouveaux objectifs au modèle (voir Figure 3.1). Seulement, entre chaque itération, il peut se passer plusieurs semaines, le temps qu'un nouveau prototype soit développé, au cours desquelles l'interaction avec les biologistes sera minime pouvant ainsi mener à des méconceptions causées par des incompréhensions et/ou malentendus de la part du modélisateur. Ces erreurs pourraient être limitées par la présence d'un biologiste éclairé sur le fonctionnement du modèle au cours de son développement.

L'objectif d'une méthodologie participative dans ce contexte, est de réduire le temps entre chaque réunion et l'introduction d'erreurs dues à des incompréhensions lors du développement. Pour cela, il est important de maintenir les

1. https://www.commod.org/

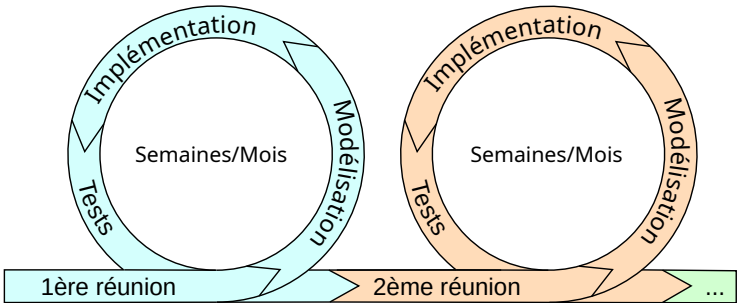


FIGURE 3.1 – **Cycle de vie typique d'un projet de modélisation.** Chaque projet de modélisation commence par une première réunion au cours de laquelle les biologistes présentent leur travail et les phénomènes étudiés et discutent avec les modélisateurs de leurs besoins et de leurs attentes à l'égard du futur modèle. Ensuite, les modélisateurs synthétisent ce qu'ils ont compris de la réalité biologique, développent un premier prototype et, après avoir conçu un premier prototype fonctionnel, une deuxième réunion est programmée pour discuter du modèle et l'améliorer. Ce processus est itératif jusqu'à l'abandon ou l'obtention d'un modèle suffisamment satisfaisant. Chaque boucle peut prendre des semaines ou des mois en fonction de la complexité du modèle, des malentendus et des calendriers.

experts biologistes proactifs dans le développement en leur proposant des outils leur rendant la compréhension du modèle plus accessible afin qu'ils soient en mesure de déceler les erreurs d'eux-même au moment de la conception. La Figure 3.2 résume le déroulement de notre méthodologie.

3.2 Mise en place d'une collaboration

En nous appuyant sur notre modèle de crypte intestinale, nous allons à présent détailler les différentes étapes de notre méthodologie permettant d'obtenir un modèle fonctionnel et répondant aux attentes spécifiées en amont du développement du modèle. Ce fil rouge nous permettra d'illustrer avec un exemple concret les différentes étapes de notre méthodologie.

Avant de débuter toute collaboration avec des biologistes, il est important de bien définir au plus tôt quelles sont leurs attentes et leurs objectifs vis à vis du modèle qui sera produit. Un des intérêts de cette méthodologie est de minimiser la frustration et pour cela le dialogue doit être entamé en amont afin que les différents protagonistes aient déjà une idée des possibilités à laquelle peuvent aboutir le projet et comment celui-ci va être mené.

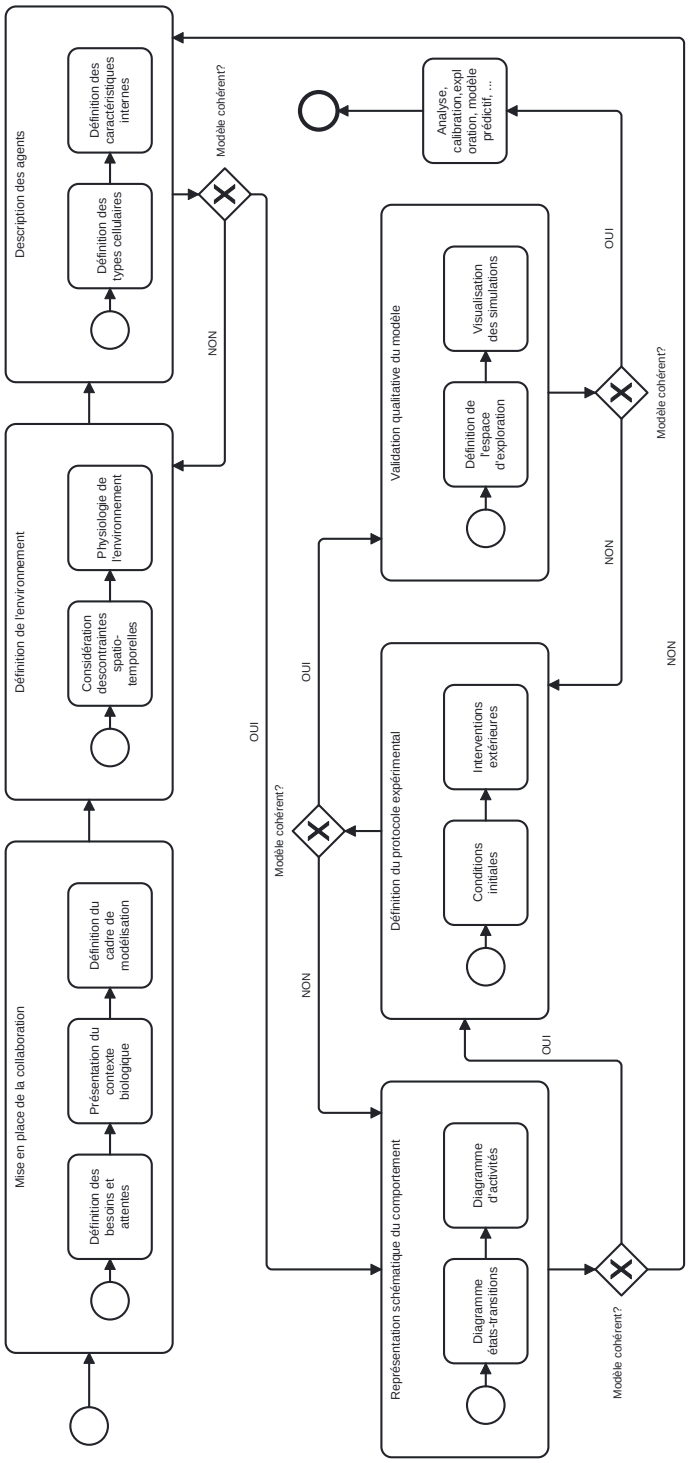


FIGURE 3.2 – **Schéma récapitulatif de notre méthodologie.**

3.2.1 Une méthodologie itérative

La méthodologie que nous employons est un processus itératif. L'idée est d'obtenir assez rapidement (atelier d'une journée environ) un premier prototype exploitable afin de le complexifier au fur et à mesure selon les besoins et les nouvelles hypothèses énoncées. Chaque séance de travail avec les biologistes est une occasion de revenir en arrière sur certaines décisions afin de modifier le modèle et ainsi tester de nouvelles hypothèses. Ces itérations successives permettent également aux différents partis prenants de s'approprier d'autant plus le modèle qu'il est nécessaire d'avoir un minimum de compréhension de celui-ci pour proposer des corrections ou formuler de nouvelles hypothèses.

Il est important de savoir, en amont du développement, le temps que chacun pourra allouer au développement du modèle pour définir des objectifs réalistes dans le temps imparti. Notre approche permet de complexifier et raffiner le modèle au fur et à mesure des itérations, il est donc pertinent de définir des objectifs à plus ou moins longs termes afin d'obtenir des prototypes intermédiaires qui répondront déjà partiellement aux attentes. Cela a pour objectif de diminuer la frustration au cours du projet s'il advenait à ne pas pouvoir se poursuivre suffisamment longtemps pour combler l'ensemble des attentes prédéfinies.

Modèle de crypte intestinale

L'illustration de notre méthodologie au travers de notre modèle de crypte intestinale sera représentée au travers de ces cadres verts. Ce modèle est plus amplement détaillé dans la section 6.1.1. Nous nous arrêterons à l'obtention d'un premier prototype et évoquerons des pistes de complexification.
Ce modèle cherche à reproduire le mécanisme de renouvellement de l'épithélium intestinale. Ce phénomène se produit à l'intérieur de cavités présentes le long de la paroi intestinale et est engendré par la migration et la différentiation de cellules au sein de ces cryptes.

3.2.2 Définition des besoins et des attentes

Avant de débuter le développement, il est donc nécessaire de savoir ce que les biologistes attendent du modèle qui résultera de la collaboration. Un modèle théorique ne nécessitera pas forcément de données pour être calibré et pertinent contrairement à un modèle qui aura pour objectif d'être prédictif et qui donc nécessitera d'avoir suffisamment de données expérimentales pour améliorer sa capacité à reproduire les phénomènes qu'il est censé simuler. Cette étape est primordiale pour définir le champ des possibles du modèle à venir, quitte à revenir dessus au cours du projet. L'objectif est de profiter des itérations successives pour garder un maximum de flexibilité tout au long de la collaboration.

En fonction des attentes, il se peut aussi que l'on dirige les biologistes vers d'autres paradigmes de modélisation. En effet, si l'objectif est, par exemple, d'obtenir une boîte noire permettant de prédire l'évolution du système à parti

des conditions initiales sans s'intéresser aux successions d'événements qui conduisent à une sortie ou à une autre, alors une solution à base d'apprentissage machine pourrait être plus appropriée. Il faut donc s'assurer, en amont, que notre méthodologie basée sur des systèmes multi-agents est pertinente vis à vis des attentes des biologistes, le but étant encore de minimiser la frustration.

Si un modèle basé agents semble pertinent pour la problématique étudiée, la question de la quantité de données disponible et les possibilités d'en obtenir de nouvelles pour calibrer et raffiner le modèle ainsi que les connaissances sur les phénomènes observés va se poser. De manière générale, la quantité et la qualité des données biologiques vont conditionner la qualité et la pertinence du modèle à développer.

Généralement, la liste non-exhaustive de questions suivante permet de débuter un projet dans les meilleures conditions :

— Quels phénomènes veut-on modéliser ?
— À quel question souhaite-t-on que le modèle réponde ?
— Est-il nécessaire de modéliser chaque cellule indépendemment les unes des autres ?
— Quelles sont les données déjà disponibles ?
— Est-il possible et pertinent d'obtenir de nouvelles données ?

Modèle de crypte intestinale

Dans le cadre de notre fil rouge, nous souhaitons modéliser les mécanismes spatialisés de division, migration et différenciation au sein d'une crypte intestinale. On cherche à reproduire la géométrie particulière d'une crypte, les effets de certaines molécules et le mouvement des cellules du fond vers le haut de la crypte (voir Figure 3.3). Pour obtenir ces dynamiques spatiales particulières des cellules de la crypte, il est intéressant de modéliser les différents processus en jeu depuis l'échelle individuelle des cellules. Nous disposons des données issus de l'article de Buske et al. [Buske et al., 2011]. Pour ce modèle, de nouvelles expérimentations pour obtenir de nouvelles données ne sont pas prévues dans l'immédiat et il faudra nous contenter de ce que l'on trouve déjà dans la littérature.

3.2.3 Présentation du contexte biologique

Après s'être assuré de la pertinence de notre méthodologie par rapport aux attentes de nos collaborateurs, il est intéressant que les biologistes présentent leurs travaux et donnent un aperçu des connaissances déjà établies sur les phénomènes à modéliser. Cette étape a pour but de vulgariser les concepts étudiés pour les modélisateurs afin que ceux-ci commencent à appréhender les différents types cellulaires, leurs comportements et interactions à modéliser. C'est aussi l'occasion pour les modélisateurs de poser des questions qui permettront d'un côté de renforcer leur connaissance des phénomènes biologiques en jeu et

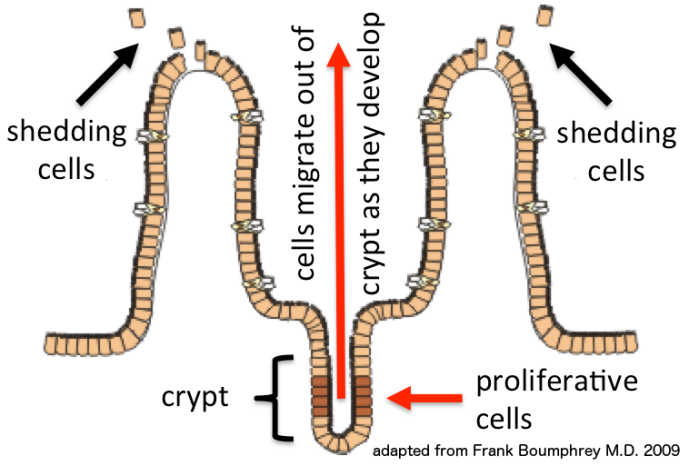


FIGURE 3.3 – **Schéma de l'épithélium intestinal.**
L'épithélium se renouvelle au sein des cryptes à l'intérieur des quelles on trouve des cellules prolifératives qui migrent vers le sommet de la crypte au fur et à mesure de leur développement. Creative Commons Attribution-Share Alike 3.0 Unported

de commencer à guider le développement du futur prototype de l'autre. L'idée sous-jacente est qu'une meilleure compréhension de la réalité biologique limite les erreurs de conception.

C'est à ce moment que l'on va discuter des principaux phénomènes en jeu qu'il sera nécessaire de prendre en compte dans la modélisation, ainsi que des hypothèses qui seront étudiées. C'est aussi l'occasion de déterminer les protocoles expérimentaux envisageables pour obtenir de nouvelles données pour la calibration du modèle. Cette étape constitue un travail préliminaire au développement du modèle pour commencer à orienter la conception.

Modèle de crypte intestinale

Pour notre modèle de crypte intestinale, il s'agit d'un modèle *in vitro* et *in vivo* fréquemment utilisé pour étudier les mécaniques de différenciation et de migration au sein des intestins des mammifères. On distingue quatre types cellulaires avec des comportements différents. Parmi ces types, certaines ont un comportement sécrétoire, certaines tendent à rester vers le fond de la crypte et d'autre se déplacent vers le sommet. Ces cellules adhèrent les unes aux autres tout en maintenant la structure de la crypte tout au long de leur migration. Les mécanismes de différenciation dépendent de la position de la cellule dans la crypte et de ses cellules voisines.

3.2.4 Définition du cadre de modélisation

De la même manière qu'à l'étape précédemment décrite, le but est de vulgariser la modélisation multi-agents dans le cadre de la biologie cellulaire aux biologistes. Ceci a pour but de rendre plus accessible et compréhensible le déroulement du développement et ainsi de permettre aux biologistes de mieux appréhender le modèle par la suite. L'objectif étant de faciliter un comportement proactif des biologistes en leur donnant les clefs de compréhension nécessaires.

C'est aussi l'occasion, de présenter le déroulement des séances de travail futures. Elles se décomposent en une succession d'étapes qui seront décrites dans les sections suivantes. Notre méthodologie se reposant notamment sur des diagrammes UML (états-transitions et d'activité, voir section 3.5), il est nécessaire de les introduire aussi en amont. Le but est que les biologistes comprennent l'intérêt de chaque étape et commencent déjà à réfléchir aux différents éléments qui vont être nécessaires à la réalisation de chaque étape. Informer les biologistes sur le déroulement des séances de travail permet de faciliter le dialogue pendant celles-ci.

Notre méthodologie doit répondre d'un côté aux exigences des biologistes mais aussi rester dans le cadre d'un MMA basé cellule. Pour cela, les différentes étapes de notre méthodologie vont permettre de déterminer les différents aspects de notre modèle tout en prenant en compte les considérations des biologistes. Nos étapes prennent en compte les aspects suivants :

- la définition de l'environnement et des mécanismes indépendants des cellules ;
- la description des cellules, leur type, leurs caractéristiques ;
- la description des comportements des cellules au travers de schéma ;
- la définition du protocole expérimental et des événements indépendants des cellules ;
- la validation qualitative du modèle par la visualisation de simulations ;
- les discussions sur les modifications potentielles du modèles.

Il faut garder à l'esprit que l'ordre des étapes prochainement abordée doit rester flexible et s'adapter en fonction des contextes, des itérations, des besoins, des hypothèses etc.

3.3 Définition de l'environnement

L'étape que nous abordons usuellement en premier est celle de la détermination de l'environnement, sa spatialisation autant que les règles physiques qui le gouvernent, dans lequel les cellules vont être plongées. En effet, il nous semble pertinent de commencer par définir les règles qui sont indépendantes de nos agents cellulaires car ce sont celles-ci qui vont principalement limiter les capacités computationnelles du modèle.

3.3.1 Considération des contraintes spatio-temporelles

Une des premières questions à se poser avant de définir nos agents est de déterminer la précision nécessaire au modèle pour être pertinent vis à vis de ses objectifs. Notamment, faire évoluer des cellules dans un environnement tridimensionnel est généralement plus coûteux que dans un environnement bi-, uni- ou a- dimensionnel ; la question se pose de la même manière avec un environnement continu ou discret. Il faut donc souvent se questionner à propos de la spatialisation de l'environnement cellulaire tôt dans le développement en essayant de choisir l'abstraction la moins coûteuse et la plus pertinente vis à vis des attentes du modèles. Cette réflexion permet de choisir, assez tôt dans le développement, la physique qui régira la disposition des cellules dans l'environnement et les unes par rapport aux autres.

De la même manière, il est intéressant de se poser assez tôt la question de la granularité temporelle adaptée aux phénomènes que l'on veut observer. Un pas de temps trop fin peut être inutilement coûteux mais un pas de temps trop large rend le modèle moins précis. A minima, quand il y a des données expérimentales, le pas de temps doit être inférieur à la fréquence d'acquisition des points expérimentaux. Il est intéressant aussi de discuter de la temporalité des événements du point de vue d'une cellule. Connaître le temps de réaction d'une cellule à des stimuli internes ou externes est un bon moyen de déterminer un pas de temps cohérent avec le comportement des cellules.

Ces différents choix d'implémentation vont entrer dans une balance entre temps de calcul, difficulté de calibration, complexité et précision de la simulation. Un modèle complexe est d'autant plus difficile à calibrer et entraînera des temps de calcul plus long, ce qui ralentira l'obtention de résultats et la visualisation. De l'autre côté, un modèle trop simple s'exécutera plus rapidement au sacrifice de sa précision et de sa capacité à reproduire finement les phénomènes étudiés.

Modèle de crypte intestinale

Pour notre modèle de crypte intestinale, une physique d'adhésion entre cellules basée sur des contacts Hertziens couplée à l'ajout de forces axiales permettent de maintenir la forme de la crypte tout en permettant la migration des cellules. Cette physique nous permet aussi de prendre en compte la pression interne de la cellule et donc d'intégrer des mécaniques d'inhibition de de contact. Quant à la question de la granularité temporelle, sachant que l'ordre de grandeur de la durée du cycle cellulaires de nos cellules est d'une quinzaine d'heures, il nous semble qu'un pas de temps d'une heure semble suffisant.

3.3.2 Physiologie de l'environnement

En plus des contraintes spatio-temporelles, il est parfois nécessaire de prendre en compte d'autres phénomènes physiologiques telle que la diffusion de molé-

cules dans un tissu ou la vascularisation. En fonction de l'abstraction, ces phénomènes qui structurent l'environnement et ses propriétés peuvent également être à l'origine d'une augmentation non négligeable du coût computationnel d'une simulation. Il est donc important de questionner leur pertinence assez tôt dans le développement pour permettre aux futures simulations de rester dans des plages de temps de calcul acceptables.

Ces différentes mécaniques environnementales pourront influencer et être influencées par le comportement des agents cellulaires. Ces choix de modélisations peuvent aussi entraîner la complexification du modèle et rendre la calibration et l'exploration plus ardue en rajoutant de nouveaux paramètres. Le dialogue est donc nécessaire pour ne pas complexifier inutilement le modèle.

Modèle de crypte intestinale

Dans le cadre de notre modèle fil rouge, il ne nous semble pas nécessaire d'utiliser un modèle de diffusion pour décrire des gradients de molécules soit globalement constant le long de la crypte, soit lié à des contacts avec des cellules sécrétoires. Ainsi, nous avons fait le choix de distinguer trois zones de la crypte selon l'axe base-sommet pour représenter le gradient de *Wnt* au sein de la crypte et de représenter la quantité de *notch* en fonction du nombre de cellules sécrétoires en contact avec la cellule concernée.

3.4 Description des différents agents

A partir de la première version de l'environnement cellulaire, la prochaine étape conseillée est celle qui consiste à définir les différents types cellulaires impliqués dans les phénomènes biologiques que l'on veut reproduire. On veut également, à cette étape, définir l'ensemble des attributs qui vont permettre de définir l'état et les caractéristiques internes de chaque agent.

3.4.1 Définition des types cellulaires

De la même manière que les cellules, en biologie, sont catégorisées en différents types en fonction de caractéristiques épigénétiques, nos agents cellulaires sont classés en différents types calqués sur ceux de la biologie. Distinguer ces différents types permet d'accorder à chacun des comportements cohérents avec les observations faites par les biologistes.

Déterminer les différents types d'agents nécessaires doit se faire dans le dialogue entre les modélisateurs et les biologistes afin de ne pas inutilement décrire trop de types différents. En effet, dans le cadre de l'abstraction qu'est un modèle, il n'est pas forcément nécessaire de distinguer deux types cellulaires différents si ceux-ci possèdent les mêmes caractéristiques et comportements. On peut alors regrouper ces types cellulaires au sein d'un même type d'agent englobant en fonction des besoins.

Modèle de crypte intestinale

Notre modèle fil rouge fait intervenir quatre types cellulaires aux comportements distincts : souches, Paneth, entérocytes et caliciformes. Les cellules souches sont capables de se différentier dans les trois autres types et se situent au fond de la crypte. Les cellules de Paneth sont des cellules sécrétoires situées au fond de la crypte qui ont un rôle anti-microbien. Les entérocytes sont des cellules qui migrent vers le haut de la crypte et qui permettent d'absorber les nutriments. Les cellules caliciformes sont des cellules sécrétoires migrant vers le haut de la crypte et qui permettent la lubrification de l'intestin.

3.4.2 Définition des caractéristiques internes

Dans un modèle basé agents, chaque agent est défini par un ensemble de paramètres décrivant ses caractéristiques internes et individuelles. Il faut donc définir cet ensemble assez tôt afin d'avoir d'un côté les paramètres qui permettront la comparaison avec les données expérimentales et de l'autre les caractéristiques internes nécessaire à la description et au fonctionnement des comportements qui seront ensuite implémentés. Chaque type d'agent pourra avoir des caractéristiques initiales différentes en fonction des cas.

Le dialogue est, à cette étape encore, important pour limiter le nombre de paramètres et éviter de prendre en considération des caractéristiques biologiques dont on ne peut obtenir de valeur dans un temps acceptable. Il est cependant possible d'obtenir des ordres de grandeur suffisants pour faire fonctionner le modèle en se basant sur les connaissances des biologistes. Il est nécessaire de questionner la pertinence de chaque paramètre afin de ne pas complexifier outre mesure le modèle.

Modèle de crypte intestinale

Dans le cadre de notre modèle, nos cellules disposeront des caractéristiques suivantes :

- les quantités de *Wnt* et de *notch*,
- les quantités seuils de *Wnt* et de *notch*,
- la durée de cycle,
- l'âge,
- la durée de vie maximale,
- la pression interne et la pression seuil (pour empêcher la division par inhibition de contact) de la cellule,
- la vitesse de déplacement des cellules,
- le rayon de la cellule.

  Le *wnt* et le *notch* correspondent à deux ensembles de protéines intervenant dans les voies de signalisation du même nom. Les unités de quantités de ces molécules sont arbitraires. La durée de cycle est tirée selon une loi normale à la naissance de la cellule et correspond au temps minimal nécessaire à la cellule pour se diviser. L'âge correspond au temps écoulé depuis la naissance de la cellule. La durée de vie maximale correspond au temps maximale d'existence d'une cellule dans la simulation, si son âge dépasse cette durée, elle meurt et est retirée de la simulation. L'âge et les durées de cycle et de vie sont mesurés en secondes. Les pressions interne et seuil correspondent à des valeurs en unités arbitraires de pression au sein de la cellule causé par les forces de répulsion des cellules voisines. La vitesse de déplacement des cellules correspond à la vitesse que peut atteindre une cellule sans prendre en compte les forces d'adhésion et de répulsion des cellules voisines et est mesurée en micromètres par seconde. Enfin le rayon de la cellule permet de déterminer le volume et la masse de la cellule et est mesuré en micromètres.

3.5 Représentation schématique du comportement

Généralement, après avoir défini l'environnement, les types d'agents et leurs différentes caractéristiques, nous conseillons de définir le comportement des cellules. Cette étape constitue le cœur de notre méthodologie et s'axe autour de l'utilisation de deux types de diagramme UML. Les diagrammes d'états-transitions permettent de décrire le comportement global des cellules et les diagrammes d'activité permettent de décrire la succession d'actions qu'effectuent les cellules dans les différents états. L'utilisation d'abstraction visuelle est un des outils recommandés pour pratiquer la modélisation participative.

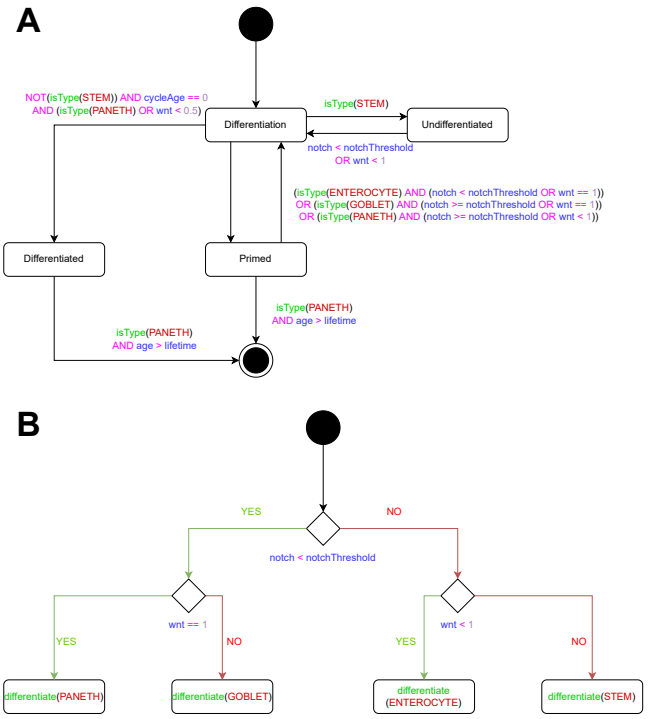


FIGURE 3.4 – **Diagramme état-transition (A) et diagramme d'activité (B) extraits du modèle de crypte intestinale. Le diagramme d'activité correspond à l'état de diférenciation du diagramme d'état-transition.** **A** : Dans un diagramme état-transition, un cercle plein correspond à la naissance d'un individu, un rectangle à un état, une flèche à une transition d'un état vers un autre et un cercle plein entouré à la mort d'une cellule. Selon les quantités de molécules auxquelles la cellule est soumise, une cellule souche peut se différencier en 3 types différents (entérocytes, Paneth ou caliciforme) ou rester indifférenciée si les conditions ne sont pas remplies. Une cellule amorcée peut encore se différencier si ses conditions changent. Après avoir rempli les conditions pour entrer dans l'état différencié, la cellule ne peut plus entrer dans l'état de différenciation. Les cellules amorcées et différenciées peuvent mourir si elles dépassent leur durée de vie sans se diviser ou sortir de la crypte. **B** : Dans un diagramme d'activités, un cercle plein correspond au point de départ, un losange à un choix binaire, un rectangle à une brique de comportement et les flèches indiquent l'ordre de succession. Les cellules en état de différenciation peuvent se différencier en 3 types différents ou rester/retourner à l'état de cellules souches. Si la quantité de notch est inférieure au seuil de notch, la cellule peut devenir l'un des types sécrétoires en fonction de la quantité de *Wnt* (Paneth ou caliciforme) ; sinon, si la quantité de *Wnt* est inférieure à 1, elle devient un entérocyte au lieu d'une cellule souche.

3.5.1 Diagramme états-transitions pour le comportement globale

Les diagrammes états-transition (voir Figure 3.4.A) sont utilisés depuis plusieurs décennies et permettent de décrire le comportement d'automates indépendants les uns des autres. Ils sont largement utilisés dans l'ingénierie logicielle pour comprendre les besoins des clients dans le cadre de la spécification UML (Unified Modeling Language) [Harel, 1987, Jacob, 1985]. Ces diagrammes peuvent être facilement traduits sans erreur en code informatique et ont l'avantage d'être facilement compréhensibles par des non-spécialistes.

Dans le cadre de notre méthodologie, ces diagrammes sont utilisés pour décrire les différents états phénotypiques dans lesquels peuvent entrer les cellules en fonction de leurs conditions internes et de leur environnement direct. Ces états peuvent s'assimiler à des expressions phénotypiques différentes des cellules que l'on peut expérimentalement observées ou être des abstractions utiles dans le cadre du modèle. On peut ainsi, par exemple, décomposer le diagramme selon les phases du cycle cellulaire ou de changement de comportements liées à l'évolution des conditions internes et externes des cellules. A chaque pas de temps, une cellule exécutera le comportement correspondant à son état actuel et pourra ensuite changer d'état en fonction de l'évolution des conditions environnementales et/ou internes.

Le dialogue doit permettre ici de mettre en lumière des états correspondant à des réalités biologiques observables ainsi qu'à segmenter la réflexion sur le comportement des cellules en fonction des situations dans lesquelles elles vont se trouver. Cette étape permet d'obtenir un diagramme qui donnera une vue d'ensemble des différents contextes auxquels une cellule peut être confrontée avant de s'intéresser aux comportements concrets que les cellules vont adopter dans chacune des ces circonstances.

Modèle de crypte intestinale

Pour notre exemple, le diagramme état-transition (voir Figure 3.4.A) est composé de quatre états :

- Différenciation, qui correspond à l'état où une cellule peut se différencier et changer de type en fonction des quantités de *Wnt* et de *notch* ;
- Différencié, qui correspond à l'état des cellules définitivement différenciée ;
- Amorcé, qui correspond aux cellules qui ont amorcé un processus de différentiation mais qui est encore réversible ;
- Indifférencié, qui correspond aux cellules souches ne s'étant pas différenciée.

et de huit transitions :

- sans condition depuis l'initialisation vers l'état Différenciation ;
- de Différenciation vers Différencié, si la cellule n'est pas souche et est haut de la crypte ou une cellule Paneth ;
- de Différenciation vers Indifférencié, si la cellule est souche ;
- de Différenciation vers Amorcé, si les deux conditions précédentes ne sont pas remplies ;
- d'Indifférencié vers Différenciation, si les quantités de *notch* ou de *Wnt* diminue en dessous de leurs seuils ;
- d'Amorcé vers Différenciation si les quantités de *notch* ou de *Wnt* ne sont plus cohérentes avec le type de la cellule ;
- de Différencié ou d'Amorcé vers la mort de la cellule, si la cellule est de type Paneth et qu'elle dépasse sa durée de vie maximale.

Ce diagramme est issu de la compréhension du fonctionnement du modèle développé dans l'article original.

3.5.2 Diagrammes d'activités pour le comportement dans chaque état

Les diagrammes d'activités (voir Figure 3.4.B) sont aussi utilisés depuis plusieurs décennies et permettent de décrire une succession d'actions et de conditions [Dumas and Ter Hofstede, 2001, Arora et al., 2017]. De la même manière, ils ont l'avantage d'être facilement convertibles en code informatique et constituent un outil efficace pour décrire le comportement des cellules avec des biologistes qui n'ont pas de connaissances préalables de ces diagrammes.

Dans le cadre de notre méthodologie, l'objectif principal de ces diagrammes est de décrire visuellement le comportement des cellules dans chacun des états du diagramme états-transitions présenté dans la section précédente. Les briques de comportements correspondront à des fonctions que le modélisateur devra implémenter ensuite pendant le développement. Les activités du diagramme correspondront à des actions les plus élémentaires possibles (telles que la croissance, le déplacement aléatoire ou la division) que l'on peut observer expérimentale-

ment. Le comportement décrit par un diagramme d'activité doit correspondre à l'ensemble des actions qu'effectuent une cellule durant un pas de temps de simulation.

Cette étape permet de rentrer dans le détail de chaque état du diagramme d'états-transitions. Cela permet de se poser la question de la pertinence de chaque état ou de devoir en créer de nouveaux car de nouveaux contextes auront été mis en évidence. Cette approche permet aux biologistes de réfléchir aux comportements des cellules dans des conditions particulières et ainsi de plus facilement formuler des hypothèses en fonction des contextes.

Modèle de crypte intestinale

Pour notre exemple, nous avons pris le diagramme d'activité de l'état Différenciation de notre modèle de crypte intestinale (voir Figure 3.4.B) composé de quatre activités correspondant à des différenciation en un des quatre types cellulaires. et de trois conditions :

- la première condition porte sur la quantité de *notch* ;
- si la quantité de *notch* est inférieure au seuil, la cellule se différenciera en Paneth si la quantité de *Wnt* est égale à 1 et en cellule caliciforme sinon ;
- sinon la cellule se différenciera en entérocyte si la quantité de *Wnt* est strictement inférieure à 1 et en cellule souche sinon.

Ce diagramme est issu de la compréhension du fonctionnement du modèle développé dans l'article original.

3.6 Définition du protocole expérimental

Après avoir décrit et détaillé les comportements des agents cellulaires, nous conseillons de faire de même avec le protocole expérimental. Pour cela, nous utilisons également des diagrammes d'activités pour modéliser la mise en place de l'expérience et des potentielles manipulations ponctuelles qui surviendraient tout au long de l'expérimentation. Les activités de ces diagrammes correspondront à des interventions qui pourront influencer l'ensemble des cellules ainsi que l'environnement.

3.6.1 Conditions initiales

Pour initialiser une simulation, il est nécessaire de décrire la mise en place des cellules et de l'environnement qui constitueront l'état initial du modèle. Le diagramme d'activité devra s'atteler à décrire la disposition des cellules dans l'espace (si la simulation est spatialisée) et de l'environnement ainsi que l'initialisation de leurs différents paramètres. Il s'agira donc de définir l'ensemble des fonctions à implémenter pour obtenir l'état initial d'une simulation.

L'idée de cette étape est de se poser les questions sur les premières étapes du protocole expérimental et sur la manière dont les expériences sont initiali-

sées. Il s'agit d'une occasion pour réfléchir aux différents bruits dus à des biais expérimentaux (que ce soit au niveau des outils de mesure employés que de la variabilité naturelle dans la population de cellules ensemencées) et donc prendre en compte ce bruit dans l'implémentation de l'initialisation. De manière générale, c'est une étape qui permet de prendre du recul sur la variabilité expérimentale due aux conditions initiales des cultures.

Modèle de crypte intestinale

Dans notre modèle exemple, les cellules sont positionnées le long de la structure de la crypte. Au premier pas de simulation, toutes les cellules sont de types *souche*, au pas suivant l'état Différenciation modifiera le type des cellules. Les cellules sont désynchronisées de sorte que les cellules n'aient pas toute le même âge ni la même durée de cycle.

3.6.2 Interventions extérieures

Pour reproduire et prendre en compte les manipulations de la culture ayant lieu après le début de l'expérimentation, notre méthodologie intègre la possibilité de les modéliser. De manière strictement similaire à l'initialisation, nous utilisons un diagramme d'activité pour décrire les potentiels interventions ou événements (ajout de cellules ou molécules, rafraîchissement du milieu, marquage de cellules, etc.) qui peuvent se produire après le début de la simulation. Ce diagramme pourra décrire autant des événements qui ont lieu à chaque pas de temps qu'à des moments précis de la simulation.

Encore une fois, cette étape est l'occasion de réfléchir aux différents aspects du protocole expérimental qui peuvent introduire du bruit, et donc de la variabilité indépendante des cellules, sur les résultats des expériences. Toutes ces réflexions doivent nourrir la prise de recul sur la précision que le modèle pourra obtenir au regard de la qualité des données et des expérimentations que les biologistes sont capables de fournir.

Modèle de crypte intestinale

Dans le cadre de notre modèle de crypte intestinale, la seule intervention extérieure modélisée est celle qui consiste à retirer les cellules sortant les de la crypte par le sommet afin de ne plus les prendre en compte dans la simulation.

3.7 Validation qualitative du modèle

L'avantage d'utiliser de la simulation multi-agents d'autant plus lorsque les agents sont spatialisés est qu'il est facile de visualiser chaque agent et leur caractéristique au cours de la simulation. Le but est ici d'utiliser des outils visuels pour permettre aux biologistes de qualitativement valider le modèle obtenu.

3.7.1 Définition de l'espace d'exploration

Avant de visualiser une simulation et de s'assurer du bon fonctionnement du modèle, il faut définir des ordres de grandeurs des différentes entrées afin d'obtenir un jeu de paramètres le plus biologiquement cohérent possible. Ce jeu de paramètres par défaut pourra ensuite servir de base pour de futures explorations.

Cette étape est l'occasion de faire un travail préliminaire sur les explorations et calibrations futures du modèle. Un modèle capable de reproduire des résultats expérimentaux à partir de jeux de paramètres biologiquement cohérent peut devenir un modèle suffisament robuste pour être prédictif dans le cadre qu'on lui a défini. De plus, des résultats pertinents indiquent que les hypothèses du modèle sont a minima plausibles et permettent donc de renforcer le socle de connaissances sur le système biologique étudié. Il est donc important de questionner assez tôt dans la collaboration quelles sont les paramètres dont on connaît déjà la valeur, ceux qu'il va falloir déterminer par l'exploration et la calibration et enfin ceux que les biologistes pourront obtenir a posteriori avec de nouveaux protocoles expérimentaux.

Modèle de crypte intestinale

Pour notre modèle fil rouge, les paramètres sur lesquels on va pouvoir interagir avec le modèle sont les suivants :

- Au niveau des cellules :
  - leur durée de cycle,
  - leurs différents seuils (pression, *Wnt*, *notch*, durée de vie),
  - leur rayon,
  - leur vitesse.
- Au niveau de l'environnement :
  - la profondeur de la crypte,
  - les forces axiales de maintien de la crypte,
  - les longueurs des trois zones de *Wnt*.

Ces différents paramètres permettent d'explorer des hypothèses et de changer des caractéristiques intrinsèques du modèle.

Ce travail en amont de la validation permet d'éprouver le modèle avec des paramètres cohérents avec les attentes du modèle.

3.7.2 Visualisation des simulations

Visualiser le comportement et l'évolution des caractéristiques de chacun des agents dans le temps permet de s'assurer d'un côté que le modèle fonctionne comme attendu et de l'autre de permettre aux biologistes d'apporter un regard critique sur le déroulement d'une simulation. L'idée est que cette étape de validation a pour objectifs de décider de la marche à suivre pour le modèle. En effet, en fonction des attentes le modèle pourra être complexifier ou corriger.

En fonction des outils à disposition et des modèles, il peut être intéressant de disposer d'outil permettant d'obtenir des graphiques en temps réels de certains paramètres de la simulation. En discutant avec les biologistes, il est possible d'obtenir des sorties qui sont plus usuelles pour eux. C'est aussi l'occasion d'explorer manuellement le modèle et de vérifier que les dynamiques sont cohérentes avec ce qui est attendu. Nous présenterons de tels outils dans le chapitre suivant.

Modèle de crypte intestinale

Avec notre modèle fil rouge, les caractéristiques des cellules (en plus de leur visualisation dans un espace tridimensionnel) que l'on veut visualiser sont les suivantes :

- leur type et leur état,
- leur âge,
- leurs quantités de *Wnt* ou de *notch*,
- leur pression interne.

La visualisation spatialisée des cellules permet de vérifier que les cellules s'organisent bien selon la forme de la crypte et se déplacent de manière cohérente par rapport à leur type. La capacité à afficher pour chaque cellule ces différentes caractéristiques permet de vérifier la cohérence du modèles vis-à-vis des choix d'implémentation qui ont été fait.

Cette étape permet, encore une fois, de rendre les biologistes proactifs dans la validation du modèle en leur faisant mobiliser leur connaissances sur la réalité biologique qu'ils observent. Cette méthode permet de minimiser la frustration en mobilisant des affects positifs d'accomplissement en disposant d'un modèle fonctionnel et compréhensible.

3.8 Devenir du modèle

Un modèle, ayant été qualitativement validé par les biologistes, pourra avoir différents destins.

Premièrement, en revenant sur les étapes précédentes, le modèle peut être raffiné et/ou complexifié afin de prendre en compte de nouveaux phénomènes ou de tester de nouvelles hypothèses. La méthodologie est suffisamment flexible pour permettre de retourner à n'importe quelle étape de conception et de constater visuellement les conséquences des changements effectués.

Le modèle peut également entrer dans une phase d'exploration plus ou moins poussées afin de mieux comprendre les différentes dynamiques sur les sorties en fonction des paramètres d'entrée. Cela peut avoir pour but de mieux comprendre les relations intrinsèques entre les différents paramètres ou alors à connaître l'espace de validité du modèle par rapport aux connaissances qu'ont les biologistes des phénomènes que l'on veut reproduire.

Si le modèle est considéré satisfaisant en l'état, il peut devenir un modèle théorique, qui pourra servir d'outil pédagogique par exemple, ou un modèle

prédictif dans les limites lesquelles il aura été calibré. Le principal but est de combler au maximum les attentes des collaborateurs biologistes.

3.9 Conclusion préliminaire

La méthodologie décrite tout au long de ce chapitre repose sur des méthodes de travail collaboratives issus de la modélisation participative. À l'aide de schémas, d'outils de visualisation et d'un cadre structuré et généralisé de nos modèles agents, nous avons pu développer une méthodologie permettant de répondre à la fois aux contraintes liés à l'architecture même de nos modèles ainsi qu'aux objectifs de minimisation de la frustration et des mésinterprétations entre biologistes et modélisateurs. Nous estimons que notre méthodologie permet de grandement faciliter le dialogue entre modélisateurs et biologistes et contribue à une meilleure compréhension du modèle par ces derniers. Cette meilleure compréhension du modèle entraîne une meilleure appropriation du modèle par les biologistes et donc une plus grande acceptabilité et confiance de leur part dans le modèle. C'est pour appuyer cette méthodologie que l'ensemble des outils de cette thèse ont été développés.

Bien que notre méthodologie ait été décrite de manière très linéaire, il est important de garder à l'esprit qu'il est possible à chaque phase du développement de retourner sur une des étapes précédentes.

Le cadre posé par notre méthodologie peut aussi très bien être appliqué dans d'autres cadres de modélisation multi-agents en utilisant des procédés similaires. De plus, la généralisation de l'emploi de diagrammes UML pour décrire le comportement des agents pourrait constituer une bonne pratique pour faciliter la reproductibilité des modèles publiés.

Dans le chapitre suivant, nous détaillerons l'emploi des différents outils développés et comment ceux-ci facilitent l'application de notre méthodologie et maximisent l'interaction entre modélisateurs et biologistes.

Chapitre 4

ISiCell : Une plateforme au service de la méthodologie

Sommaire

Dans le chapitre précédent, nous avons détaillé les tenants et aboutissants de notre méthodologie participative. Dans celui-ci, nous aborderons les outils que nous avons développés dans le cadre de la mise en place de cette méthodologie avant de nous intéresser à leur implémentation dans le chapitre suivant. Les outils développés pour la mise en place de notre méthodologie ont fait l'objet d'une publication dans un conférence internationale [Cogoni et al., 2024]. Nous nous appuierons encore sur notre modèle fil rouge de crypte intestinale pour illustrer les différents outils présentés.

4.1 Introduction à ISiCell

ISiCell est une plateforme web qui a été développée au cours de ce travail de thèse en collaboration avec Dr. David Bernard en contrat post-doctorale au sein de notre équipe. Elle constitue une application directe et un outil de la méthodologie précédemment expliquée. Ses objectifs sont les suivants :

- **Maximiser l'interactivité**, par l'utilisation de diagrammes afin de disposer d'un support graphique qui facilite le dialogue.
- **Accélérer le développement de modèles**, par la généralisation de l'architecture qui permet l'automatisation de la génération du code.
- **Permettre la validation qualitative de prototypes**, par l'utilisation d'outils de visualisation qui permettent aux biologistes de juger du bon fonctionnement du modèle par rapport à leurs attentes.
- **Permettre l'analyse quantitative et l'exploration des modèles**, en utilisant des algorithmes d'analyse, d'exploration et d'optimisation qui permettent de calibrer le modèle et de vérifier des comportements attendus du modèle.

ISiCell est composé de trois principaux outils (voir Figure 4.1) répondant aux objectifs précédemment énoncés :

- **ISiCell Builder**, qui est l'outil qui constitue l'interface principale côté client. Elle permet de construire les différents diagrammes et communique avec la partie serveur pour générer le code et lancer sa compilation. ISiCell Builder se repose sur des outils graphiques pour maximiser l'interactivité et permettre de dessiner directement les différents diagrammes de notre méthodologie dans l'interface du Builder.
- **ISiCell Viewer**, qui permet de visualiser les simulations à partir du code compilé en communicant avec la base de données que les simulations génèrent pendant leur exécution. L'interface du Viewer dispose d'outils de visualisation permettant de générer des graphes et de colorer les cellules visualiser en fonction de leur caractéristiques internes pendant l'exécution d'une simulation.
- **ISiCell Explorer et ISiCell Workbench**, qui permettent d'analyser le modèle en Python grâce à une version encapsulée du code généré pour permettre l'interfaçage entre les deux langages. Ces deux outils permettent de pousser l'étude du modèle après l'obtention d'un prototype satisfaisant.

Tous ces outils permettent et facilitent la mise en place de notre méthodologie tout en accélérant le développement des modèles et augmentant l'interaction entre les différents partis prenants du projet. De plus, le web permet à la plateforme d'être compatible avec tous les appareils disposant d'un navigateur récent et d'être accessible en ligne sur une machine plus performante. Cela nous permet également de faciliter le partage et la reproduction de modèles.

ISiCell a été conçu pour faciliter la collaboration entre modélisateur et biologiste mais également pour permettre la plus grande liberté possible aux modélisateurs. Ainsi, la plateforme ne sacrifie pas les possibilités des informaticiens

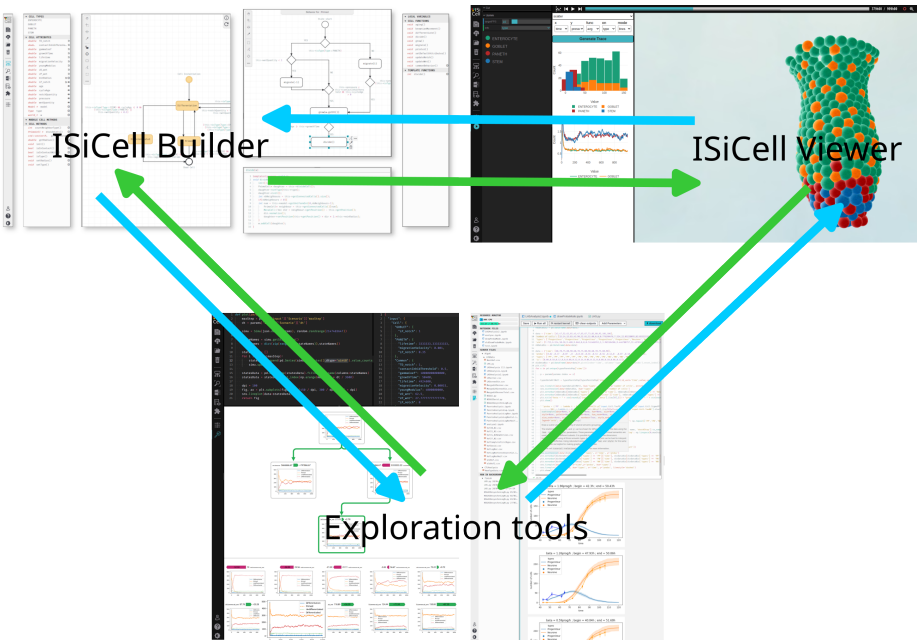


FIGURE 4.1 – **Vue générale des différents outils de la plateforme.** ISiCell peut être décomposée en 3 entités : ISiCell Builder, ISiCell Viewer et les outils d'exploration qui regroupent ISiCell Explorer et le système de notebooks intégrés. La plateforme permet de jongler entre ses différents outils pour améliorer de manière incrémental le modèle.

durant le développement et l'analyse des modèles.

Dans les sections suivantes, les différents éléments constitutifs de la plateforme vont être successivement présentés. Les grands principes de la plateforme seront abordés mais le fonctionnement plus détaillé de la plateforme le sera dans le chapitre 5.

4.2 ISiCell Builder : un outil de programmation visuel

Dans le but de permettre la construction des différents diagrammes de manière interactive et intuitive, ISiCell Builder a été créé. Il constitue l'interface principale de la plateforme et permet d'appliquer les premières étapes de notre méthodologie. Elle permet également d'accélérer le développement en offrant des outils de programmation visuelle et de génération automatique de code à partir de schémas. Afin de permettre la flexibilité voulue par la méthodologie, il est possible de naviguer entre les différentes étapes à n'importe quel moment

de manière fluide sans perdre ce qui a déjà été fait.

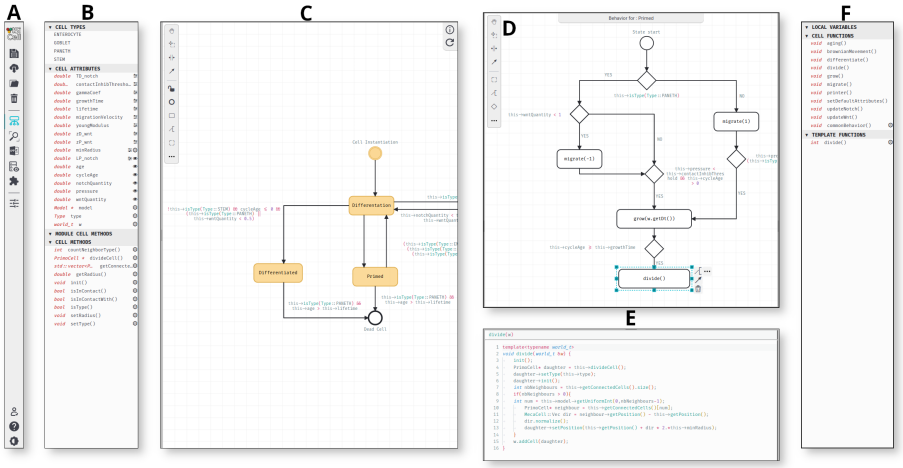


FIGURE 4.2 – **Interface d'ISiCell Builder**
ISiCell Builder est l'outil principal pour le prototypage des modèles avec les biologistes grâce à une interface facile à suivre. Cette figure représente la vue du Builder pour le modèle de crypte entérite mentionné plus tôt. De gauche à droite, la barre de menu principale (**A**) permettant de naviguer dans les différents outils ; les listes d'attributs (**B**), de méthodes et de types où vous pouvez ajouter de nouveaux types à la simulation et de nouvelles caractéristiques aux cellules ; le diagramme d'état (**C**), spécifiant tous les états par lesquels une cellule peut passer et les transitions entre ces états ; le diagramme d'activité de l'état amorcé (**D**) avec le code de la fonction de division écrite en C++ en dessous (**E**) ; les fonctions créées et les modèles génériques (**F**) qui peuvent être glissés et déposés dans le diagramme d'activité.

4.2.1 Mise en place d'un nouveau modèle

Pour définir l'environnement, il est possible de sélectionner des modules pré-implémentés issus d'anciens travaux depuis un menu accessible depuis la barre latérale (voir Figure 4.2.**A**). Ces modules permettent d'ajouter des nouvelles mécaniques au niveau de l'environnement ou des comportements des cellules, cela peut se traduire par l'ajout de nouveaux attributs et de nouvelles méthodes accessibles par les cellules et le protocole depuis les listes du panneau latérale (voir Figure 4.2.**B**). Il est également possible de modifier directement un module pré-existant ou d'en créer un nouveau directement depuis l'interface du Builder. Ces modules permettent d'accélérer le développement en offrant des briques de programmation fonctionnelles en amont du développement du modèle. De plus,

la possibilité de créer et modifier ces modules permet également une grande flexibilité pour le modélisateur. Ce système modulaire sera développé dans la section dédiée du chapitre 5.

Au niveau de la mise en place des agents cellulaires, l'interface propose d'ajouter des noms pour les types cellulaires qui constitueront les valeurs que l'attribut type des cellules pourront prendre dans la simulation, par défaut il existe un type nommé « *Common* ». Cela peut servir pour conditionner des comportements en fonction du type ou pour définir des sets de paramètres différents en fonction des agents. De la même manière, on peut définir et modifier le type (au sens informatique) et le nom des différents attributs de nos agents. Par défaut, les attributs ne peuvent être que de type entier, flottant ou booléen mais il est possible d'en prendre en compte d'autres en les ajoutant depuis l'interface dédiée. On peut aussi créer de nouvelles énumérations (au sens informatique), c'est-à-dire des listes de valeurs que peut prendre un paramètre, de manière similaire. L'interface permet de distinguer en fonction des attributs s'ils doivent être enregistrés dans la base de données pendant l'exécution d'une simulation afin de pouvoir être affichés et s'ils ont une valeur par défaut à l'initialisation des cellules. Cette valeur pourra alors être saisies dans l'interface dédiée et être différente en fonction des types cellulaires (voir section 4.3.1).

Ces différentes fonctionnalités sont accessibles depuis le menu latéral de la plateforme ; la liste des attributs est directement juxtaposée à ce menu (voir Figure 4.2.**A**).

4.2.2 Comportement et diagramme d'états-transitions

La partie centrale de l'interface du Builder (voir Figure 4.2.**C**) est dédiée à l'implémentation du comportement des cellules via un diagramme états-transitions. Cette interface permet d'appliquer directement notre méthodologie pour décrire le comportement général des cellules de manière interactive avec les biologistes.

Les états peuvent être glissés-déposés et des flèches peuvent être dessinées d'un état vers un autre pour signifier les transitions. Chaque état dispose d'un nom modifiable par un double clique et l'on peut spécifier les conditions de transition sur les flèches de la même manière. Les conditions sont à écrire en C++ et le builder dispose d'un outil d'autocomplétion permettant de faciliter leur écriture en proposant les attributs et méthodes déjà renseignés. Sauf cas particuliers, le code des transitions est suffisamment simple pour être intelligible par une personne n'étant pas familière avec la programmation. En effet, les transitions vont pour la plupart être conditionnées par des valeur seuils d'attributs et/ou par le type des agents.

4.2.3 Décomposition en diagrammes d'activités

Par un simple clique sur un des états du diagramme précédent, il est possible d'ouvrir une nouvelle interface permettant de dessiner le diagramme d'activités correspondant à l'état en question (voir Figure 4.2.**D**). De la même manière les

activités et les conditions (représentées par des losanges) peuvent être glissées-déposées et modifiées par un double clique. On peut également inverser les sorties « *YES/NO* » des losanges de condition, cela est principalement utile lorsqu'il n'y a qu'une seule sortie.

Il est possible de glisser-déposer des méthodes déjà pré-implémentées depuis la liste (à droite sur l'interface) contenant toutes les fonctions génériques déjà intégrées à la plateforme, celles étant fournies par les modules sélectionnés et enfin celles qui ont été créées pendant le développement du modèle. Toutes les fonctions présentes dans cette liste ne constituent pas forcément des activités dans les différents diagramme d'activité. En effet, on peut aussi appeler les méthodes créées dans une autre activité ou dans une condition du diagramme états-transitions.

Lorsque l'on clique sur une activité ou sur une fonction de la liste de droite, une interface d'édition du code C++ apparaît (voir Figure 4.2.**E**). Elle permet de modifier directement le code de la méthode en question, ce qui permet une grande flexibilité pour le modélisateur. Il faut cependant garder à l'esprit que notre méthodologie encourage à rendre ces méthodes les plus atomiques et simples possibles afin qu'elles soient plus compréhensibles pour un biologiste sans compétence en programmation. Dans le cas d'une activité du diagramme, la sélection permet également de stocker le résultat de la fonction dans une variable locale et/ou de spécifier les paramètres d'appel de la méthode.

En cliquant dans l'arrière-plan, on peut faire apparaître le code correspondant au diagramme d'activité de l'état que l'on est en train de dessiner. Ce code est généré automatiquement au fur et à mesure que l'on connecte les différentes activités et permet d'avoir une vision du code généré correspondant au diagramme. Enfin, il existe une fonction personnalisable nommée *commonBehavior* qui est appelée au début de chaque état. Cette fonction permet d'appliquer à chaque état le même comportement, cela peut s'avérer utile pour, par exemple, augmenter l'âge de toutes les cellules à chaque pas de temps ou pour mettre à jour des quantités de molécules avant chaque exécution de comportement des états.

4.2.4 Implémentation du protocole expérimental

Pour développer, le protocole expérimentale, ISiCell Builder dispose d'une interface similaire à celle du comportement. Dans cette seconde interface, le modélisateur peut spécifier l'initialisation de la simulation ainsi que la succession des événements indépendants des cellules. On peut ensuite naviguer entre l'implémentation du comportement des cellules et celle du protocole expérimental.

L'interface permettant de générer les diagrammes d'activité correspondant à l'initialisation des simulations et à la boucle d'application du protocole est sensiblement la même que pour les diagrammes d'activités des différents états, à la différence que les listes sont actualisées de sorte à ce que la liste d'attributs à gauche et celle de méthodes à droite correspondent au protocole. On pourra donc y trouver des attributs gérant le pas de temps et la durée de la simulation, des paramètres liés à l'environnement, des variables d'initialisations ainsi que des

méthodes permettant de modifier l'environnement ou d'interagir avec l'ensemble des cellules.

Le diagramme d'activité d'initialisation permet de mettre en place le protocole expérimental. On pourra ainsi créer les premières cellules, les disposer dans l'environnement et les initialiser comme l'on veut selon les besoins de la simulation. C'est aussi dans ce diagramme que l'on va initialiser toutes les variables liées à l'environnement et à sa gestion. Quant au diagramme correspondant à la boucle d'exécution de la simulation, il a pour objectif de mettre à jour et intervenir sur l'environnement et les cellules si besoin. Les interventions peuvent aussi bien être périodiques que ponctuelles en fonction de ce que l'on cherche à reproduire.

4.2.5 Génération de la simulation

Lorsqu'on lance la compilation du modèle depuis l'interface dédiées du Builder, la partie client génère un fichier JSON contenant l'ensemble des informations nécessaires au serveur pour générer le code. On retrouvera donc dans ce fichier le code des différents modules créés et/ou modifiés, l'ensemble des types cellulaires créés, les différents attributs ainsi que le code des méthodes ajoutés au cellules et au protocole, le code correspondant au comportement de chaque état, etc.

Ce fichier est ensuite utilisé par un ensemble de scripts Python côté serveur afin de créer l'arborescence du projet et générer le code C++ des différents fichiers à partir de modèles pré-existants. Ce processus sera expliqué plus en détails dans le chapitre suivant. Ces scripts déclenchent ensuite la compilation du code généré et les messages d'erreurs, s'il y en a, sont ensuite renvoyés côté client sinon le bouton de compilation devient un bouton de visualisation ou d'exploration en fonction du code généré.

Le fonctionnement plus détaillé de la communication entre les parties client et serveur du Builder ainsi que l'architecture du code des modèles générés seront développés dans le chapitre 5.

4.2.6 Facilitation du développement

Afin de faciliter la prise en main de la plateforme par les modélisateurs, ISiCell Builder dispose d'autres outils qui permettent d'améliorer l'expérience utilisateur lors du développement. Dans cette section, nous allons présenter ces outils.

Système d'éditeur de code

L'apport principal du modélisateur dans l'utilisation de la plateforme est lorsque du code doit être écrit. Afin d'accélérer ces phases et de faciliter le développement, nous avons intégré à la plateforme un système d'édition de code et d'auto-complétion. Cela permet de minimiser le temps allouer à ces phases pour maximiser les temps d'interaction avec les biologistes.

Notre système d'édition de code est basé sur l'éditeur de code de Microsoft Monaco [1] qui est notamment utilisé dans Visual Studio Code. Cet éditeur est basé sur des technologies web et est donc facilement employable au sein d'une application basée web. L'éditeur que nous utilisons dispose d'outils d'auto-complétion qui s'adaptent en fonction du contexte où l'éditeur est instancié. Ainsi, si l'on est en train de modifier le code d'une méthode dans un diagramme d'activité d'un état, l'auto-complétion proposera les attributs et méthodes liés aux cellules et de la même manière, si l'on est en train de modifier une méthode liée au protocole, l'auto-complétion proposera des éléments liés à ce dernier. La coloration syntaxique basée sur des expressions régulières a aussi été améliorée par rapport à ce que propose Monaco de base afin de rendre l'utilisation plus agréable pour le modélisateur. Dans le cas du Builder la coloration est adaptée à l'écriture de code C++.

Dans le cas spécifique des diagrammes d'activité, l'éditeur sert également à afficher le code correspondant au diagramme (voir Figure 4.3). Cela permet au modélisateur de vérifier que le code du comportement de l'état est cohérent et d'en avoir une vision plus globale. La génération de ce code se fait à la volée à chaque modification du diagramme d'activité. L'algorithme de génération de ce code sera détaillé dans le chapitre suivant.

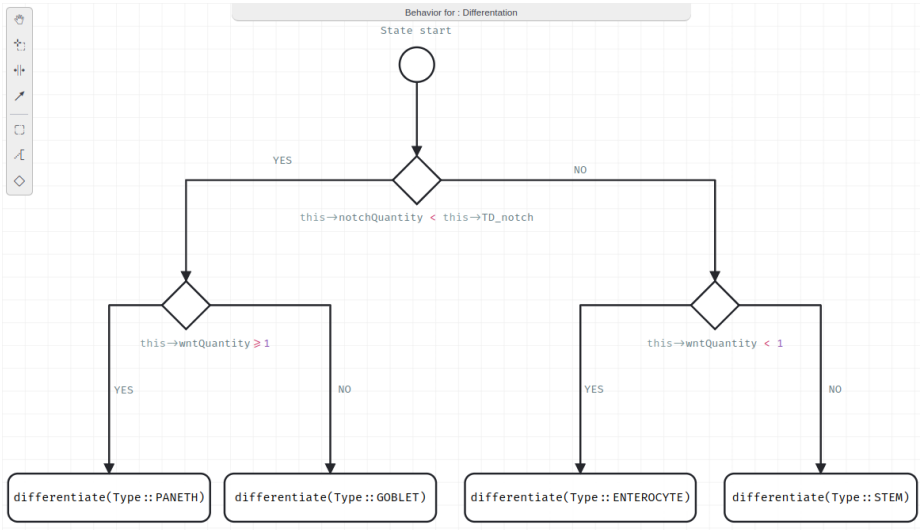
Les éditeurs de code utilisés dans les outils d'interaction Python présentés dans la section 4.4 sont aussi basés sur Monaco avec une coloration adaptée au Python.

Outil de recherche et gestion d'erreur

Pour plus facilement naviguer entre les différents éditeurs de code, ISiCell Builder dispose d'un outil de recherche qui permet de retrouver très facilement chaque mention de la chaîne de caractère recherchée et d'accéder directement à la boîte correspondante (flèche de condition, code d'une fonction, nom d'attribut ou de méthode, etc.). Cela permet de très rapidement retrouver la mention d'une caractéristique cellulaire ou l'usage d'une fonction et de les modifier.

De la même manière, grâce au système de balises présenté dans le chapitre 5, si une ou plusieurs erreurs se produisent pendant la compilation, l'affichage des erreurs permet d'accéder au code responsable en un simple clique. Cette fonctionnalité permet de rapidement corriger les erreurs pour relancer la génération et la compilation. Cette fonctionnalité permet de minimiser le temps passé à corriger les erreurs de compilation et permet donc de maximiser encore une fois le temps d'interaction avec les biologistes.

1. Site officiel de Microsoft Monaco Editor : https://microsoft.github.io/monaco-editor/




```
commonBehavior(w);
if(this→notchQuantity < this→TD_notch){
    if(this→wntQuantity ⩾ 1){
        differentiate(Type::PANETH);
    } else {
        differentiate(Type::GOBLET);
    }
} else {
    if(this→wntQuantity < 1){
        differentiate(Type::ENTEROCYTE);
    } else {
        differentiate(Type::STEM);
    }
}
```

FIGURE 4.3 – **Exemple de code généré à partir d'un diagramme d'activité.**
Le code correspond à l'état *Differentiation* du diagramme états-transitions du modèle de crypte entérite déjà mentionné. La méthode *commonBehavior* est appelée avant le code du diagramme quelque soit l'état.

4.3 ISiCell Viewer : un outil de visualisation pour la validation qualitative

ISiCell Viewer est l'outil permettant de visualiser les modèles qui ont été précédemment compilés. Vis à vis de notre méthodologie, cet outil permet de valider qualitativement et de manière interactive les modèles.

4.3.1 Sélection des paramètres

Avant de lancer une simulation, l'interface de sélection de paramètres permet de choisir les valeurs de toutes les variables à initialiser. Cela permet, d'une part, d'avoir une réflexion sur les ordres de grandeurs et les domaines de valeurs cohérents pour ces paramètres, et d'une autre part, de commencer à tester des hypothèses simples. Ces hypothèses peuvent être par exemple de vérifier que

l'augmentation ou la diminution d'un paramètre a bien l'effet escompté sur les dynamiques de la simulation ou d'activer/désactiver des mécaniques conditionnées par un booléen et d'observer les changements que cela induit ou pas.

Dans cette interface, il est possible de sauvegarder différents jeux de paramètres mais également différentes versions d'un même modèle (voir menu de gauche sur la Figure 4.4). En effet, cela permet de jongler facilement entre différentes versions d'un modèle pour tester de nouvelles hypothèses et conserver des versions plus anciennes. Les jeux de paramètres enregistrés pour une version peuvent facilement être réemployés d'une version à une autre. La plateforme dispose aussi d'une interface pour comparer ces différentes versions entre elles.

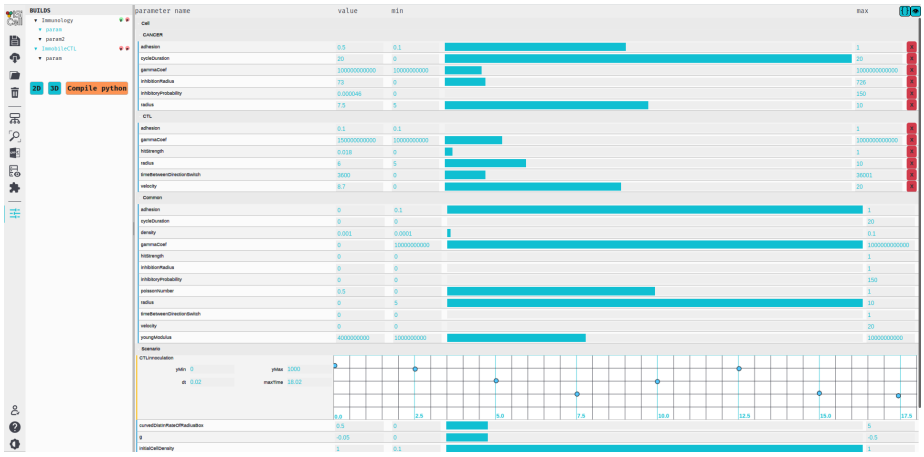


FIGURE 4.4 – **Sélection des paramètres sur le modèle d'interaction Cancer/CTL.**
Cette interface permet de choisir les différents paramètres d'une simulation avant de la visualiser. C'est cet outil qui permet aussi de naviguer entre les différentes versions d'un même modèle et de sauvegarder des jeux de paramètres.

4.3.2 Outils de visualisation et validation qualitative

Lorsqu'on lance la visualisation d'une simulation à partir d'un jeu de paramètres, l'interface change pour celle de visualisation (voir Figure 4.5). Cette interface dispose de différents outils permettant d'accéder à différentes informations sur la simulation de manière interactive et instantanée. En effet, on peut lister :

— la barre supérieure (voir Figure 4.5.**C**) permettant de d'avancer et remonter dans le temps de la simulation pour visualiser des moments précis ;
— le système de coloration (voir Figure 4.5.**B**) permettant de visualiser les valeurs de différents paramètres internes des cellules qu'ils soient continus

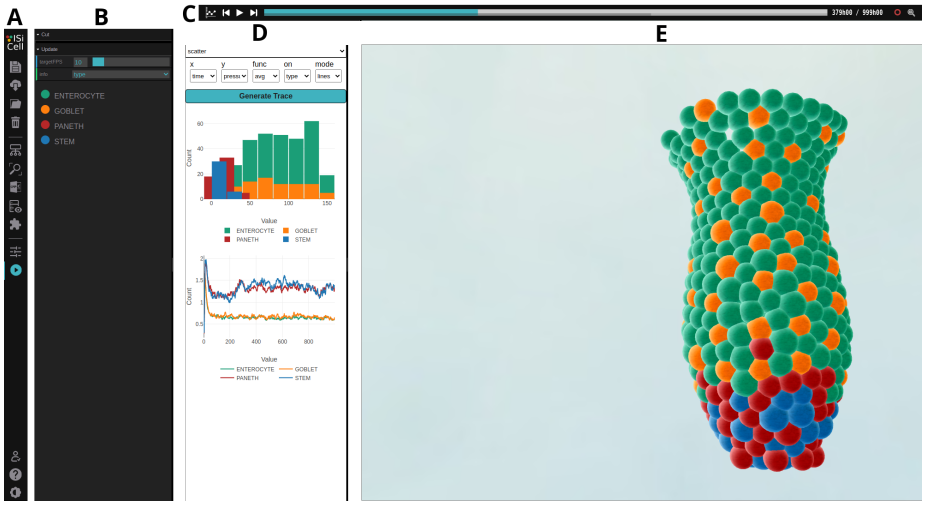


FIGURE 4.5 – **Vue d'ensemble de CellViewer sur un modèle de crypte intestinale.**
La barre supérieure (**C**) permet de naviguer d'avant en arrière dans le temps de la simulation. Sur le panneau de gauche (**B**), la coloration peut être modifiée. Le panneau blanc (**D**) permet la génération dynamique de graphiques. La barre latérale (**A**) permet de naviguer entre les différentes interfaces d'ISiCell. Enfin, la simulation est visualisable sur la vue principale (**E**) et il est possible de manipuler la caméra pour naviguer au travers des cellules et de réaliser des coupes.

ou discrets ;
— l'outil de création de graphiques (voir Figure 4.5.**D**) permettant de tracer différents diagrammes à partir des données des cellules et du protocole, spatialement ou temporellement au besoin.

Tous ces outils participent à faciliter l'évaluation qualitative par les biologistes. En effet, les différents graphiques et la possibilité de naviguer entre les différents moments permettent de facilement capter les dynamiques de la simulation. De plus, les outils de caméra offrent la possibilité de se déplacer et de changer l'orientation de la caméra afin d'observer les dynamiques spatiales du modèle. Enfin, les outils de coloration permettent de rendre compte que les différentes caractéristiques des cellules ne prennent pas de valeurs biologiquement aberrantes et que les comportements dans chaque état sont cohérents avec ce que l'on voulait.

4.4 Outils d'analyses quantitatives Python

ISiCell dispose également d'outils permettant d'explorer et/ou d'optimiser un modèle. Ces outils se décomposent en deux interfaces :

— ISiCell Explorer, qui permet d'explorer manuellement et assez facilement l'impact de différentes entrées sur les sorties du modèle ;

— ISiCell Workbench, qui permet d'analyser sans restrictions le modèle.

Tous ces outils utilisent la version encapsulée en Python de l'exécutable du modèle, ce qui permet l'interfaçage avec le langage Python et donc l'accès à tous les outils déjà existants dans ce langage.

4.4.1 ISiCell Explorer : un outil d'exploration interactif

Les premières versions de la plateforme ne permettait pas d'explorer les dynamiques du modèle avec les biologistes afin de vérifier certaines dynamiques attendues ou de tester rapidement des hypothèses assez simples. Les biologistes disposent généralement, plus d'informations sur le comportement du système à l'échelle de la population, il nous paraissait donc pertinent d'intégrer un outil à la plateforme permettant de facilement étudié les dynamiques du système. Cela nous a permis de disposer d'un nouvel outil de validation qualitative, de la même manière qu'ISiCell Viewer.

ISiCell Explorer correspond à l'interface d'exploration interactif (voir Figure 4.6). Depuis celle-ci, il est possible d'éditer un script Python permettant de lancer un certains nombre de réplicas de simulations en parallèle et de tracer un graphique pour chaque jeu de paramètres. Il suffit ensuite de sélectionner les paramètres que l'on veut explorer, les bornes minimales et maximales pour chacun et, enfin, le nombre de jeu de paramètres que l'on veut générer pour lancer l'exploration. Les jeux de paramètres seront générés selon un hypercube latin [Loh, 1996, Helton and Davis, 2003] dans l'espace des paramètres à explorer dépendant du nombre de jeux que l'on veut générer.

Pour chaque configuration, on verra apparaître au fur et à mesure le graphe agrégeant les réplicas dès qu'ils auront été calculés. En fonction des résultats, on peut choisir de repartir d'un jeu de paramètres qui semble plus satisfaisant au regard de ce que l'on veut obtenir puis relancer une phase d'exploration à partir de cette nouvelle configuration de référence. L'Explorer génère ainsi au fur et à mesure un arbre d'exploration, permettant de remonter ou redescendre dans l'arbre au fur et à mesure des explorations.

Cet outil permet d'explorer manuellement et de manière interactive les dynamiques du modèle plus finement qu'avec le Viewer. Si l'on veut ensuite visualiser le comportement induit par certains jeux, il suffit d'importer le jeu de paramètres voulu et de lancer une visualisation depuis l'interface de sélection de paramètres. En revanche, pour une analyse plus fine, il sera plus pertinent d'utiliser le système de notebooks, il sera cependant plus compliqué de garder de l'interactivité entre modélisateurs et biologistes comme nous allons le voir dans la section suivante.

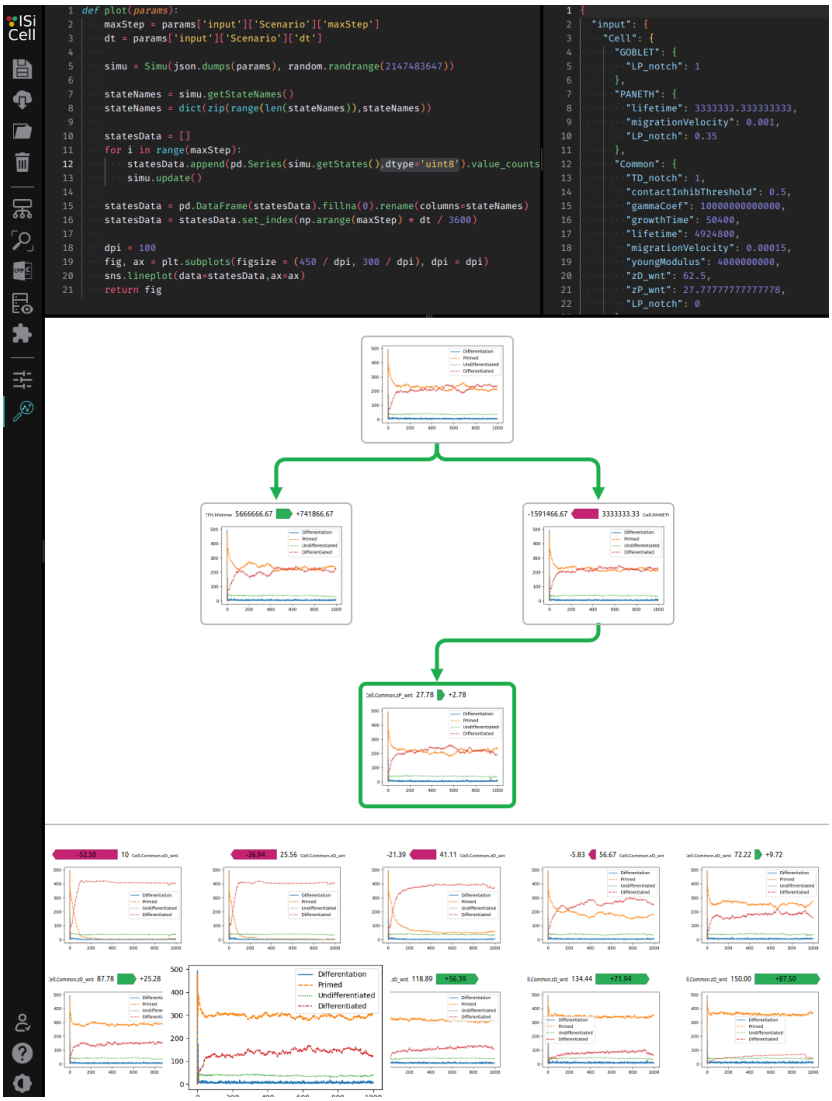


FIGURE 4.6 – **Vue de CellExplorer après quelques explorations de paramètres.**
Le panneau supérieur gauche contient le script Python qui lance les simulations et génère les graphiques. Le panneau supérieur droit contient les valeurs d'entrée au format JSON.

4.4.2 ISiCell Workbench : un outil d'analyse complet

ISiCell Workbench implémenté dans ISiCell s'adresse principalement aux modélisateurs et constitue un outil permettant de pousser l'analyse des modèles

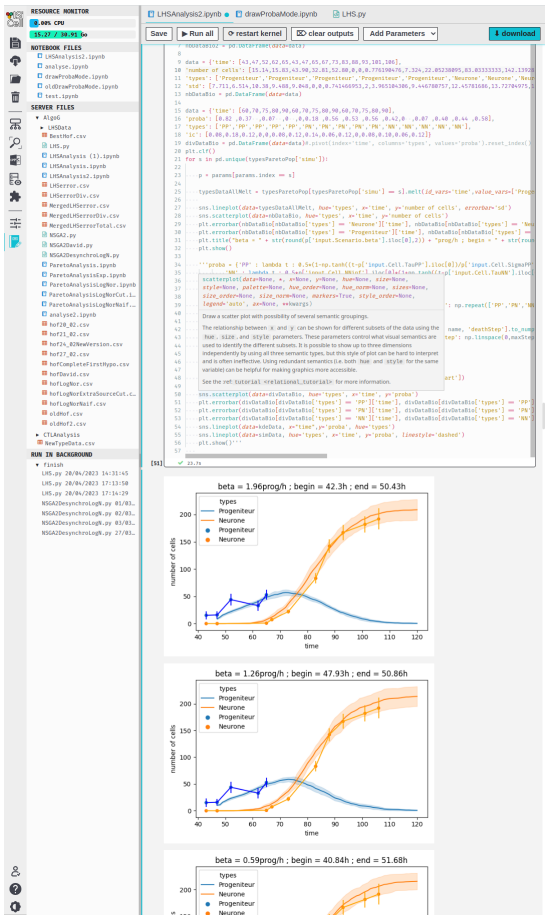


FIGURE 4.7 – **Interface des notebooks.**
Le menu à gauche permet de naviguer entre les différents notebooks, scripts Python et tout autres fichiers. Cet outil permet au modélisateur de facilement interagir avec le modèle et de l'analyser grâce aux outils fournit par les différentes librairies Python disponibles.

en dehors du cadre de notre méthodologie. En effet, cet outil repose sur le développement de scripts Python directement exécutable, ce qui permet d'utiliser l'ensemble des librairies disponibles pour ce langage et donc de plus finement analyser un modèle en exécutant des algorithmes d'exploration, d'optimisation, d'analyse de sensibilité, etc.

ISiCell Workbench permet l'utilisation de notebooks (voir Figure 4.7) qui ne permettent pas une interaction facilitée entre biologistes et modélisateurs car ils nécessitent un bagage technique plus grand et un temps de mise en place et d'exécution des scripts plus longs. Cet outil a été pensé pour l'étude du modèle

entre deux sessions de travail collaboratif. En effet, à la fin d'une session, il est fréquent que l'on discute des analyses intéressantes à faire pour le modèle dans son état actuel et que cela constitue le point de départ de la session suivante. Le temps entre deux sessions pouvant être long, cela donne le temps aux modélisateurs de déployer et d'exécuter différents algorithmes d'exploration ou d'optimisation pour analyser ou calibrer le comportement du modèle. Les résultats intermédiaires obtenus permettent également de minimiser la frustration entre deux sessions en montrant l'avancée progressive du modèle et les réponses qu'il permet d'obtenir ainsi qu'en ne limitant pas les possibilités du modèles vis-à-vis du modélisateur.

Cet outil facilite également le travail du modélisateur car il intègre et permet l'utilisation de l'exécutable encapsulé, donnant accès facilement aux résultats de simulation pas par pas tout en gardant la rapidité d'exécution du C++. Le langage Python dispose, de plus, d'un nombre conséquent de librairie dédiée à l'analyse [McKinney et al., 2011, McKinney, 2012] (algorithmes génétiques avec DEAP [De Rainville et al., 2012], analyses de sensibilité avec SALib [Herman and Usher, 2017], etc.) et à la création et visualisation de graphiques (avec Matplotlib [Ari and Ustazhanov, 2014, Yim et al., 2018] et seaborn [Waskom, 2021]) de tout type, ce qui en fait un environnement totalement adapté aux besoins du modélisateur. En plus des notebooks, il est aussi possible de lancer des scripts Python sur le serveur sur lequel la plateforme est déployé afin de lancer des analyses plus poussées. Les résultats peuvent ensuite être visualisés et analysés grâce aux notebooks.

Enfin, cette interface dispose d'un gestionnaire de fichiers. Chaque modèle peut avoir ses propres notebooks et scripts enregistrés et spécifiquement liés à lui pour pouvoir y accéder depuis n'importe quelle instance de la plateforme. Il est aussi possible de sauvegarder ces fichiers sur le serveur où une instance d'ISiCell est déployée afin de pouvoir y accéder depuis n'importe quel modèle et donc facilement réemployer du code précédemment développé. La plateforme prend également en charge la visualisation des fichiers au format CSV qui est un format fréquemment utilisé pour manipuler des données utilisées dans des tableurs. De plus, le modélisateur a également accès à un terminal shell, ce qui lui permet un encore plus grand contrôle sur sa session.

4.5 Outil de gestion de la plateforme

La plateforme pouvant être déployée en ligne sur un serveur, il a donc été nécessaire de réfléchir à la gestion des utilisateurs et aux questions de sécurité qui découle de son accès à distance par n'importe quelle personne. De plus pour faciliter le partage, nous avons réfléchi à la manière de partager nos modèles pour être exécuté sur d'autres instances de la plateforme.

4.5.1 Système de comptes utilisateurs

ISiCell est actuellement déployée sur un serveur et accessible depuis n'importe quel navigateur qui a accès à internet. La plateforme permet donc à plusieurs utilisateurs de l'utiliser simultanément, ce qui a fait émerger les problèmes suivants :

— Comment restreindre l'accès à la plateforme à des usagers malveillants ?
— Comment éviter que deux utilisateurs modifient le même fichier ?
— Comment limiter la capacité des utilisateurs à accéder directement au serveur ?

En effet, notre plateforme permet l'utilisation d'un terminal shell et l'écriture de code C++ et Python, il était donc primordial que nous adressions ces questions.

Pour limiter l'accès au reste du serveur, nous conseillons de déployer ISiCell au sein d'un conteneur docker [Potdar et al., 2020, Brady et al., 2020] qui limite les possibilités de sortir de cet environnement virtuel. Cela permet également de faciliter l'installation de la plateforme en fournissant un conteneur fonctionnel avec un système d'exploitation déjà éprouvé, ce qui empêche les problèmes de compatibilités.

De plus, nous avons mis en place dans la plateforme un système de compte utilisateur qui nécessite de demander à un administrateur la création d'un compte protégé par un mot de passe. Ceci nous permet de limiter les utilisateurs ayant accès au plein potentiel de la plateforme. Si un utilisateur n'est pas connecté, il ne peut pas modifier de schéma, ni compiler de code, ni accéder aux outils d'interaction Python. Cela limite grandement, les capacités des utilisateurs qui n'ont pas de compte.

Enfin chaque utilisateur connecté correspond à un utilisateur sur le système d'exploitation qui aura son propre dossier de travail auquel les autres utilisateurs n'ont pas accès. Un utilisateur connecté a ses droits limités à son dossier de travail et ne peut effectuer des actions nécessitant des droit d'administrateur. Cela assure que chaque utilisateur puisse gérer son propre dossier avec ses propres fichiers sans qu'un autre utilisateur puisse interférer avec. Chaque utilisateur dispose aussi de son propre environnement virtuel Python qui dispose déjà de plusieurs librairies par défaut. Il est ensuite possible d'installer ou désinstaller des librairies en fonction des besoin sans interférer avec l'environnement Python des autres utilisateurs.

4.5.2 Système de sauvegarde

Afin de pouvoir sauvegarder et réutiliser un projet de modèle ISiCell, la plateforme dispose de son propre format de fichier (extension ".isidiag") qui correspond à un fichier zippé contenant un fichier au format JSON. Ce fichier sérialise version par version, toutes les listes des attributs et méthodes (code compris) des cellules et du protocole, le code de chaque état, tous les diagrammes d'activité et celui d'états-transitions, les listes des jeux de paramètres, les modules créés et le script Python de l'Explorer. On retrouve également dans ce fichier l'ensemble des notebooks avec leur sorties créé pour le modèle, indépendemment

de la version.

Les modèles peuvent également être enregistrés directement sur le serveur et seront liés à l'utilisateur qui les a sauvegardé. Ainsi un utilisateur pourra naviguer entre les différents modèles qu'il a développé grâce à une interface dédiée. Il peut ensuite facilement récupérer localement et envoyer des modèles sur le serveur depuis cette interface en fonction de ses besoins.

De plus, il est possible d'enregistrer des modèles sur le serveur en tant que modèles démonstratifs. Ces modèles seront accessibles par n'importe quel utilisateur sans avoir besoin d'être connecté, seulement, ils ne pourront pas modifier le modèle et seulement regarder le code dans chaque état et lancer une exécution pour le visualiser. Pour les utilisateurs connectés, cela permet aussi de créer un nouveau modèle à partir d'un ancien modèle déjà existant.

Ce système a pour but de faciliter le partage de modèles et de rendre accessibles et reproductibles les modèles développés avec la plateforme.

4.6 Conclusion préliminaire

L'intégralité des outils développés pour la plateforme permettent et facilitent l'emploi de notre méthodologie participative. Ainsi, ISiCell Builder permet la définition du modèle à l'aide de diagrammes mais en plus, il permet la génération en direct du code correspondant à chaque étape. La plateforme accélère également le processus en permettant la génération et la compilation à la volée du code correspondant au modèle. L'exécutable obtenu peut directement être visualisé dans l'interface du Viewer, ce qui permet la validation qualitative et raccourcit considérablement le temps nécessaire à déployer et configurer un outil permettant de visualiser en direct nos simulations. Enfin les outils d'exploration permettent de rentrer plus finement dans l'analyse du modèle et permettent de répondre à des questions plus précises tout en donnant de nouvelles perspectives aux modèles entre deux sessions.

De plus, l'utilisation de tous ces outils n'est pas pensée pour forcément être séquentielle. En effet, il est possible à chaque moment de retourner en arrière en modifiant le code dans le Builder puis de retourner explorer ou visualiser le modèle, comme il est possible de pouvoir visualiser une simulation utilisant un jeu de paramètres obtenus grâce aux outils d'exploration et d'optimisation employés. Ce système est d'autant plus facilité par la possibilité de sauvegarder et de naviguer entre différentes versions du modèle selon les besoins.

En somme, ISiCell est une plateforme pensée pour employer et faciliter l'application de notre méthodologie. Chaque session est l'occasion d'utiliser ou de réutiliser les outils à disposition afin d'enrichir et de complexifier/simplifier le modèle de manière cyclique et incrémentale en fonction des besoins et des nouvelles hypothèses que l'on souhaite confirmer ou infirmer.

Dans le prochain chapitre, nous aborderons l'implémentation des différents outils qui constituent la plateforme. Le chapitre 6 présentera quant à lui des exemples de modèles développés avec la plateforme.

Chapitre 5

Architecture logicielle

Sommaire

Dans ce chapitre, nous allons détailler l'architecture des simulations développées ainsi que le fonctionnement de la communication client/serveur entre les différentes entités qui constituent la plateforme. L'ensemble de ces interactions sont résumées dans la Figure 5.1.

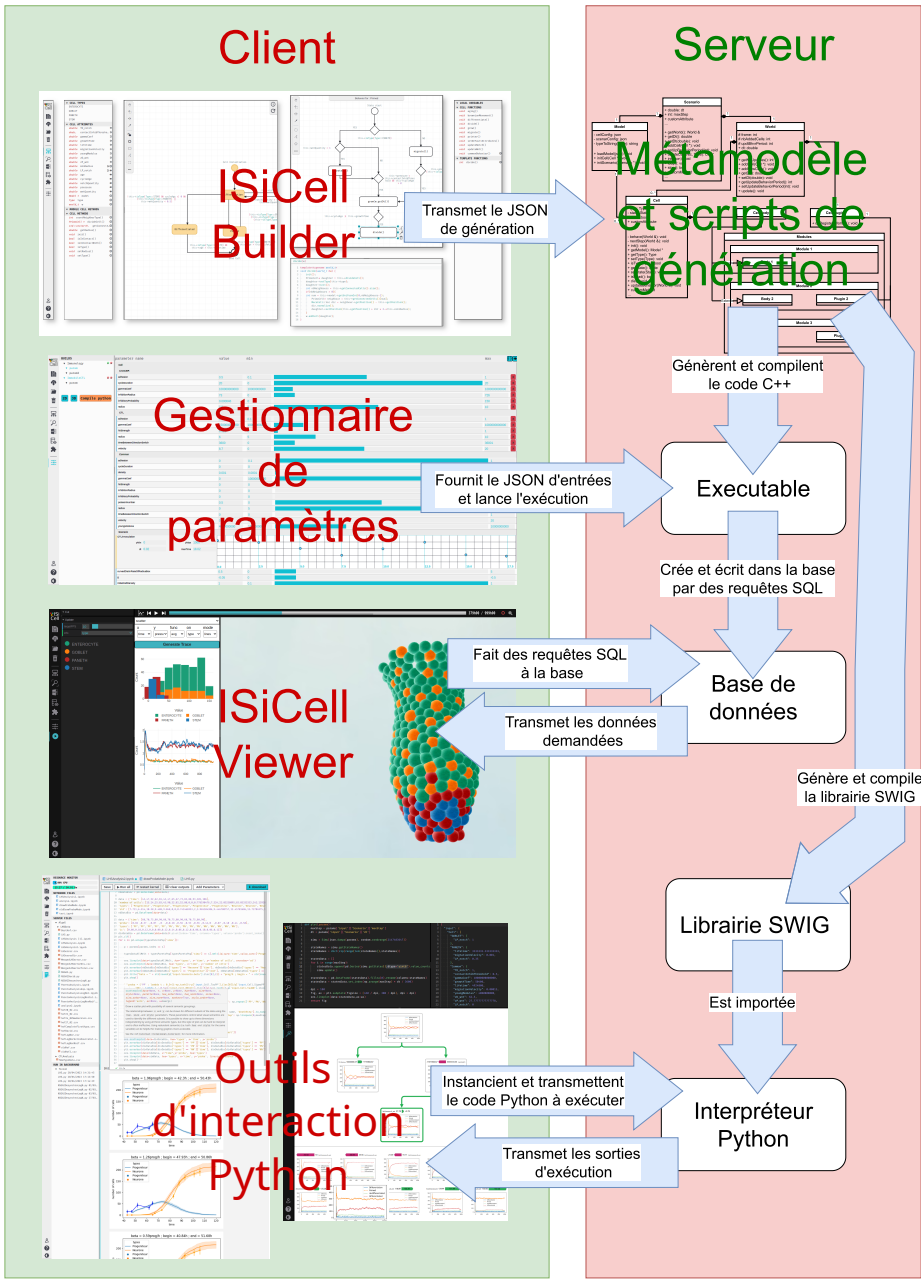


FIGURE 5.1 – **Vue d'ensemble des interactions entre les différentes entités constitutives d'ISiCell côté client et côté serveur.**

ISiCell a une architecture client-serveur et les différentes entités qui composent la plateforme interagissent et communiquent avec les deux parties. Ainsi ISiCell Builder communique au serveur les informations nécessaires à la génération du code du modèle. Le serveur peut ainsi générer et compiler le code à la volée et transmettre le modèle au Viewer ou aux outils d'analyse Python. L'interface de gestion de paramètres peut communiquer au serveur les paramètres d'entrées que l'on veut visualiser au serveur. Ce dernier va exécuter le modèle avec ces paramètres et le Viewer pourra accéder aux informations à afficher grâce à une base de données dans laquelle l'exécutable écrit. Enfin, les outils d'interactions Python transmettent du code Python que le serveur va exécuter, puis il renverra au client le résultat des exécutions.

Ce chapitre va donc se décomposer comme suit :

— Dans un premier temps, nous allons décrire le fonctionnement client du Builder au niveau de la génération du codes des diagrammes d'activité et du fichier JSON permettant de générer le code du modèle côté serveur ;
— Nous présentons ensuite l'architecture de nos modèles et la manière dont nous générons et injectons le code dans notre architecture pour obtenir le modèle voulu à partir des données fournies par le client ;
— La section suivante explique le fonctionnement du Viewer du point de vue de l'exécution du modèle précédemment compilé avec les paramètres fournis par l'interface client et par le prisme des requêtes SQL permettant l'affichage des différents éléments graphiques ;
— Enfin, nous détaillons le fonctionnement de nos outils d'interactions Python en abordant le système d'encapsulation du C++ pour l'interfaçage et comment elle est utilisée dans les différents outils.

5.1 Fonctionnement client du Builder

Au niveau du client du Builder, l'ensemble de l'affichage et des données propres à chaque modèle sont gérés par un fichier JSON. On trouvera ainsi dans ce dernier, la liste des modules utilisés, le code de chaque état et de chaque fonction ainsi que les listes de tous les attributs. Dans cette section, nous allons détailler l'algorithme de génération du code des diagrammes d'activité puis la structure du JSON qui permet de décrire et générer le modèle.

5.1.1 Génération du code des diagrammes d'activité

De manière simplifiée, le code des diagrammes d'activité sont générés selon l'Algorithme 1. Il consiste en l'instanciation des variables locales suivies de l'appel à la méthode *commonBehavior* et de l'appel récursif à la fonction de génération à partir du diagramme d'activité. Cet fonction récursive est appelée avec l'élément initial du diagramme, une indentation nulle et le code précédemment écrit et renvoie le code complet de l'état. Le code est écrit au fur et à mesure des appels récursifs selon que l'élément en paramètre est une activité ou une condition. On rappelle ensuite la fonction sur les éléments cible (cor-

respondant aux flèches du diagramme). L'indentation est incrémentée à chaque condition. Cela permet d'instancier les différentes variables et d'appeler les différentes méthodes dans l'ordre indiqué par le diagramme.

Algorithm 1 Génération du code des diagrammes d'activité

```
function recGen(element, indent, code)
    i ← 0
    while i ≤ indent do
        code ← code + HT        ▷ HT est le caractère ASCII pour la tabulation
        i ← i + 1
    end while
    if element.type == "activity" then
        code ← code + element.name + "(" + element.params + ");" + CR
        if size(element.targets) == 1 then
            code ← recGen(target[0], indent, code)
        end if
    else if element.type == "condition" then
        if size(element.targets) ≥ 1 then
            code ← code + "if(" + element.name + "){" + CR
            code ← recGen(element.targets[0], indent + 1, code)
            code ← code + "}"
            if size(element.targets) == 2 then
                code ← code + "else{" + CR
                code ← recGen(element.targets[1], indent + 1)
                code ← code + "}"
            end if
            code ← CR
        end if
    end if
    return(code)
end function
code ← ""
for v in localVariables do        ▷ pour toutes les variables créées localement
    code ← code + HT + HT + v.type + SP + v.name + ";" + CR
    ▷ SP et CR sont les caractères ASCII pour l'espace et le retour à la ligne
end for
code ← code + HT + "commonBehavior(" + commonBehaviorParams +
");" + CR
code ← recGen(activityStart, 0, code)
```

Les éléments qui ne sont pas reliés à celui de départ ne sont pas pris en compte dans la génération de code et la gestion des erreurs, cela permet d'assurer l'unicité du point de départ du diagramme. La Figure 4.3 du chapitre précédent présente un exemple de code généré à partir d'un diagramme d'activité. Le code de l'état *Differentiation* de notre modèle fil rouge de crypte intestinale donne le

```
commonBehavior(w);
if(this->notchQuantity < this->TD_notch){
        if(this->wntQuantity>=1){
                differentiate(Type::PANETH);
        } else {
                differentiate(Type::GOBLET);
        }
} else {
        if(this->wntQuantity < 1){
                differentiate(Type::ENTEROCYTE);
        } else {
                differentiate(Type::STEM);
        }
}
```

FIGURE 5.2 – **Code du diagramme d'activité de l'état *Differentiation* du modèle de crypte intestinale.**

code de la Figure 5.2

La méthode *commonBehavior* est appelé dans tous les états du diagramme, cela permet de centraliser des comportements qui sont toujours nécessaires, comme la mise à jour de certaines variables internes. La suite du code correspond au reste du diagramme à partir du point de départ. Dans notre exemple, on retrouve les quatre appels de la méthode *differentiate* dans les quatre contextes différents décrits par les trois conditions du diagramme. Cela permet aux cellules entrant dans l'état *Differentiation* de se différencier en un des quatre types du modèles en fonction des quantités de *notch* et de *wnt*.

Le code est généré pour chaque état et stocké dans le fichier JSON que nous allons ensuite décrire.

5.1.2 JSON de génération

Le fichier JSON que nous allons à présent détailler correspond à l'ensemble des métadonnées permettant de générer le code du modèle en C++. La structure des fichiers JSON nous permet de facilement manipuler ces données côté client (nativement en javascript) et serveur (grâce à des librairies Python et C++). Ce fichier contient les informations suivantes pour chaque version conservée du modèle :

- la liste des modules pré-implémentés utilisés,
- l'ensemble des modules créés pour le modèle avec le code des différentes classes pour chaque module,
- l'ensemble des énumérations créées sous forme de liste,
- la liste des types cellulaires,
- la liste des attributs du protocole avec leur nom et leur type C++,
- le code de l'initialisation de la simulation et de la boucle d'exécution de la simulation,
- l'ensemble des codes des méthodes du protocole indexée par un *hash*,

— la liste des attributs des cellules avec leur nom et leur type C++,
— le code de l'état initial,
— la liste des transitions depuis l'état initial avec la condition et le nom de l'état d'arrivée,
— l'ensemble des états avec leur nom et le code associé,
— la liste des transition pour chaque état avec la condition et le nom de l'état d'arrivée,
— l'ensemble des codes des méthodes du protocole indexée par un *hash*,
— les listes des paramètres d'entrée avec leur type C++ et leur nom,
— les listes des paramètres à enregistrer dans la base de données pour pouvoir être visualiser avec leur nom et leur type C++.

Ce fichier généré par la partie client de la plateforme est ensuite fourni au serveur pour générer le code du modèle. La structure du fichier JSON de génération est disponible dans l'annexe A.

5.2 Métamodèle et génération de code

Dans cette section, nous allons détailler l'architecture du métamodèle ainsi que la génération du code à partir des diagrammes dessinés dans l'interface d'ISiCell Builder. Ce métamodèle décrit la structure de tous nos modèles et permet de construire tous les modèles à partir de lui et des métadonnées décrites précédemment. L'architecture de nos simulations précède l'existence d'ISiCell, sa structure est le résultat d'une volonté d'uniformisation et de généralisation des architectures développées lors de précédents modèles. L'architecture globale est décrite dans le diagramme de classe de la Figure 5.3.

5.2.1 Introduction à la métaprogrammation

La métaprogrammation est définie par Cordy et al. [Cordy and Shukla, 1992] comme le processus de spécification de modèles génériques de sources logicielles à partir desquels des classes de composants logiciels, ou des parties de ceux-ci, peuvent être automatiquement instanciés pour produire de nouveaux composants logiciels. En effet, la métaprogrammation regroupe l'ensemble des techniques et pratiques permettant à un programme d'en générer un autre ou au moins une partie. De cette manière, on peut prendre en considération les usages allant de l'utilisation de la générécité grâce aux *templates* en C++ [Abrahams and Gurtovoy, 2004] qui permet au compilateur de générer le code nécessaire en amont de la compilation jusqu'au programme auto-modifiant capable d'inspecter et de modifier son propre code à la volée [Anckaert et al., 2007], en passant par les annotations Java permettant de générer du code à la compilation [Yu et al., 2019].

Lilis et al. propose de définir les langages de métaprogrammation selon 4 axes [Lilis and Savidis, 2019] :

— le modèle de métaprogrammation, c'est-à-dire les techniques sur lesquels le langage se repose ;

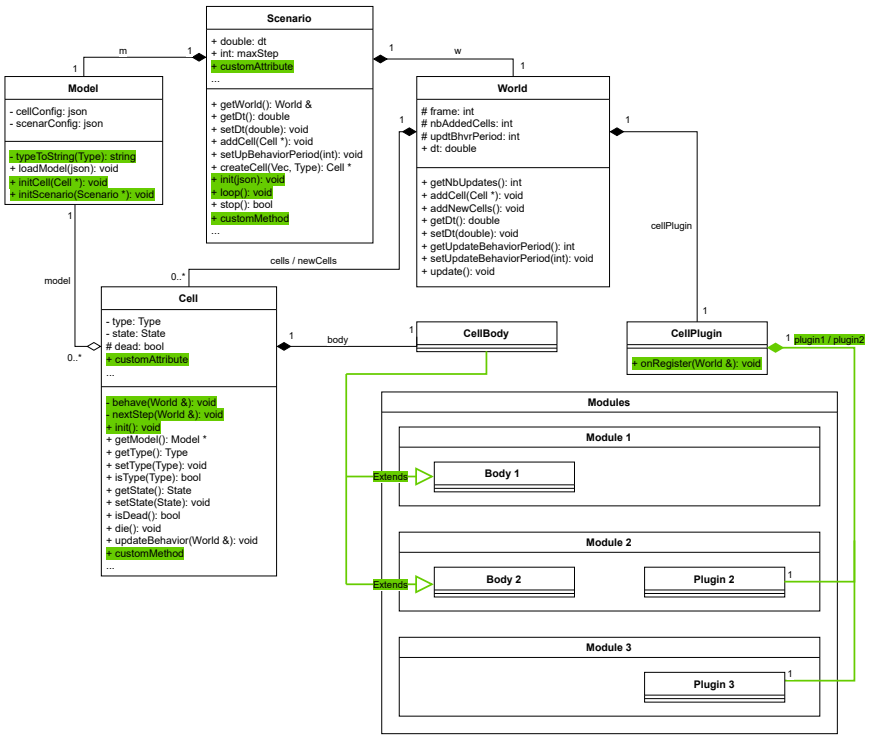


FIGURE 5.3 – **Diagramme de classe de l'architecture des modèles générés par la plateforme.**
Les éléments surlignés en vert correspondent à des méthodes, des attributs, des héritages ou des agrégations partiellement ou totalement générés à partir des métadonnées contenues dans le fichier JSON fourni par le client. Les classes *Scenario* et *Cell* peuvent être enrichies des différents attributs et méthodes ajoutés dans l'interface du Builder. Chaque module peut être constitué d'une classe *Body* et/ou d'une classe *Plugin*. La classe *CellBody* héritera de toutes les classes *Body* tandis que la classe *Plugin* sera composée des différentes classes *Plugin* utilisées. La classe *Model* permet d'initialiser tous les paramètres d'entrée du protocole et des cellules.

— la phase d'évaluation du métaprogramme, c'est-à-dire le moment où la métaprogrammation a lieu ;
— la source du métaprogramme, c'est-à-dire là où on trouve le code source du métaprogramme ;
— la relation du métalangage avec le langage du programme générée.

Dans notre cas, la métaprogrammation se base sur de la génération de code à partir d'un métamodèle constitué d'une arborescence de classes contenant

des balises et les scripts Python à évaluer permettant de générer le code C++ nécessaire à partir de métadonnées au format JSON avant la compilation. Une première phase de génération est effectuée en amont côté client à partir des diagrammes d'activités comme mentionné dans les sections précédentes.

Dans les sections suivantes, nous allons nous attarder sur la structure de nos modèles, sur la constitution du métamodèle à partir de cette structure et sur la génération du code effectuée côté serveur.

5.2.2 Agents cellulaires et comportements

Les modèles que nous développons s'inscrivant dans le paradigme des Systèmes Multi-Agents (SMA), il paraît donc pertinent de commencer par détailler la classe décrivant nos agents cellulaires.

Classe Cell

En termes d'attributs, nos cellules doivent a minima implémenter l'ensemble des caractéristiques que l'on veut observer dans le cadre du modèle. Ces attributs doivent correspondre à des caractéristiques biologiques expérimentalement mesurables ou observables. Cette classe pourra donc implémenter, au besoin, des caractéristiques correspondant à son fonctionnement interne (durée de cycle, quantité d'ADN, nombre de divisions, etc.) ou à son environnement local (quantités de molécules, nombre de voisines, quantités de nutriments, etc.). Ces attributs peuvent aussi bien être créés spécifiquement pour les besoins précis d'un modèle ou bien être issus d'un *Body* d'un module. En effet, les différents attributs de la classe *CellBody* sont accessibles par la classe *Cell* ce qui permet d'ajouter facilement de nouvelles caractéristiques déjà implémentées à nos agents cellulaires. Toutes les caractéristiques ajoutées ne provenant pas de modules sont en accessibilité *public* afin de faciliter l'autocomplétion niveau client et éviter d'avoir à générer l'encapsulation de tous les paramètres ajoutés. Enfin, peu importe le modèle développé, la classe *Cell* dispose toujours des attributs *type* et *state* qui sont des énumérations de chaînes de caractères correspondant aux différents types et états qui peuvent caractériser une cellule. Ces deux paramètres sont encapsulés et les valeurs qu'ils peuvent prendre dépendent du modèle.

De la même manière, les méthodes implémentées pour le modèle possèdent une accessibilité publique par défaut à la génération afin qu'elles puissent être appelées depuis l'instance de *World* (qui contient l'ensemble des instances de *Cell*) de la classe *Scenario* (qui correspond au protocole). L'idée étant de pouvoir accéder aux différents attributs et méthodes des cellules ajoutés depuis l'interface du Builder facilement sans avoir à gérer des cas spécifiques ou ajouter des *getters* et *setters* en fonction des attributs. De plus, le code généré pouvant être téléchargé, il est aussi possible de modifier ce code a posteriori et de le recompiler ensuite en fonction des besoins.

Comportement à base de machines à états

Le comportement des cellules s'assimile à celui d'une machine à états finis décrite par les diagrammes états-transitions correspondant au modèle. Les méthodes privées *behave* et *nextStep* sont appelées l'une après l'autre dans *updateBehavior* et permettent à nos agents d'adopter un fonctionnement de machine à états. Les deux fonctions sont constituées d'une structure à choix multiple (correspondant à un *switch case* en C++) couvrant chacun des états dans lesquels la cellule peut être. La première régit le comportement en fonction de l'état et la seconde les transitions possibles vers un autre état pour le pas de simulation suivant. Ainsi, à chaque appel de la fonction *updateBehavior*, une cellule exécutera le comportement associé à l'état dans lequel elle se trouve puis changera ou non d'état si les conditions sont réunies.

L'initialisation constitue un état particulier. En effet, à la création d'une cellule ou après une division, la méthode *init* est supposée être appelée afin d'initialiser les différents paramètres de la cellule et l'amener vers un des états directement relié à l'état initial. Dans cette fonction, la méthode *initCell* de l'instance de *Model* est appelée. En fonction du type cellulaire, les différents paramètres sont initialisés selon les valeurs sélectionnées dans l'interface de sélection de paramètres.

L'intérêt de l'utilisation de machines à états dans le cadre de la modélisation de phénomènes biologiques a déjà été mis en avant. En effet, cette structure permet de mettre en lumière la séquentialité de certains phénomènes notamment au niveau moléculaire au sein des cellules [Liu et al., 2022, Oishi and Klavins, 2014, Martínez-Pérez et al., 2007] mais aussi dans des cadres similaires au notre pour décrire le comportement de cellules dans un cadre multi-agents [Sütterlin et al., 2009]. Pour nos modèles, l'avantage est que nos différents diagrammes décrivant le comportement peuvent être facilement transformé en code.

5.2.3 Environnement et protocole expérimental

Le deuxième ensemble d'entités qui gère l'ensemble des nos agents et l'environnement dans lequel ils évoluent est présenté dans cette section.

Classe World

La classe *World* est la classe qui contient l'ensemble des cellules et gère les interactions entre cellules et entre les cellules et leur environnement gouvernés par les différents *Plugins*. C'est donc dans cette classe que l'actualisation des cellules et de l'environnement aura lieu à chaque pas de simulation, la méthode de mise à jour nommée *update* permet d'appeler les différents *hooks*[1] provenant des classes *Plugins* utilisées, d'incrémenter le nombre de pas représenté par l'attribut *frames*, ainsi que d'ajouter les cellules nées pendant ce pas de temps à

1. Un *hook* correspond à un point d'entrée d'un programme permettant de déclencher la réalisation d'actions supplémentaires à des moments précis.

la simulation. Les différents *hooks* et les autres fonctions de la méthode *update* sont appelés dans l'ordre suivant :

- *beginUpdate*, le *hook* regroupant l'ensemble des actions qui vont être exécutées en tout début de boucle ;
- *preBehaviorUpdate*, le *hook* regroupant l'ensemble des actions exécutées juste avant la mise à jour des comportements de chaque cellule ;
- *callUpdateBehavior*, cette fonction appelle la fonction *updateBehavior* de toutes les cellules, tous les *updtBhvPeriod* pas de simulation ;
- *addNewCells* ajoute l'ensemble des cellules créées précédemment à l'ensemble des cellules déjà présentes et appelle l'ensemble des *hooks onAddCells* des *Plugins* ;
- *preDeleteDeadCellsUpdate*, le *hook* regroupant l'ensemble des actions exécutées avant l'appel de la fonction de suppression des cellules mortes ;
- *deleteDeadCells* supprime toutes les cellules mortes ;
- *postBehaviorUpdate*, le *hook* regroupant l'ensemble des actions exécutées juste après la mise à jour des cellules ;
- *endUpdate*, le *hook* regroupant l'ensemble des actions qui vont être exécutées à la toute fin de la boucle.

Après avoir exécuté, l'ensemble des ces fonctions, on incrémente l'attribut *frames* afin de signifier l'avancée d'un pas de simulation.

Classe Scenario

La classe *Scenario* est celle qui implémente l'instance de *World* de la simulation. C'est dans cette classe que sont contenus tous les attributs et méthodes implémentées liés à la partie Protocole de l'interface du Builder. Depuis une instance de cette classe, il est possible de modifier les différents paramètres du *World* et aussi de créer, initialiser et ajouter une cellule à la simulation. En effet, c'est dans cette classe que l'on retrouve l'instance de *Model* qui permet d'intialiser les différents paramètres du *Scenario* et des cellules. La classe *Scenario* implémente deux méthodes principales :

- *init*, qui initialise l'instance de *Model* à partir du fichier JSON contenant les différents paramètres de la simulation, initialise ensuite les attributs du *Scenario* par l'appel de la méthode *initScenario* du même *Model* précédemment initialisé ainsi que le pas de temps de la simulation et exécute ensuite le code généré correspondant au diagramme d'activité de la partie init du Protocole ;
- *loop*, qui appelle la méthode *update* du *World* puis exécute le code généré à partir du diagramme d'activité construit dans l'interface loop du Protocole.

Le code d'une simulation correspond globalement à l'instanciation d'un *Scenario* à l'appel de sa fonction *init* puis en une boucle *while* conditionnée par la méthode *stop* (qui renvoie *true* lorsque le nombre de pas de simulation est supérieur à *maxStep*) dans laquelle on appelle la méthode *loop*.

5.2.4 Modularité

L'architecture de nos modèles permet de créer, d'ajouter et de retirer facilement différents modules. Ces modules peuvent être constitués d'une classe *Body* et/ou d'une classe *Plugin*. La première peut ajouter des caractéristiques et des briques comportementales aux cellules tandis que la deuxième permet d'ajouter des mécaniques au niveau de l'environnement et de la population. Ces modules sont pensées pour répondre à des besoins récurrents comme la gestion de la spatialisation, de la mécanique intercellulaire ou de la diffusion de molécule.

Classes Body

Comme énoncé précédemment, les classes *Body* sont pensées pour ajouter des caractéristiques et des fonctionnalités aux agents cellulaires. Cela peut être, par exemple, des paramètres permettant de connaître la quantité de molécules au sein de la cellule et des méthodes permettant de gérer l'absorption ou l'émission de ces mêmes molécules dans l'environnement ou alors des attributs mécaniques permettant le calcul des forces d'attraction répulsion entre deux cellules ainsi que des fonctions de migration pour permettre aux cellules de se déplacer dans un tissu. Ces méthodes et attributs seront accessibles via l'instance de *CellBody* que possède chaque cellule. Cette classe hérite de toutes les classes *Body* sélectionnées dans l'interface du Builder ce qui lui permet d'implémenter tous les attributs et méthodes spécifiques aux modules voulus.

Classes Plugin

De la même manière, les classes *Plugin* sont conçues pour permettre l'ajout de paramètres et de mécaniques au niveau de l'environnement et de la population. On pourra retrouver, par exemple, des attributs contenant le rayon d'un sphéroïde ou le nombre de cellules mortes et des méthodes permettant de faire se diffuser une molécule dans un espace bi- ou tri-dimensionnel, discret ou continu et de mettre à jour les concentrations dans chaque cellules ou alors des fonctions permettant de recalculer toutes les positions des cellules en fonction des forces qui s'appliquent. Les différentes méthodes et attributs pourront être accéder via l'instance de chaque *Plugin* contenu dans l'instance de *CellPlugin* depuis le *World* de la simulation. Ce dernier appellera à chaque pas de simulation toutes les fonctions de *hooks* contenues dans les *Plugins* sélectionnés. Pour être accessibles, les instances de *Plugin* de *CellPlugin* sont publiques. La fonction *onRegister* est un *hook*, appelé à l'initialisation du *World*, qui permet d'enregistrer tous les *hooks* des *Plugins* instanciés dans *CellPlugin*.

Modules développés

L'architecture modulaires de nos modèles précède l'existence de la plateforme [Bernard, 2020] et a permis le développement de nombreux modules répondant à des besoins spécifiques de certaines simulations. Avant l'élaboration d'ISiCell, les modules suivants avaient déjà été développés :

— *MassSpringDamper*, un module composé d'un *Body* et d'un *Plugin* permettant d'appliquer de la physique de masse-ressort entre cellules adjacentes dans un environnement tridimensionnel continu [Van Liedekerke et al., 2018] ;
— *CellCycle*, composé d'un *Body* et d'un *Plugin* facultatif, implémentant un système de cycle cellulaire à base de Processus de Bernoulli successifs issus de précédents travaux [Pascalie et al., 2010, Pascalie et al., 2011, Bernard et al., 2019] et permettant la mise en place de traitement impactant les phases du cycle ;
— *2DGrid*, un module composé d'un *Body* et d'un *Plugin* permettant de gérer la prolifération de cellules dans un environnement bidimensionnel discret développé dans le cadre de l'étude de Bernard et al. [Bernard et al., 2019] ;
— *Diffusion*, un module composé d'un *Body* et d'un *Plugin* permettant de gérer la diffusion de plusieurs molécules dans un environnement tridimensionnel continu, l'espace de diffusion est cependant discrétisé par l'emploi de voxels ;
— *EduMarker*, un module composé d'un *Body* et d'un *Plugin* permettant de reproduire les effets d'un marquage 5-Ethynyl-2'-deoxyuridine (Edu) et dépendant du module de cycle cellulaire ;
— *GrimesOxygen*, un module basé sur un article de Grimes et al [Grimes et al., 2014] composé d'un *Body* et d'un *Plugin* permettant de gérer la quantité d'oxygène des cellules dans un sphéroïde ;
— *HertzianPhysics*, un module développé dans le cadre de cette thèse et composé d'un *Body* et d'un *Plugin* permettant d'appliquer de la physique de contacts hertziens entre cellules adjacentes dans un environnement tridimensionnel continu [Van Liedekerke et al., 2018, Van Liedekerke et al., 2015].

Au cours du développement de la plateforme, les modules précédents ont pu être améliorés et raffinés selon les besoins et les spécifications de l'architecture. D'autres modules ont aussi vu le jour en parallèle du développement d'ISiCell :
— *3dEnvironment*, un module composé d'un *Body* et d'un *Plugin* permettant d'ajouter des contraintes physiques spatiales dans un environnement tridimensionnel continu ;
— *Diffusion2D*, un module composé d'un *Body* et d'un *Plugin* permettant de gérer de la diffusion de molécules en 2D selon le même principe que la fonction *diffuse* de NetLogo ;
— *SpheroidManager*, un module composé d'un *Body* et d'un *Plugin* permettant de gérer de calculer et afficher les informations liées à des simulations de sphéroïdes.

5.2.5 Décomposition du modèle

Notre architecture étant normalisée, il devient possible de générer le code d'une simulation en injectant le code spécifique à chaque modèle aux bons endroits. Pour se faire, il est nécessaire dans un premier temps de dégager un

métamodèle en distinguant les variants et invariants de notre structure. Puis, dans un second temps, de structurer les caractéristiques propres de chaque modèle afin de déterminer l'ensemble des informations nécessaires à la génération d'un modèle à partir du métamodèle. Chaque modèle constitue un dérivé de notre métamodèle et on utilisera aussi le terme de modèle dérivable pour le désigner.

Constitution du métamodèle

Dans notre cas, le métamodèle se constitue de l'ensemble de nos classes, d'un diagramme état-transition, des diagrammes d'activités associés et de listes de modules, de paramètres et de fonctions. En effet, chacun de nos modèles respectera la même architecture décrite par la Figure 5.3, la différence se fera sur les modules utilisés, le comportement des cellules et les mécaniques d'environnement implémentées. Ainsi à partir de cette architecture on peut définir dans un premier temps les invariants d'un modèle à un autre :

- pour la classe *Cell*, la gestion de la mort, l'ensemble des *getters* et *setters*, le constructeur, les attributs *type* et *state*, le pointeur de *Model* et l'instance de *CellBody* ainsi que la méthode *updateBehavior* ;
- l'ensemble de la classe *World* ;
- pour la classe *Scenario*, les attributs *dt* et *maxStep*, les instances de *Model* et de *World*, les *getters* et les *setters*, les méthodes de création et d'ajout de cellule ainsi que *stop* ;
- pour la classe *Model*, l'ensemble de ses attributs et la méthode *loadModel*, qui permet de charger les paramètres du fichier JSON d'entrée.

Tous ces éléments sont suffisament génériques pour fonctionner avec l'ensemble des modèles sur lesquels nous pouvons travailler.

De la même manière, on peut distinguer des éléments dans chaque classe que l'on retrouvera dans chaque modèle mais de manière spécifique pour répondre au besoin de chaque modèle :

- pour la classe *Cell*, les méthodes *behave* et *nextStep* qui vont dépendre du diagramme états-transitions et des diagrammes d'activités associés, toujours sur la forme d'un *switch case* sur les états, et la méthode *init* qui appellera toujours la méthode *initCell* du *Model* avant d'exécuter le code correspondant au diagramme d'activité de l'état initial ;
- pour la classe *Scenario*, le code de la fonction *init* exécutera le code du diagramme d'activité correspondant au *init* du Protocole après avoir initialisé le *Model*, le *Scenario* et le *dt* du *World*, et celui de la fonction *loop* appelle la méthode *update* du *World* avant d'exécuter le code du diagramme d'activité correspondant au *loop* du Protocole ;
- les valeurs des *enum State* et *Type* ;
- pour la classe *Model*, la méthode *typeToString*, permettant de convertir un type en chaîne de caractères, dépend de la liste des types existants, de même que les méthodes *initCell* et *initScenario* dépendent des attributs propres à chaque modèle ;

— pour la classe *CellBody*, sa définition dépend des *Body* dont elle hérite, et son constructeur également ;
— pour la classe *CellPlugin*, l'ajout des différents *Plugins* en tant qu'attributs et leur enregistrement dans le *hook onRegister*.

Ces éléments sont présents peu importe le modèle, leur code sera par contre différents selon les situations.

Enfin, il faut prendre en compte que pour le comportement et la gestion de l'environnement, de nouveaux attributs et méthodes peuvent être directement ajouté aux classes *Cell* et *Scenario* ainsi que de nouveaux modules répondant à des besoins n'étant pas encore couvert par ceux déjà pré-implémentés.

5.2.6 Génération et injection du code

Chacun de nos modèles reprennent l'architecture précédemment présentée qui est celle du métamodèle à partir duquel nous générons le code. Les métadonnées contenues dans le fichier JSON fourni par le client d'ISiCell Builder permettent de décrire chaque modèle et contiennent l'ensemble des informations nécessaires pour obtenir le code correspondant à partir du métamodèle.

Dans cette section, nous décrirons, dans un premier temps, le fonctionnement général des balises ISiGen disposées dans le code de notre métamodèle et qui permettent l'injection de code pour générer le modèle à partir de ses métadonnées. Puis nous détaillerons plus précisément chacune de ces injections, dans un second temps.

Fonctionnement des balises ISiGen

Le code est généré grâce à un système de scripts Python et de balises nommées ISiGen contenant elles-mêmes le code Python permettant de générer le code C++ nécessaire à leur emplacement. Le script principal prend en paramètres un jeton unique, une chaîne de caractère, permettant de distinguer si le code doit être encapsulé pour permettre l'interfaçage entre le C++ et le Python, et un fichier au format JSON fourni par le Builder et contenant toutes les informations nécessaires à la génération du code du modèle. Après avoir chargé l'ensemble des fichiers JSON contenant les informations essentielles, il génère ensuite l'arborescence du dossier qui contiendra le code généré et l'exécutable. Il copie, ensuite, l'ensemble des classes nécessaires ainsi que l'ensemble des modules utilisés. Il crée et ajoute les fichiers correspondant aux modules personnalisés. En parcourant l'ensemble des fichiers possédant des balises ISiGen, le script génère le code en exécutant le code des balises de chaque fichier un par un. Les balises *eval* permettent de simplement évaluer le code Python de la balise, cela permet, par exemple, de définir des fonctions à appeler pour générer le code des méthodes *behave* et *nextStep* dans la classe *Cell* à partir des bons fichiers JSON chargés précédemment. Tandis que les balises *insert* signifient que le code en-dessous renvoie une chaîne de caractères sur la sortie standard qu'il faut injecter à l'emplacement de la balise. Enfin, le script appelle

les commandes pour compiler le code qui vient d'être généré. Ces balises sont également nommées pour faciliter la gestion des erreurs.

La Figure 5.4 correspond à la balise ISiGen permettant de déclarer toutes les énumérations dans le fichier de la classe *Model* :

Les balises sont des chaînes de caractères contenues dans des commentaires C++. Le code contenu dans ces chaînes est celui qui va être exécuté. La première ligne de la balise permet de déterminer le type (*insert* ou *eval*) de la balise et nom de la balise (après le "@"). *BuilderData* correspond au fichier JSON provenant du client.

En fonction du type d'exécutable, avec écriture en base de données pour la visualisation ou encapsulé pour l'exploration, que l'on cherche à compiler, le script diffère un peu. En effet, dans le premier cas, un module spécifique est ajouté et complété grâce à des balises ISiGen afin de générer les requêtes SQL pour créer et remplir la base de données à chaque pas de simulation. Dans le second cas, un autre CMakeList est utilisé permettant la compilation en utilisant un module swig [Cottom, 2003, Beazley, 1998, Beazley et al., 1996] (extension en ".i") afin de permettre l'interfaçage entre le Python et le code C++. Dans le premier cas, l'exécutable est appelable directement en utilisant une chaîne de caractères représentant le JSON des paramètres d'entrée, dans le second cas, le fichier compilé correspond à une librairie accessible en Python. Cette librairie implémente une classe *Simu* qui prend un JSON de paramètres sous forme de chaine de caractères et une graine aléatoire en entrée et qui permet d'exécuter pas par pas une simulation avec les paramètres fournis. Cette classe dispose de méthodes permettant d'accéder aux différentes caractéristiques de la simulation en cours, que ce soit au niveau des cellules, du protocole ou de l'environnement. Ces différentes fonctions sont implémentées lors de la génération du code grâce aux balises ISiGen dont la classe Simu dispose.

Détails du code généré et injecté

Nous allons à présent détailler le code généré et injecté dans chaque classe où les balises ISiGen sont présentes.

Dans la **classe *Cell***, on trouve huit balises ISiGen :

- Une première balise *eval*, pour définir la fonction *buildConditionsCode* qui génère le code sous forme de chaîne de caractères des différentes transitions pour un état donné à partir du fichier JSON.
- La balise suivante est de type *insert* et permet d'ajouter les balises *include* nécessaire s'il y en a dans la liste correspondante dans le fichier JSON.
- Il y a ensuite une balise dans la méthode *behave* pour générer le code du comportement pour chaque état à partir des métadonnées.
- De la même manière, dans la méthode *nextStep*, il y a une balise *insert* qui utilise la fonction *buildConditionsCode* définie dans la première balise pour générer le codes transitions dans chaque état.
- Il y a ensuite trois balises *insert* pour insérer tous les attributs et les méthodes issus des métadonnées en publique dans la classe.

FIGURE 5.4 – **Code de la balise ISiGen permettant de déclarer les énumérations dans la classe *Model*.**
En haut, extrait de la classe dérivable du métamodèle contenant les balises ISiGen (en vert) permettant de déclarer toutes les énumérations. Au milieu, extrait du Json de génération du modèle d'inflammation musculaire (voir chapitre 6.1.2). En bas, extrait de la classe dérivée obtenu après exécution de la balise ISiGen correspondante. Le code généré est encadré en rouge.

— Enfin, il y a une balise dans la méthode *init* pour insérer le code lié à l'état initial et les transitions depuis cet état.

Dans la **classe *Scenario***, on trouve quatre balises ISiGen *insert* :

— Les deux premières permettent d'ajouter tous les attributs et méthodes contenues dans les métadonnées en accessibilité publique.

— Les deux suivantes permettent d'insérer le code lié au protocole expérimental dans les méthodes *init* et *loop*.

Dans la **classe *CellBody***, on retrouve quatre balises ISiGen *insert* :

— Les deux premières balises permettent d'insérer au début du fichier, les en-têtes *include* des différentes classes *Body* à hériter et de déclarer tous les *namespaces* des modules utilisés.

— La balise suivante permet de générer la définition de la classe en ajoutant l'héritage de toutes les classes *Body* des modules utilisés.

— Ensuite, on trouve une balise pour générer le code du constructeur qui appelle les constructeurs de toutes les classes *Body* héritées.

La **classe *CellPlugin***, quant à elle, contient cinq balises ISiGen :

— Une première balise *eval* pour déclarer la fonction *lowerFirstLetter* qui permet de passer en minuscule le premier caractère d'une chaîne de caractères afin de nommer les instances de chaque classe *Plugin*.

— Comme pour la classe *CellBody*, il y a deux balises *insert* qui permettent d'ajouter les en-têtes *include* et les *namespaces* pour les différentes classes *Plugin* des modules utilisés.

— La balise suivante permet de déclarer toutes les instances des classes *Plugin* avec un nom correspondant au nom de leur classe passé par la fonction *lowerFirstLetter*.

— La dernière balise permet de générer le code de la méthode *onRegister* en déclarant chacune des instances des classes *Plugin*.

Pour la **classe *Model***, le code est généré à partir des quatre balises *insert* ISiGen suivantes :

— La première balise permet de générer la déclaration de tous les *enums* comme les types et les états.

— La balise suivante permet de générer le code de la méthode privée *typeToString* qui permet de transformer l'attribut *enum Type* des cellules en chaîne de caractères. Cette méthode est utilisée par la méthode *initCell* pour accéder au paramètres correspondant dans le dictionnaire au type de la cellule à l'initialisation.

— Les deux balises suivantes permettent de générer le code des méthodes *initCell* et *initScenario* qui permettent d'initialiser les cellules et le protocole expérimental à partir du dictionnaire de paramètre de la simulation.

Si l'on veut visualiser la simulation, l'exécutable compilé a besoin que l'on ajoute le module SQL permettant d'écrire les données sélectionnées dans une base de données. Ce module n'est pas utilisé dans le cas de la compilation pour l'interfaçage Python. Ce module est notamment composé d'une **classe *SchemeSQL*** qui contient quatre balise ISiGen :

— Une première balise *eval* pour déclarer les fonctions *convert* et *cast_enum*. La première permet de convertir le type des attributs C++ en

types acceptés en SQL. La seconde permet de convertir les variables de type *enum* en entier.
— Les deux secondes balises se situent dans le constructeur. Elles permettent de générer les chaînes de caractères correspondant au requêtes SQL qui vont permettre de créer les deux tables *world* et *cells*. La première contiendra les données de pas, temps (en heure), nombre de cellules tandis que la seconde contiendra pour chaque cellule, à chaque pas de simulation, toutes les données sélectionnées.
— La dernière balise sert à générer le code des requêtes SQL pour ajouter dans la table *cells* les différentes valeurs de toutes les cellules pour un pas de simulation donné.

Dans le cas de la compilation à destination de l'interfaçage du code de la simulation C++ avec du Python pour l'utilisation dans les outils d'exploration, on génère une classe nommée *Simu* qui permettra de générer des instances de simulation et d'interagir avec en Python. La **classe *Simu*** contient six balises ISiGen *insert* :

— Les deux premières balises génèrent le code des méthodes *getTypeNames* et *getStateNames* qui permettent respectivement de renvoyer les vecteurs de chaînes de caractères des types et des états dans le même ordre que leurs *enums*.
— Les quatre balises suivantes permettent d'insérer les accesseurs pour tous les attributs des classes *Cell* et *Scenario*, les *enums* et leur conversion en chaînes de caractères.

5.2.7 Code résultant

A l'issue de l'exécution de toutes les balises ISiGen, on obtient le code nécessaire pour obtenir l'exécutable utilisée par le Viewer ou la librairie Python utilisé par les outils d'analyse Python. Dans les deux cas, le code reprend l'architecture du métamodèle précédemment présenté et correspond aux choix de modélisation faites dans l'interface client d'ISiCell Builder.

L'interface de choix de paramètres permet de fournir au serveur le fichier JSON de paramètres d'entrée pour exécuter une simulation côté serveur qui générera la base de données permettant au Viewer d'accéder aux données qu'il peut afficher. Pour les outils d'interaction Python, le fonctionnement est similaire. La partie client fourni le code à exécuter au serveur et ce dernier renvoie les résultats de l'exécution au client. L'environnement Python importe par défaut la librairie permettant d'exécuter le modèle.

Le fonctionnement et l'utilisation plus détaillés de l'exécutable et de la librairie seront abordés dans les sections suivantes dédiées au fonctionnement du Viewer et des outils d'interaction Python.

5.3 Fonctionnement du Viewer

Le Viewer, dans sa forme actuelle, dispose de deux modes de rendus (2D et 3D) en fonction des modèles. Hormis les différence dans le traitement des données à afficher, le fonctionnement reste globalement le même peu importe le mode. Dans tous les cas, l'affichage fait intervenir la base de données et la mémoire tampon.

5.3.1 Gestion des paramètres

L'interface qui permet de lancer la visualisation des simulations est la même que celle qui permet de naviguer entre les différentes versions du modèle mais aussi celle qui permet de modifier, copier et sauvegarder des jeux de paramètres. Quand on modifie le jeux de paramètres directement dans l'interface de modification des paramètres au format JSON, on retrouve un éditeur Monaco avec une coloration adaptée au format. Lorsqu'on lance la visualisation d'un modèle (préalablement compilé sinon les boutons n'apparaissent pas), une simulation est lancée avec pour paramètre la chaine de caractères au format JSON correspondant aux paramètres courants. C'est cette chaîne de caractère qui va permettre d'initialiser l'instance de la classe *Model* avec tous les paramètres d'initialisation pour les cellules et le Protocole.

Les paramètres d'entrées correspondent à ceux préalablement indiqués comme dans l'interface dédiée du Builder. Contrairement au paramètre du protocole, il est possible de spécifier pour les cellules des paramètres d'initialisation différents en fonction du type. En effet la méthode *initCell* de la classe *Model* permet d'assigner des valeurs différentes aux attributs des cellules en fonction du type cellulaire. S'il n'y a pas de valeur spécifiée pour un type donné, c'est la valeur contenue dans le type *Common* qui est prise. Cela impose que pour chaque paramètre, il y ait une valeur par défaut contenu dans *Common*. Pour chaque paramètres, on sauvegarde aussi les bornes minimale et maximale afin de pouvoir gérer l'affichage des glissières.

5.3.2 Lecture/Écriture en base de données

Au lancement d'une simulation, une base de données SQL est créée grâce au module dédié. Ce même module permet d'écrire à chaque pas de simulation, à l'appel du *hook endUpdate*, l'ensemble des attributs du protocole et des cellules qui ont été préalablement indiqués comme visualisables dans l'interface dédiée du Builder. Ce module implémente une librairie PostgreSQL [Makris et al., 2021, Martins et al., 2021] pour créer la base de données et effectuer les différentes requêtes pour la remplir. Cette base de données permet de stocker à chaque pas de simulation l'ensemble des caractéristiques demandées pour chacune des cellules de la simulation. Pour ce faire, le code de génération de la base contient des balises ISiGen qui vont permettre d'ajouter les colonnes nécessaires pour chacun des attributs que l'on a indiqué vouloir stocker.

L'entité qui gère la mémoire tampon côté client est aussi celle qui gère les requêtes à la base de données. Dans cette mémoire, vont être stockés l'ensemble des informations nécessaires à l'affichage de la simulation et des graphes à chaque pas de temps. Si des données manquent pour afficher un pas de temps alors le Viewer demandera dans sa prochaine requête au serveur les données nécessaires et les stockera dans la mémoire tampon. Cela permet de garder une visualisation fluide ce qui facilite l'interaction avec les biologistes.

Pour l'affichage, les données nécessaires à un pas de temps précis sont les coordonnées spatiales, le rayon (en 3D) et la valeur de l'attribut actuellement sélectionné pour la coloration pour chaque cellule. Pour les graphes, d'autres données peuvent être nécessaires selon ce que l'on veut tracer. Ainsi d'autres attributs peuvent être requis en plus de celui lié à la coloration. De plus, s'il est nécessaire d'afficher des données en fonction du temps, il faut alors agréger toutes les données de chaque cellules pour chaque pas de temps depuis le début de la simulation jusqu'au pas de temps le plus haut récupéré stocké en mémoire tampon. Dans notre cas, l'agrégation se fait en comptant le nombre de cellules qui ont la même valeur pour les attributs qu'on veut afficher temporellement. Si les valeurs en question sont continus, la somme sera faite. Les données sont toutes regroupées par états, types et pas de temps, le traitement des informations contenues dans la mémoire tampon sont ensuite traitées par les entités qui gèrent l'affichage des cellules et les graphes.

Si les requêtes prennent moins d'une seconde, la mise à jour de l'avancée de la simulation (pas de temps maximal atteint) se fait toutes les secondes. Cela permet de mettre à jour la barre d'avancement de la simulation. À chaque requête, lorsque la visualisation n'est pas en pause, la requête demande les informations nécessaires pour les pas contenus entre le dernier pas pour lequel on possède déjà les informations et le pas situé le double de la valeur *targetFPS* du Viewer plus loin. Cela a pour but de permettre une visualisation fluide selon le nombre de pas que l'on veut afficher à la seconde et limiter le volume de données demandé par requête. Si l'on demande une valeur plus grande que ce que la simulation a déjà atteint alors la requête retournera les données manquantes jusqu'au dernier pas de simulation enregistré et la visualisation pourra être saccadée car il n'y aura pas assez de pas chargés dans la mémoire tampon pour suivre la cadence d'affichage.

5.3.3 Visualisation 2D/3D

Que ce soit pour un affichage bi- ou tri-dimensionnel, les deux méthodes de visualisation des cellules utilisent des librairies basées sur du *Web Graphics Library* (WebGL [Leung and Salga, 2010, Yogya and Kosala, 2014]) qui permet d'exploiter les standards *Open Graphics Library* (OpenGL [Carr, 1997]) au sein d'une page web en étant compatible avec le JavaScript. L'intérêt du WebGL étant sa compatibilité avec la grande majorité des navigateurs actuels grâce à sa capacité à s'intégrer à du JavaScript et sa capacité à pouvoir générer des rendus aussi bien 2D que 3D. L'intérêt de notre Viewer par rapport à d'autres solutions comme ParaView [Kačeniauskas et al., 2010] basé sur la librairie Visualization

ToolKit (VTK [Schroeder et al., 1996]) est qu'il est directement intégré à notre plateforme et permet la visualisation en direct plutôt que d'installer un logiciel à part et de visualiser une simulation déjà terminée. Les deux modes d'affichage sont représentés dans la Figure 5.5.

Pour afficher nos cellules dans un espace tri-dimensionnel continu, notre Viewer nécessite que chaque cellule dispose de trois coordonnées spatiales, d'une couleur ainsi que d'un rayon. Elles seront ensuite représentées par des sphères de ce même rayon et disposées dans l'espace selon leur position. En fonction de l'attribut choisi pour la coloration, elles prendront la couleur correspondant à leur valeur de cet attribut. La visualisation en 3D se base sur la librairie Three.js [Danchilla, 2012] qui permet de gérer des rendus 3D en WebGL.

Dans le cas de la visualisation dans un espace bi-dimensionnel discret, notre Viewer emploie la librairie Pixi.js [2] qui permet de faire des rendus 2D en WebGL. De la même manière, pour gérer l'affichage de chaque cellule, il faut, pour chacune, que le Viewer reçoive leur deux coordonnées spatiales ainsi que la couleur qui correspondra à leur valeur de l'attribut sélectionnée pour la coloration. Avec cette visualisation, il est possible d'afficher plusieurs cellules à la même position, elles se diviseront la surface de la case sur laquelle elles sont.

Les données utilisées pour la visualisation sont contenues dans la mémoire tampon. Le rafraîchissement peut être déclenché manuellement en cliquant sur le moment de la simulation que l'on souhaite visualiser ou alors être déclenché automatiquement si la visualisation n'est pas en pause et selon le taux de rafraîchissement sélectionné dans l'interface. Peu importe la méthode, si les données ne sont pas déjà contenues dans la mémoire tampon, l'affichage aura lieu lorsque les données auront été chargées en mémoire après une requête à la base de données.

5.3.4 Gestion des graphiques

Enfin, la dernière entité principale qui participe au fonctionnement de l'outil de visualisation est celle qui permet la gestion et l'affichage des graphiques. Pour fonctionner, l'instance gérant les graphiques a besoin de récupérer dans la mémoire tampon les données nécessaires à la génération des graphiques. La visualisation d'un pas de simulation ne peut se rafraîchir tant que toutes les données nécessaires à la visualisation et à la génération des graphiques n'ont pas été chargées en mémoire tampon.

Pour générer les graphiques, notre Viewer se base sur l'utilisation de la librairie plotly [3] qui se repose sur stackGL et *Data Driven Documents* (D^3) [Bostock et al., 2011]. Plotly est compatible avec de nombreux langage dont le JavaScript, ce qui en fait un standard assez répandu pour la génération de graphiques interactifs et permet de facilement l'intégrer à une application web.

2. Site officiel de Pixi.js : https://pixijs.com/
3. Site officiel de plotly : https://plotly.com/

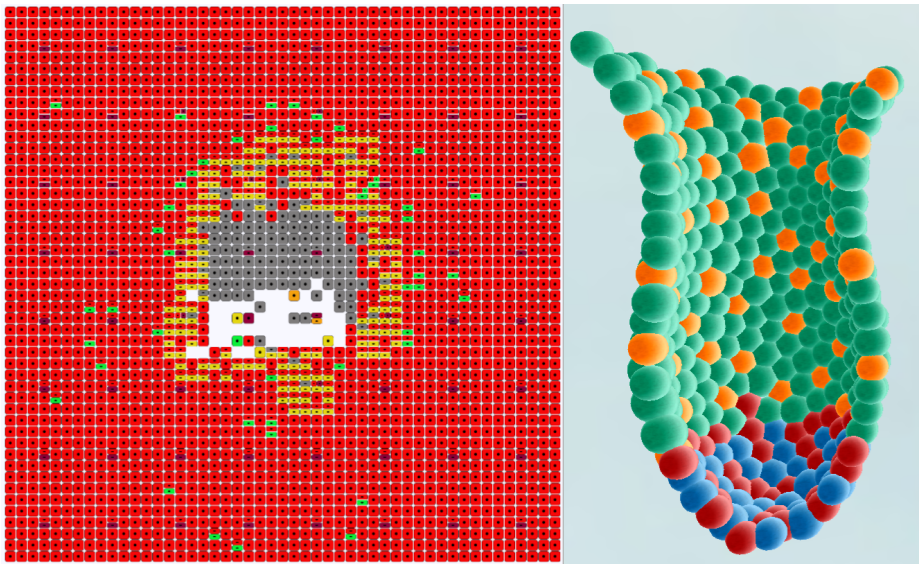


FIGURE 5.5 – **Aperçu de la visualisation 2D (à gauche) et 3D (à droite) du Viewer.**
La visualisation de gauche correspond à l'exécution d'une simulation d'inflammation de tissu musculaire, celle de droite à une simulation de crypte entérite. La visualisation en deux dimensions discrète permet d'afficher plusieurs cellules à la même position. En trois dimensions continues, il est possible de réaliser des coupes pour n'afficher que les cellules derrière un plan de coupe.

5.4 Fonctionnement des outils d'interactions Python

ISiCell Workbench comme ISiCell Explorer exploite tous les deux le serveur pour interpréter du code Python. Dans les deux cas aussi, pour gérer l'édition du code Python ou de paramètres au format JSON (dans le cas de l'Explorer), on retrouve des éditeurs Monaco avec une coloration et une auto-complétion adaptées au contexte spécifique. Dans les sections suivantes, nous allons expliquer plus en détail les interactions clients-serveur au niveau de ces outils d'exploration.

5.4.1 Encapsulation Python

Afin de pouvoir utiliser la simulation produite dans l'interface du Builder en Python, nous employons la librairie swig qui permet notamment d'interfacer du code C++ avec du Python. Cette librairie permet d'encapsuler un modèle

et de pouvoir instancier une simulation depuis un script Python en important le code spécifiquement compilé comme une librairie Python. L'encapsulation se fait au niveau d'une classe appelée *Simu* qui implémente une instance de *Scenario*. Cette classe dispose de méthode pour initialiser le *Scenario* et avancer pas par pas dans la simulation. Grâce à des balises ISiGen, la classe implémente des accesseurs en lecture afin de pouvoir accéder à chaque pas de simulations à toutes les informations voulues sur les cellules et l'environnement. Ces accesseurs renvoient notamment des listes des valeurs prises pour un attribut donné par l'ensemble des cellules instanciées. De la même manière, l'ensemble des méthodes contenues dans la classe *Scenario* sont accessibles directement depuis la classe *Simu*.

Pour permettre la compilation du code C++ en une librairie importable en Python, la librairie swig nécessite la présence d'un module swig sous la forme d'un fichier avec une extension ".i". Ce fichier renseigne les chemins vers les différents fichiers C++ à inclure et permet de définir des patrons de types de données particuliers à prendre en compte côté Python notamment ceux des cellules, des vecteurs de positions ou des vecteurs de valeurs. Ce fichier est suffisament générique pour pour fonctionner dans l'ensemble des cas que nous avons rencontré jusqu'à maintenant, c'est donc toujours le même module swig qui est utilisé à la compilation pour l'ensemble des modèles.

Que ce soit pour ISiCell Explorer ou pour ISiCell Workbench, les deux outils utilisent la même encapsulation du modèle pour faire fonctionner la simulation de manière compatible avec le langage Python.

5.4.2 Fonctionnement de l'Explorer

L'interface de l'Explorer utilisent deux instances d'éditeurs de texte. Le premier correspond à la fenêtre de gauche qui permet de modifier le code Python lançant la simulation et générant les graphiques. Le second correspond à l'éditeur de droite où l'on peut modifier le fichier JSON de paramètres par défaut de la simulation. Les deux éditeurs disposent d'outils de coloration adapté au langage et d'auto-complétion cohérent avec le contexte de leur utilisation. Cela rend l'utilisation de l'Explorer plus agréable et intuitive.

Côté script Python, il est impératif de définir une fonction nommée *plot* pour que l'exploration fonctionne. C'est cette fonction qui est appelée pour chaque jeu de paramètres pour lancer la simulation et générer le graphique que l'on veut afficher. Dans cette fonction, il faut donc gérer l'exécution de la simulation mais aussi la récupération des données que l'on veut afficher en les stockant afin de pouvoir générer le graphique à partir de celles-ci.

Lorsqu'une exploration est lancée, l'exécution est parallélisée avec autant de *threads* exécutant la fonction *plot* que de jeux de paramètres demandées, dans la limite du nombre de coeurs disponibles. Pour pouvoir afficher les figures générées, elle sont converties en base 64 puis affichée au format PNG lorsque l'exécution de la simulation est terminée. Les JSON de paramètres sont transmis sous forme de chaînes de caractères et sont générés à partir de celui situé dans la fenêtre de droite. La génération des valeurs des différents paramètres à explorer

se fait de manière linéaire entre les deux bornes s'il n'y a qu'un paramètre sélectionné et selon un hypercube latin s'il y en a deux ou plus. Ces jeux de paramètres sont générés en amont de la parallélisation des simulations.

Le code Python est exécuté côté serveur. Pour chaque exploration, c'est un interpréteur Python qui est créé et qui exécute le script générant les jeux de paramètres, exécutant les simulations et générant les graphiques. Chaque fois qu'un thread termine, il fournit côté client une chaîne de caractères au format JSON contenant l'index de la simulation pour la replacer au bon endroit, le graphique PNG en base 64 ainsi que les paramètres d'entrée qui diffère du jeu de paramètre initial afin d'afficher la différence par rapport aux paramètres initiaux. Lorsque le script a fini son exécution, le processus gérant l'exécution du script Python se termine. Ainsi à chaque exploration, un seul processus Python est exécuté et ce dernier est tué une fois qu'il a fini d'exécuter tous ses *threads*.

5.4.3 Fonctionnement du Workbench

ISiCell Workbench dispose d'un système de notebooks permettant d'analyser et d'explorer les modèles. Leur fonctionnement et leur structure s'approche du système de notebooks Jupyter [Randles et al., 2017, Pimentel et al., 2019, Perkel, 2018] au point qu'ils soient compatibles. La différence principale se situent dans le fait que chaque case des notebooks ISiCell est gérée par un éditeur Monaco et que l'interpréteur Python qui exécute les cases importent par défaut la simulation encapsulée.

Pour chaque notebook actif, un processus d'interpréteur Python personnalisé est lancé côté serveur jusqu'à ce que le notebook soit fermé, que l'utilisateur se déconnecte ou qu'il soit resté inactif trop longtemps. L'instance qui gère l'interpréteur importe par défaut certaines librairies et redéfinit certaines fonctions pour gérer l'affichage des sorties côté client (transmission des graphiques au format PNG en base 64, mise en forme de certaines structures de données, ...).

Entre le serveur qui exécute le code Python et le client qui affiche les sorties, la communication se fait au travers d'un flux. Le client transmet le code correspondant à la case que l'on souhaite exécuter et le serveur l'exécute dans le contexte du notebook correspondant. Pendant l'exécution, les sorties (sous forme de chaîne de caractères au format JSON) du serveur sont branchées sur l'entrée du client, ce dernier s'assure que la chaîne correspond bien à un JSON complet avant de générer l'affichage, sinon le client attend la suite de la chaîne. Le client attend des entrées jusqu'à ce que le serveur indique qu'il a fini d'exécuter le code de la case.

5.4.4 Autres fonctionnalités du Workbench

ISiCell Workbench dispose également de fonctionnalités permettant la gestion de fichiers. En effet, on peut sauvegarder des notebooks (extensions ".ipynb") directement dans un modèle. Ces notebooks seront uniquement liés au modèle en cours. Cependant, il est également possible de sauvegarder des dossiers et leur arborescence de fichiers côté serveur qui seront liés à l'utilisateur courant.

Parmi les fichiers qui sont gérés par ISiCell Workbench, on compte les fichiers CSV qui peuvent être nativement affichée de manière interactive (tris, filtres, etc.) et les scripts Python (extension ".py") qui peuvent être édité grâce à un éditeur Monaco spécifique au contexte. Ces derniers peuvent être exécuté en tâche de fond côté serveur et générer eux même des arborescence de fichiers côté serveur afin de pouvoir accéder aux résultats d'exécution. L'exécution de ces scripts est gérée par une section "*RUN IN BACKGROUND*" qui répertorie tous les scripts lancés en cours et terminés. Depuis cette liste, on peut afficher les sorties du script et l'arrêter au besoin, cela va envoyer une requête au serveur pour tuer le processus correspondant.

ISiCell Workbench donne accès à un terminal shell, ce qui permet aux utilisateurs un plus grand contrôle sur leur environnement de travail côté serveur. Ainsi, il est possible, par exemple, d'installer des librairies Python qui n'affecteront que l'environnement virtuel Python de l'utilisateur. De manière générale, cela permet d'utiliser l'ensemble des fonctionnalités du langage shell pour gérer son espace personnel de travail sur le serveur.

5.5 Conclusion préliminaire

Dans cette section, nous allons faire le bilan et résumer les différents aspects abordés dans ce chapitre et proposer des perspectives d'amélioration pour la plateforme et son architecture.

5.5.1 Bilan du chapitre

Tout au long de ce chapitre, nous avons abordé le fonctionnement de la plateforme depuis l'interface du *Builder* jusqu'à l'obtention d'exécutable C++ ou de librairie Python utilisés dans les différents outils de visualisation et d'analyse. Nous avons ainsi pu décrire les interactions client-serveur des différentes entités de la plateforme, l'architecture de nos modèles et la manière dont nous générons leur code.

La section 5.1 s'est attelée à présenter le fonctionnement client du *Builder* préalable à la génération du code des modèles. En effet, c'est au niveau du client que le code des diagrammes d'activité est d'abord généré et stocké dans le fichier JSON transmis au serveur pour permettre la génération du modèle entier.

La section 5.2 a, quant à elle, permis de décrire l'architecture de nos modèles en présentant notre modèle dérivable et comment le code était généré puis injecté pour obtenir nos modèles. Cette section a permis d'aborder la structure de nos balises ISiGen et comment celles-ci permettent d'obtenir le code de nos modèles.

Enfin, les sections 5.3 et 5.4 expliquent le fonctionnement des outils de visualisation et d'analyses Python du modèle. Ces derniers permettent la validation qualitative et l'analyse plus poussée des modèles. Ces sections ont présenté, d'un côté, la communication entre le *Viewer* et l'exécutable C++ au travers d'une base de données, et de l'autre côté, le fonctionnement de l'encapsulation Python

du modèle qui permet d'interagir avec notre modèle en Python au travers d'une librairie dans les différents outils d'analyse et d'exploration de la plateforme.

5.5.2 Perspectives futures

L'état actuel de la plateforme est satisfaisant vis-à-vis des attentes de la méthodologie et implémente même des outils d'exploration qui n'étaient initialement pas envisagés dans les premières esquisses de notre approche. Les pistes d'amélioration concernent principalement l'accessibilité de la plateforme pour des biologistes avec un minimum de bagage en informatique, la modularité du Viewer et le partage de modèles et de modules.

En ce qui concerne l'accessibilité, une piste, que nous souhaitons explorer, consiste en l'implémentation d'un métalangage pour remplacer le code C++ dans la définition du comportement. Cela permettrait d'utiliser une syntaxe simplifiée et plus accessible mais de toujours générer le code en C++ ensuite. Le frein principal au développement de cette fonctionnalité est la potentielle perte de flexibilité par rapport à l'usage du C++ si le métalangage ne permet pas d'exploiter toutes les possibilités du C++.

Pour la modularité du Viewer, cela permettrait d'offrir d'autres méthodes de visualisation que celles déjà existantes et de fournir aussi potentiellement de nouveaux outils d'analyse en direct de la simulation. Dans l'état actuel du Viewer, cela demanderait une refonte assez conséquente de sa structure et de son fonctionnement.

Enfin, concernant le partage et la diffusion de modèles et de modules, ISiCell Hub est une idée déjà évoquée pendant le développement de la plateforme. L'idée de ce nouvel outil consisterait à offrir une plateforme d'échanges à ISiCell. Les utilisateurs pourraient partager leurs modèles et modules sur le Hub et les autre utilisateurs pourraient charger les modules ou les modèles en fonction des besoins. Cela nécessite d'avoir déjà une communauté active outre le temps de développement que le Hub nécessite pour être fonctionnel.

Chapitre 6

Modèles développés avec la plateforme

Sommaire

Dans ce chapitre, nous détaillerons chacun des modèles qui ont été développés avec ISiCell et quels ont été leur apport à la plateforme. Le développement de chaque modèle a été une opportunité d'enrichir la plateforme en nouvelles fonctionnalités et d'éprouver sa pertinence et son efficacité vis-à-vis de notre méthodologie et de ses objectifs.

Parmi ces modèles, on compte :

— un modèle de crypte intestinale tridimensionnel de renouvellement du tissu du colon,
— un modèle d'inflammation musculaire bidimensionnel qui fait intervenir des mécanismes de diffusion,
— un modèle de traitement de cellules cancéreuses par l'injection programmée de Lymphocyte-T,
— un modèle de sphéroïde cancéreux faisant intervenir de la diffusion de molécules dans un volume,

— un modèle de neurogenèse chez l'embryon de poulet.

Les trois premiers modèles sont basés sur des modèles de la littérature. Ils ont permis avant tout de tester les capacités de la plateforme dans plusieurs contextes biologiques et de s'assurer qu'elle permet bien de reproduire plusieurs cas d'application. Les deux derniers sont des modèles qui ont été développés avec ISiCell en collaboration avec des biologistes pour évaluer la pertinence de notre méthodologie dans des conditions réelles.

Nous détaillerons, dans un premier temps, les modèles issus de la littérature, puis, dans un second temps, les modèles issus de collaborations avec des biologistes. Enfin, nous pousserons une réflexion plus large sur la plateforme et notre méthodologie et leurs apports concrets à la modélisation participative de modèles basés cellules.

6.1 Modèles issus de la littérature

Afin d'éprouver notre méthodologie et la plateforme, nous avons réimplémenté des modèles déjà existants dans la littérature dans le but de les reproduire en utilisant ISiCell. L'intérêt principal de ceci est de vérifier les capacités de la plateforme dans différents contextes biologiques et de fournir des modèles démonstratifs pour la plateforme en ligne. Tous ces modèles sont disponibles et constituent les modèles démonstratifs disponibles directement depuis l'url de la plateforme : https://isicell.irit.fr/.

Chaque modèle a été l'occasion d'enrichir la plateforme en nouvelle fonctionnalités, en briques comportementales et en nouveaux modules. Avec le développement de chaque modèle, leurs spécificités ont requis des améliorations et des enrichissements de la plateforme de manière incrémentale.

Chacune des sections suivantes présente un de ces modèles. Le modèle de crypte intestinale constitue un modèle intéressant pour illustrer l'intérêt de nos diagrammes pour faciliter la compréhension du fonctionnement du modèle et la reproductibilité. Le modèle d'inflammation musculaire a permis d'éprouver la visualisation en 2D discrète, de développer un module de diffusion, d'utiliser l'outil d'exploration et de tester l'usage des diagrammes sur un système plus complexe faisant intervenir un plus grand nombre de types cellulaires avec leurs propres états. Enfin le modèle d'injection de Lymphocytes-T a aidé à tester le système de protocole de l'interface des paramètres et de reproduire les constats de l'article dont il est issu.

6.1.1 Modèle de crypte intestinale

Le modèle de la crypte intestinale est utilisé depuis longtemps en modélisation informatique [Almet et al., 2020]. Il a suscité l'intérêt des modélisateurs depuis des décennies en raison de la complexité des mécanismes impliqués dans le renouvellement des cellules et leur différenciation spatio-temporelle. Ce modèle est issus de l'article de Buske et al. [Buske et al., 2011] et décrit le processus de

renouvellement cellulaire de l'épithélium intestinal à partir des cellules souches se situant à la base des cryptes intestinales.

Ce modèle est disponible à l'URL suivant : https://isicell.irit.fr/app.html?demo=Intestine_crypt.

6.1.1.1 Contexte biologique

Les cryptes du côlon, aussi appelées cryptes de Lieberkühn ou glandes intestinales, sont le lieu du renouvellement de l'épithélium des intestins [Clevers, 2013] et de la sécrétion d'enzymes participant aux processus d'absorption et de digestion. On distingue 4 types cellulaires qui participent au fonctionnement de ces glandes :

- les entérocytes, qui absorbent les lipides, les glucides et les autres produits issus de la dégradation des aliments,
- les cellules de Paneth, qui se situent au fond des cryptes et qui sécrètent des enzymes,
- les cellules caliciformes, qui sécrètent du mucus et facilitent la propagation des aliments dans l'intestin,
- les cellules progénitrices, qui se divisent lentement pour donner des cellules qui vont migrer vers le haut et se différencier en entérocytes ou en cellules caliciformes ou vers le bas et devenir des cellules de Paneth.

Dans notre cas, ce sont la structure de la crypte et les mécaniques de renouvellement et de migration qui nous intéressent. Dans la section suivante, nous allons détailler le fonctionnement du modèle développé à partir de l'article de Buske et al. [Buske et al., 2011].

6.1.1.2 Fonctionnement du modèle

Dans ce modèle, quatre types cellulaires (souches, Paneth, entérocytes et caliciformes) interagissent pour maintenir en permanence la crypte intestinale et suivent le diagramme états-transition de la Figure 6.1. Ce processus de renouvellement est piloté par deux conditions principales : le gradient de *Wingless and Integration site* (Wnt) [Pinto and Clevers, 2005] (plus élevé à la base de la crypte et plus faible au sommet) et les quantités de *notch* [VanDussen et al., 2012] dans la crypte. Ces deux molécules régulent la différenciation des cellules. Pour simplifier, la quantité de Wnt a une valeur de 1 sous une position z_P, une valeur de 0 au-dessus d'une position z_D et une valeur de 0,5 entre les deux. La position $z = 0$ correspond au bas de la crypte. De la même manière, au lieu d'un maillage 3D de la crypte auquel les cellules adhèrent dans l'article original, nous avons implémenté des forces perpendiculaires à l'axe de la crypte pour maintenir sa forme.

Les cellules souches, qui se trouvent principalement dans le fond de la crypte, peuvent se diviser lorsque la pression est inférieure à un seuil donné, reproduisant ainsi les mécanismes d'inhibition de contact. Pour se diviser, elles grossissent pendant une période donnée avant de finalement effectuer leur division. Ce comportement correspond à l'état indifférencié (*Undifferentiated* sur la Fi-

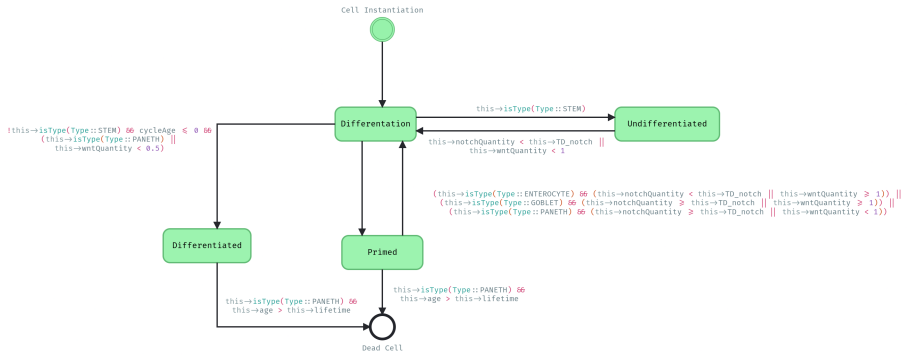


FIGURE 6.1 – **Diagramme états-transitions issus de l'interface du Builder pour le modèle de glande intestinale.**
Selon les quantités de molécules auxquelles la cellule est soumise, une cellule souche peut se différencier en trois types différents (entérocytes, Paneth ou caliciformes) ou rester indifférenciée si les conditions ne sont pas remplies. Une cellule amorcée peut encore se différencier si ses conditions changent. Après avoir rempli les conditions pour entrer dans l'état différencié, la cellule ne peut plus entrer dans l'état de différenciation. Les cellules amorcées et différenciées peuvent mourir si elles dépassent leur durée de vie sans se diviser ou sortir de la crypte.

gure 6.1) du diagramme états-transitions. Les cellules souches peuvent également entrer dans un processus de différenciation si leur quantité de *notch* n'est pas assez élevée ou si elles entrent dans une partie supérieure de la crypte où la quantité de Wnt est inférieur à 1. Si la quantité de *notch* n'est pas assez élevée, elles peuvent se différencier en type Paneth si elles se trouvent encore au fond de la crypte où la quantité de Wnt est plus élevée, ou devenir des cellules caliciformes si elles se trouvent au-dessus de la position z_D. Ces deux types sont dits sécrétoires car ils augmentent la quantité de notch dans les cellules avec lesquelles ils sont en contact. Si la quantité de *notch* est suffisamment élevée, les cellules souches peuvent se différencier en entérocytes lorsqu'elles quittent la partie inférieure de la crypte. Toutes ces différenciations correspondent au comportement lié à l'état différentiation *Differentiation* sur la Figure 6.1) du diagramme états-transitions.

Les cellules récemment différenciées peuvent entrer dans un état amorcé (*Primed* sur la Figure 6.1) où elles peuvent se diviser ou se différencier à nouveau si leurs quantités de *notch* ou de Wnt ne correspondent pas à leur type. Si les cellules caliciformes ou les entérocytes atteignent la partie supérieure de la crypte, où la quantité de Wnt est la plus faible, ils cessent de se diviser après une dernière division et entrent dans un état différencié (*Differentiated* sur la Figure 6.1) dans lequel ils ne pourront plus se répliquer ou se différencier. Les

cellules de Paneth ont tendance à entrer immédiatement dans l'état différencié si elles se trouvent au fond de la crypte.

6.1.1.3 Apport du modèle à la plateforme

Notre méthodologie associée à la plateforme ISiCell a permis de reproduire simplement le modèle de Buske et al. Nous avons pu démontrer la capacité de la plateforme à représenter un modèle plutôt bien décrit de la littérature et à reproduire une cinétique comparable à celle obtenue dans le travail des auteurs originaux (voir Figure 6.2). La principale contribution de notre plateforme à ce modèle réside dans la représentation graphique des règles internes du modèle et la visualisation graphique directe de la simulation générée à la volée, comme présenté dans la Figure 6.3. La Figure 6.1 montre que le comportement des cellules peut être facilement lu par un utilisateur n'étant pas familier avec la programmation. En outre, un tel diagramme peut être ajouté tel quel dans une publication pour améliorer la reproductibilité du travail : un diagramme d'état-transition peut être assez facilement réimplémenté pour obtenir un modèle équivalent.

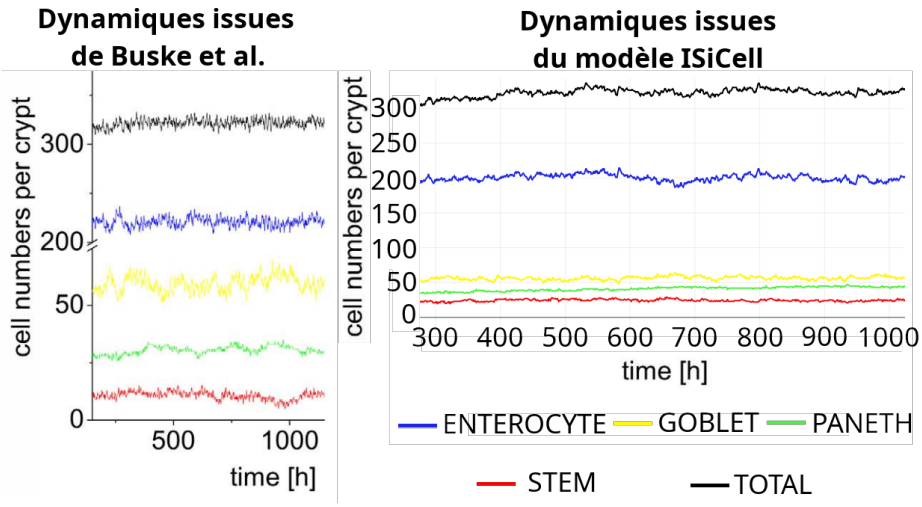


FIGURE 6.2 – **Comparaison des dynamiques de population.** A gauche, les dynamiques de population issus de l'article de Buske et al. [Buske et al., 2011]. A droite, les dynamiques de population générés avec ISiCell Viewer d'une simulation de crypte. Le modèle produit avec ISiCell est capable de reproduire des cinétiques similaires à celles de l'article originel.

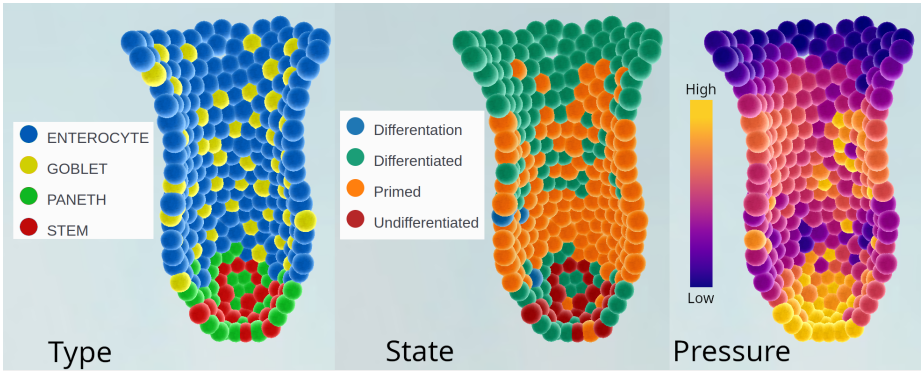


FIGURE 6.3 – **Visualisations des coupes de la crypte intestinale.** De gauche à droite, la distribution des types dans la crypte, suivie de la distribution des états et de la pression locale dans chaque cellule. La visualisation a permis de valider qualitativement le modèle en le comparant aux résultats attendus.

6.1.2 Modèle d'inflammation musculaire

La réponse immunitaire est un phénomène complexe impliquant différents types de cellules et de transmetteurs chimiques qui s'influencent dynamiquement les uns les autres. Cette dynamique complexe fait l'objet de plusieurs modèles informatiques [Vodovotz et al., 2004, Vodovotz et al., 2006, Li et al., 2008, Carbo et al., 2013] depuis de nombreuses années.

Le modèle présenté dans cette section est basée sur l'article de Gopalakrishnan et al. [Gopalakrishnan et al., 2013] qui décrit une grille torique discrète en deux dimensions de cellules musculaires (aussi désignées par le sigle IMCs pour *Individual Muscle Cells*). Dans ce modèle, une lésion centrale cause une infection bactérienne du tissu ce qui entraîne une réponse inflammatoire du système immunitaire. Le modèle original a été développé à l'aide de NetLogo [Wilensky and Rand, 2015].

Ce modèle est disponible à l'URL suivant : `https://isicell.irit.fr/app.html?demo=Muscular_inflammation`.

6.1.2.1 Contexte biologique

La réaction inflammatoire est une réponse du système immunitaire à une infection ou une lésion d'un tissu [Spector and Willoughby, 1963, Sherwood and Toliver-Kinsky, 2004]. Ce phénomène s'applique à l'ensemble du corps humain à l'exception des muqueuses et fait intervenir le systèmes immunitaires inné et adaptatif. Dans le cadre de ce modèle, nous nous intéresserons uniquement aux mécaniques liées au système immunitaire inné et à la réparation du tissu. L'inflammation est un processus en plusieurs temps qui fait intervenir plusieurs

types cellulaires. Parmi ceux-ci, on peut compter :

- les neutrophiles capables d'une phagocytose (capacité d'ingérer des particules étrangères) avant de mourir et de devenir du pus, qui appartiennent la famille des leucocytes (ou globule blanc) dont ils sont le représentant le plus nombreux dans l'organisme et qui font partie des premières cellules recrutées lors de l'inflammation ;
- les monocytes, des globules blancs qui deviennent des macrophages en passant dans les tissus ;
- les macrophages, capable de plusieurs phagocytoses dont celle du pus avant de mourir, responsables aussi de l'émission de messages chimiques pro-inflammatoires ou pro-résolutives en fonction de l'état et de l'avancement de l'inflammation.

Certains macrophages sont dits résidants car déjà présents dans le tissu, dans le cadre de ce modèle, nous nous intéressons uniquement au macrophage dit circulants issus de la différenciation d'un monocyte arrivé, comme les neutrophiles, depuis le sang dans le tissu.

Le processus de résolution de l'inflammation est enclenché, entre autres, par les macrophages présents sur le site de l'infection. Dans le cadre d'une lésion au niveau d'un tissu musculaire, la réparation du tissu endommagé est gérée par les myoblastes qui sont des cellules souches de fibres musculaires et peuvent donc se différencier pour remplacer les cellules mortes de lors de l'infection et de la réponse inflammatoire, en suivant le gradient pro-résolutif des macrophages.

6.1.2.2 Fonctionnement du modèle

Dans cette section, nous allons détailler par type cellulaire, le comportement qui lui est associé et qui correspond au schéma états-transitions de la Figure 6.4. À noter, que les molécules émises et consommées se diffusent dans une grille gérée par le module *Diffusion2D* mentionné au chapitre 5 spécifiquement développé pour les besoins de ce modèle.

Les IMCs sont initialement dans un état de repos correspondant à l'état *IMCRest* de la Figure 6.4. Elles passent à l'état activée (*IMCActivated* dans la Figure 6.4) si la quantité de bactéries, d'espèces oxygénées réactives (ROS pour *Reactive Oxygen Species* en anglais) ou de molécules DAMP (*Damage Associated Molecular Pattern* en anglais) est supérieure à leur seuil respectif et commencent à produire de l'interleukine 8 (IL-8). Elles peuvent revenir à leur état antérieur si les trois quantités mentionnées précédemment repassent en dessous de leur premier seuil, l'IMC cessera alors d'émettre de l'IL-8. Toutefois, si l'une de ces trois quantités dépasse son deuxième seuil, l'IMC devient endommagé (*Damaged* sur la Figure 6.4) et commence à être attaqué (ce qui se traduit par la perte de points de vie en fonction des quantités de ROS, de DAMP et de bactéries) jusqu'à ce que la cellule entre en apoptose (*IMCApoptosis* sur la Figure 6.4) (lorsqu'elle n'a plus de point de vie) ou qu'elle soit phagocytée par un macrophage. Lorsqu'elle entre dans l'état endommagé, une IMC émet une grande quantité de DAMP. Dans les états endommagé et d'apoptose, les IMC endommagées produisent une faible quantité de DAMP. Cette sécrétion de

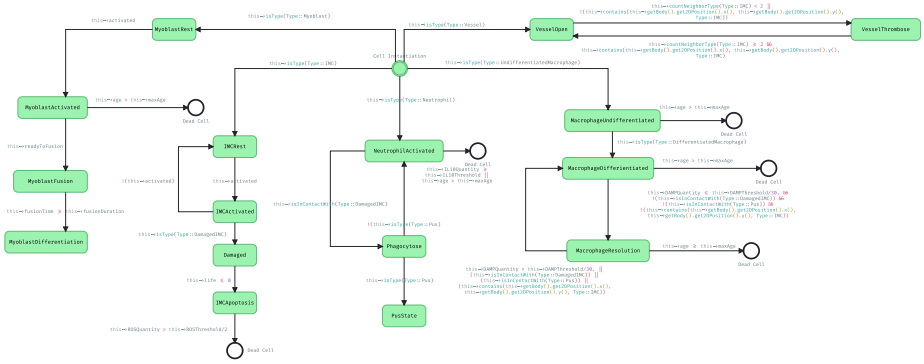


FIGURE 6.4 – **Diagramme d'états-transitions de la plaie musculaire.** Ce diagramme reflète la complexité de ce modèle, ce qui le rend plus facilement compréhensible et donc plus facilement reproductible.

DAMP n'était pas spécifiée dans l'article originel mais semblait nécessaire pour guider les macrophages dans l'élimination des cellules endommagées et mortes. Ce choix de modélisation a été prise après consultation de biologistes spécialisés dans l'inflammation. Une grande quantité de ROS peut également éliminer de la simulation une IMC endommagée en état d'apoptose.

Les neutrophiles entrent dans la simulation par les vaisseaux sanguins et sont activés par la quantité d'IL-8 à leur position. Dans leur état initial (*NeutrophilActivated* sur la Figure 6.4), ils suivent le gradient d'IL-8 et produisent la protéine chimioattractante monocytaire 1 (MCP-1), l'interféron gamma (IFN-γ) et des ROS jusqu'à ce qu'ils se retrouvent dans la même position qu'une IMC endommagée, qu'ils soient confrontés à une quantité insuffisante d'IL-8 ou qu'ils deviennent trop vieux (ce qui signifie qu'ils se déplacent dans la grille sans trouver d'IMC endommagée pendant trop longtemps). Dans les deux derniers cas, le neutrophile est retiré de la simulation. Dans le premier cas, le neutrophile change d'état (*Phagocytose* sur la Figure 6.4) et phagocyte les bactéries à son emplacement, à moins qu'un autre neutrophile ou un macrophage ne l'ait fait juste avant. Dans le second cas, le neutrophile retourne à son état précédent. Sinon, il élimine toutes les bactéries à sa place et émet une grande quantité de ROS avant de devenir du pus. Dans cet état (*PusState* sur la Figure 6.4), le pus émet une faible quantité de DAMP pour la même raison que les IMC endommagées.

Les macrophages indifférenciés (correspondant à des monocytes et à l'état *MacrophageUndifferentiated* sur la Figure 6.4) entrent dans la simulation par les vaisseaux sanguins de la même manière que les neutrophiles. Dans cet état, ils suivent le gradient de MCP-1 jusqu'à ce qu'ils deviennent des macrophages différenciés (*MacrophageDifferentiated* sur la Figure 6.4) ou qu'ils deviennent trop vieux (c'est-à-dire qu'ils se déplacent dans la grille sans se différencier depuis trop longtemps) et qu'ils soient retirés de la simulation. La différenciation

est déclenchée par une quantité suffisante d'IFN-γ ou de DAMP. Les macrophages différenciés commencent à émettre du facteur de nécrose tumorale alpha (TNF-α pour *Tumor Necrosis Factor alpha* en anglais), à suivre et à absorber du DAMP. S'il y a des bactéries, une IMC endommagée ou du pus à sa position, un macrophage les phagocyte en les retirant de la simulation. Si la quantité de DAMP est suffisamment faible et que le macrophage se trouve à proximité d'un IMC sain, sans IMC endommagée ni pus à proximité, il entre dans un état résolutif (*MacrophageResolution* sur la Figure 6.4) où il commence à produire des interleukines 6 et 10 (IL-6 et IL-10) et cesse d'émettre du TNF-α, de se déplacer et d'absorber du DAMP. Ils peuvent revenir à leur état antérieur si les conditions nécessaires pour entrer dans l'état résolutif ne sont plus respectées. Cet état n'était pas mentionné dans l'article originel mais semblait nécessaire pour reproduire la dynamique de réparation du tissu et a été implémenté après consultation auprès de biologistes spécialisés dans l'inflammation.

Les myoblastes entrent également dans la simulation par les vaisseaux sanguins dans l'état *MyoblastRest* sur la Figure 6.4. Ils commencent par suivre le gradient d'IL-6 et s'activent en présence de cette même molécule. Une fois activés (*MyoblastActivated* sur la Figure 6.4), les myoblastes sécrètent de l'IL-6 en présence d'une quantité suffisante d'IL-10. Ils continuent à suivre le gradient d'IL-6 jusqu'à ce qu'ils trouvent une position sans bactéries et avec une quantité de DAMP suffisamment faible, à côté d'une IMC saine ou d'un autre myoblaste. Dans ces conditions, ils commencent leur fusion (qui correspond au comportement de l'état *MyoblastFusion* sur la Figure 6.4) pour restaurer le tissu endommagé. Dans cet état, les myoblastes ne sécrètent plus aucune molécule. Après un certain temps, les myoblastes entrent dans un état de différenciation (*MyoblastDifferentiation* sur la Figure 6.4) et deviennent de nouvelles IMC saines.

Le dernier type d'agent dans ce modèle est le type vaisseau sanguin. Ils sont disposés de manière homogène dans la grille lors de l'initialisation. Ils n'ont que deux états : ouvert ou thrombose (*VesselOpen* et *VesselThrombose* sur la Figure 6.4). Les vaisseaux restent ouverts tant qu'il y a une IMC saine à leur position et au moins 2 autres IMC saines dans leur voisinage direct. Dans cet état, ils peuvent apporter des neutrophiles, des macrophages ou des myoblastes dans la simulation en fonction de la quantité de molécules dédiées. Les neutrophiles sont appelés par une quantité suffisante d'IL-8 ou de TNF-α, les macrophages indifférenciés par les quantités de MCP-1 ou de DAMP et les myoblastes par l'IL-6. Après avoir recruté au moins une de ces cellules, un vaisseau sanguin ne peut plus recruter de cellules pendant un certain temps. Ce mécanisme d'attente n'était pas mentionné dans l'article original mais semblait nécessaire pour réguler le nombre de cellules entrant dans la simulation.

Dans cet modèle, les bactéries ne sont pas traitées comme des agents mais comme des valeurs locales dans la grille avec leur propre gestion. Il existe deux types de bactéries, les régulières qui peuvent se développer sur les IMC endommagées et les virulentes qui se multiplient plus lentement mais peuvent se développer sur des IMC saines. Le premier type produit une bactérie virulente pour mille bactéries normales en se multipliant. Lorsque la capacité de charge

d'une position de la grille est atteinte, les bactéries commencent à se multiplier sur les positions directement voisines s'il y a une IMC endommagée ou saine (uniquement pour le type virulent). Leur nombre diminue également en fonction de la quantité de ROS ; les bactéries virulentes ont également une meilleure résistance aux ROS que les bactéries normales.

La Figure 6.5 inspirée de l'article originel permet d'illustrer le déroulement d'une simulation dans le cas d'un tissu dont la réparation a réussi et a permis de qualitativement s'assurer du bon fonctionnement du modèle.

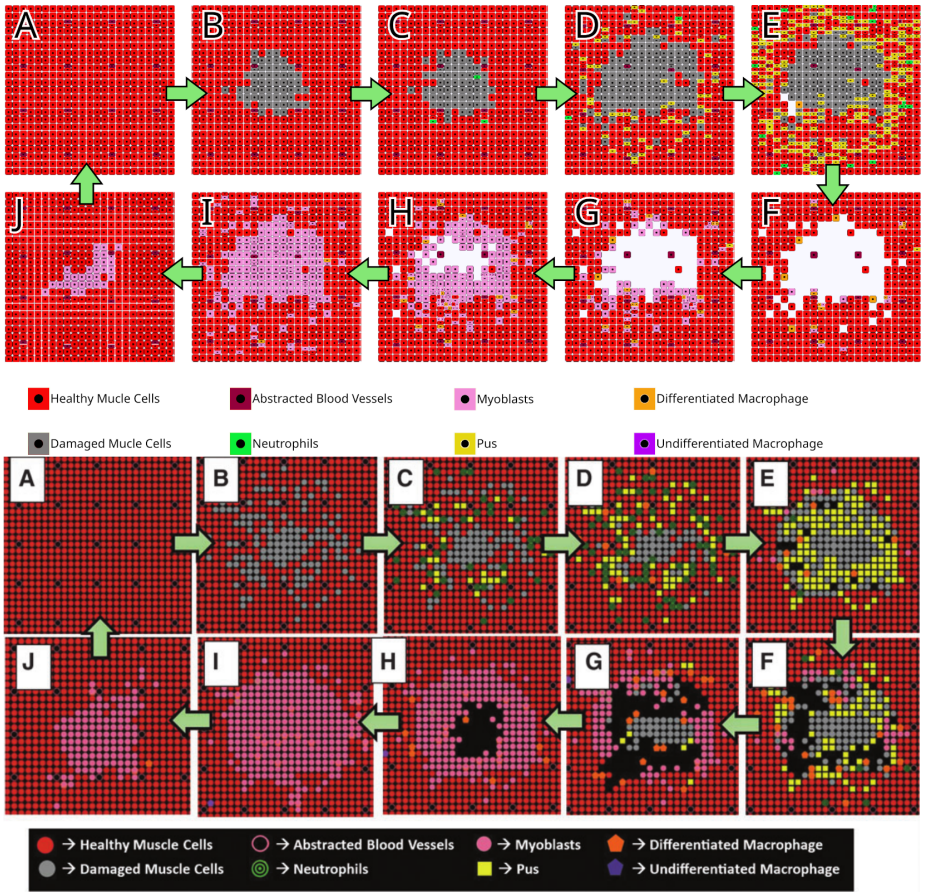


FIGURE 6.5 – **Série de captures d'écran d'une seule exécution représentative d'une guérison réussie (Haut) reproduisant des résultats similaires à ceux trouvés dans l'article originel (Bas).**

(A-J) montrent la progression du tissu, de sain à restauré après avoir été endommagé. ISiCell Viewer a permis d'obtenir cette reproduction qualitative. L'image du bas est issu de l'article de Gopalakrishnan et al. [Gopalakrishnan et al., 2013].

6.1.2.3 Apport du modèle à la plateforme

Ce modèle a été l'occasion de développer le module *Diffusion2D* qui permet de diffuser des molécules dans un environnement bi-dimensionnel discret. Ce module se base sur le fonctionnement de la fonction *diffuse* nativement présente dans *NetLogo* comme l'utilise le modèle issu de la littérature. L'ajout de nouveau module à la plateforme est l'occasion de l'enrichir en nouvelles fonctionnalités qui pourront servir dans d'autres contextes biologiques. Ainsi ce nouveau module de diffusion pourra être réemployé pour des modèles 2D discrets futurs.

Pour sommairement calibrer le modèle, nous avons utilisé ISiCell Explorer pour faire correspondre qualitativement les graphiques de dynamique de population de l'article originel (comme le montre la Figure 6.6). Et comme mentionné dans la section précédente, le Viewer nous a permis de nous assurer que le comportement des cellules étaient cohérents avec ce que l'on cherchait à reproduire.

Sur la base du modèle de blessure musculaire de Gopalakrishnan et al., nous avons obtenu un modèle d'inflammation similaire reproduisant une dynamique complexe comparable à celle fournie dans l'article original (voir Figure 6.7). La complexité de ce modèle se reflète dans son diagramme de transition d'état (voir Figure 6.4) résultant de l'application de notre méthodologie. Cependant, ce diagramme permet de mieux comprendre les différents comportements des types. En outre, les outils d'exploration et de visualisation rendent ce modèle-jouet intéressant pour étudier les interactions complexes et observer directement l'impact de différents ensembles de paramètres les uns sur les autres. Le grand nombre de paramètres de ce modèle rend l'utilisation de l'outil d'exploration pertinente.

Enfin l'utilisation des diagrammes d'activités et états-transitions facilitent, d'après nous, grandement la reproductibilité de ce modèle complexe. En effet, le diagramme d'états-transitions permet de visualiser le devenir des différents types cellulaires présents dans le système et les diagrammes d'activités de chaque état facilite la compréhension du comportement dans chacun des états dans lesquelles les cellules peuvent être.

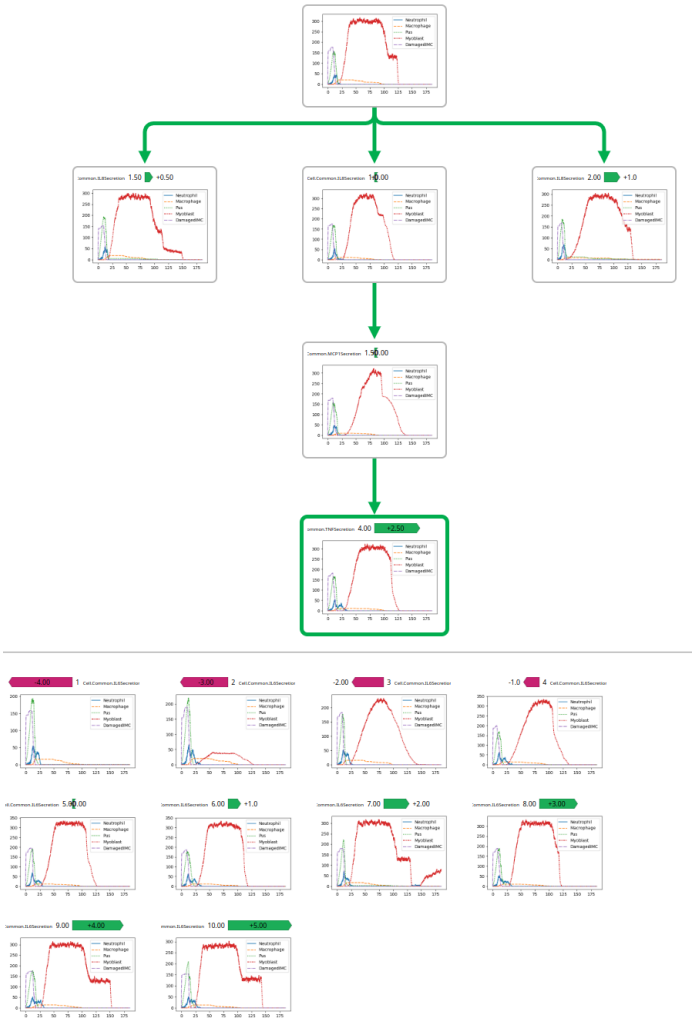


FIGURE 6.6 – **Exploration des paramètres du modèle de blessure musculaire à l'aide d'ISiCell Explorer.**
Cet outil s'est révélé utile pour étudier l'impact des sécrétions de chaque molécule sur la dynamique du type en nous aidant à s'approcher des dynamiques de l'article originel.

6.1.3 Modèle d'injection séquentielle de Lymphocytes T Cytotoxiques

Les modèles in silico peuvent être utilisés pour l'exploration de protocoles thérapeutiques [Benzekry et al., 2012, Swierniak et al., 2009, Pappalardo et al., 2010]. En effet, un modèle bien calibré constitue un outil prédictif qui peut

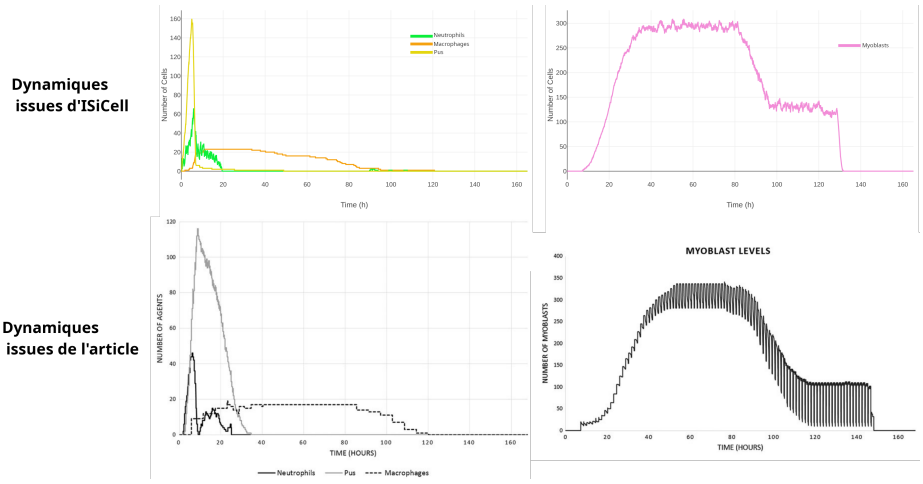


FIGURE 6.7 – **Comparaison des Dynamique des types cellulaires.** Le modèle développé avec la plateforme a permis de reproduire des dynamiques similaires à celles présentées dans l'article original.

aider à explorer les différentes possibilités de protocoles plus rapidement et plus efficacement. Ils peuvent ainsi fournir des protocoles optimisés répondant à des contraintes spécifiques.

Ce modèle est basé sur des travaux antérieurs sur le traitement immunitaire de culture de cellules cancéreuses [Khazen et al., 2019]. Il se concentre sur la dynamique de croissance en monocouche de cellules cancéreuses en présence de Lymphocytes T Cytotoxiques (LTC) qui sont ajoutés de manière séquentielle.

Ce modèle a permis de développer et d'employer l'outil de protocole pour injecter les LTCs au moment et avec la quantité voulue. Nous avons pu également tester nos modules de physiques continue 3D pour un contexte continu monocouche qui implique une disposition des cellules sur une surface même si nos cellules sont représentés par des sphères.

Ce modèle est disponible à l'URL suivant : `https://isicell.irit.fr/app.html?demo=Immunology`.

6.1.3.1 Contexte biologique

Le cancer est une maladie qui concerne l'ensemble du règne animal. Les cellules cancéreuses sont issues de mutations induites par des facteurs exogènes (rayonnement ultraviolet, tabac, alcool, etc.) ou endogènes (comme les espèces réactives à l'oxygène) qui peuvent altérer directement l'ADN ou favoriser des inflammations qui fragilisent les cellules. Ces dernières disposent de mécanismes pour réparer l'ADN altéré. Si les mutations empêchent la réparation de l'ADN, les cellules disposent également d'un mécanisme d'apoptose qui permet à la cel-

lule mutante d'initier sa mort d'elle-même. Enfin si aucun de ces mécanismes fonctionnent plus ou qu'ils sont inhibés, la cellule devient potentiellement immortelle et présente des risques de devenir cancéreuse. Ces cellules mutantes peuvent présenter des antigènes différents des cellules reconnues de l'organisme et ainsi être repérée par des cellules du système immunitaire adaptatif et enclencher la production de Lymphocytes T Cytotoxiques (LTC) capables de se fixer sur les cellules présentant l'antigène mutant repéré et de les détruire. Seulement, les cellules cancéreuses sont des cellules qui prolifèrent rapidement sans mécanisme de restauration de l'ADN, ce qui a pour effet d'induire de nouvelles mutations qui peuvent à terme leur permettre d'échapper plus efficacement à la réponse immunitaire du corps. Ces mécanismes peuvent, par exemple, entraîner une inhibition locale des LTCs et les rendre inopérants.

6.1.3.2 Fonctionnement du modèle

Les cellules cancéreuses se développent jusqu'à la fin de leur cycle avant de se diviser. Si elles entrent en contact avec au moins un LTC actif, elles entrent dans un état défensif. Dans cet état, elles ont une probabilité d'inhiber les LTCs dans un rayon prédéfini à chaque étape de la simulation. S'il n'y a plus de LTC en train d'attaquer, elles reprennent leur cycle où il en était, sinon elles ont été éliminés par des LTCs et sont retirées de la simulation. Les LTCs actifs se déplacent au hasard dans l'environnement de simulation jusqu'à ce qu'ils entrent en contact avec une cellule cancéreuse et passent à l'état offensif. Dans cet état, ils attaquent les cellules cancéreuses avec lesquelles ils sont en contact jusqu'à ce qu'ils soient inhibés ou qu'ils ne soient plus en contact avec aucune cellule cancéreuse. Après avoir tué les cellules cancéreuses avec qui ils étaient en contact, les LTCs actifs retournent à leur état de patrouille du tissu. S'ils sont inhibés par une cellule cancéreuse, les LTCs entrent dans un état d'inhibition où ils errent indéfiniment sans plus attaquer. Le diagramme d'états-transitions associé correspond à la figure 6.8. Le module de physique hertzienne [Van Liedekerke et al., 2015, Van Liedekerke et al., 2018], succinctement abordé au a été utilisé pour façonner ce modèle continu 2D et gérer les mouvements des cellules, il permet en effet de gérer indépendemment les forces d'adhésion et de répulsion, ce qui facilite les comportements de migration dans des milieux congestionnés. Un exemple de protocole du modèle est illustré dans la figure 6.9. Dans ce protocole, on injecte au total 1000 LTCs à trois intervalles de 4h (t0, t0+4h, t0+8h) et nous observons l'évolution de la population de cellules cancéreuses avec ce protocole. On peut également observer les dynamiques des LTCs qui sont inhibés au fur et à mesure qu'elles attaquent les cellules cancéreuses.

6.1.3.3 Apport du modèle à la plateforme

L'outil de conception de protocoles de la plateforme offre la possibilité de définir plusieurs injections de LTCs pendant la simulation afin de reproduire un traitement clinique sur la culture cellulaire. Le nouveau modèle construit a réussi à reproduire les dynamiques de population de l'étude originale, comme le

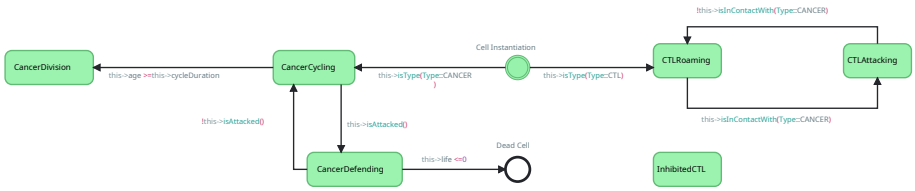


FIGURE 6.8 – **Diagramme états-transitions du modèle d'injection séquentielle de LTCs.**
Le comportement est partagé entre celui des LTCs d'un côté et celui des cellules cancéreuses de l'autre.

montre la figure 6.10. Ces expériences in silico suggèrent qu'étaler les injections de LTCs est plus efficace pour contrôler la croissance de la tumeur.

6.2 Modèles issus de collaborations

Dans ce chapitre, nous allons présenter deux modèles résultant de collaborations avec des biologistes. Ces modèles ont été développés grâce à la plateforme ISiCell et à notre méthodologie participative, ce qui nous a permis de les éprouver dans les conditions pour lesquelles elles ont été conçues.

Chaque modèle a été l'occasion de raffiner la plateforme en nouvelle fonctionnalités, en briques comportementales et en nouveaux modules. Avec le développement de chaque modèle, leurs spécificités ont requis des améliorations et des enrichissements de la plateforme de manière incrémentale.

Chacune des sections suivantes présente un de ces modèles. Le premier modèle présenté dans cette section est le modèle de sphéroïde cancéreux qui constitue le premier modèle développé en collaboration avec notre méthodologie et ISiCell alors qu'elle ne disposait que du Builder et du Viewer. Ce modèle nous a permis de travailler dans un contexte déjà connu et d'améliorer la plateforme au fur et à mesure de la construction du modèle et des besoins qui étaient soulevés. Le second modèle correspond à une collaboration sur un plus long terme et qui exploite l'ensemble des outils de la plateforme et a été réalisé à partir de zéro. Il nous a permis d'employer notre méthodologie dans un contexte nouveau et de mesurer l'efficacité de cette dernière et d'ISiCell dans les conditions pour lesquelles ils ont été conçus.

6.2.1 Modèle de sphéroïde

Les sphéroïdes cancéreux constituent un modèle *in vitro* largement utilisé en biologie comme présenté dans la section 2.2.2 pour étudier les premières étapes de formation des tumeurs. Les sphéroïdes ont également été le sujet de plusieurs modèles *in silico* [Amereh et al., 2021, Mao et al., 2018, Schaller and Meyer-

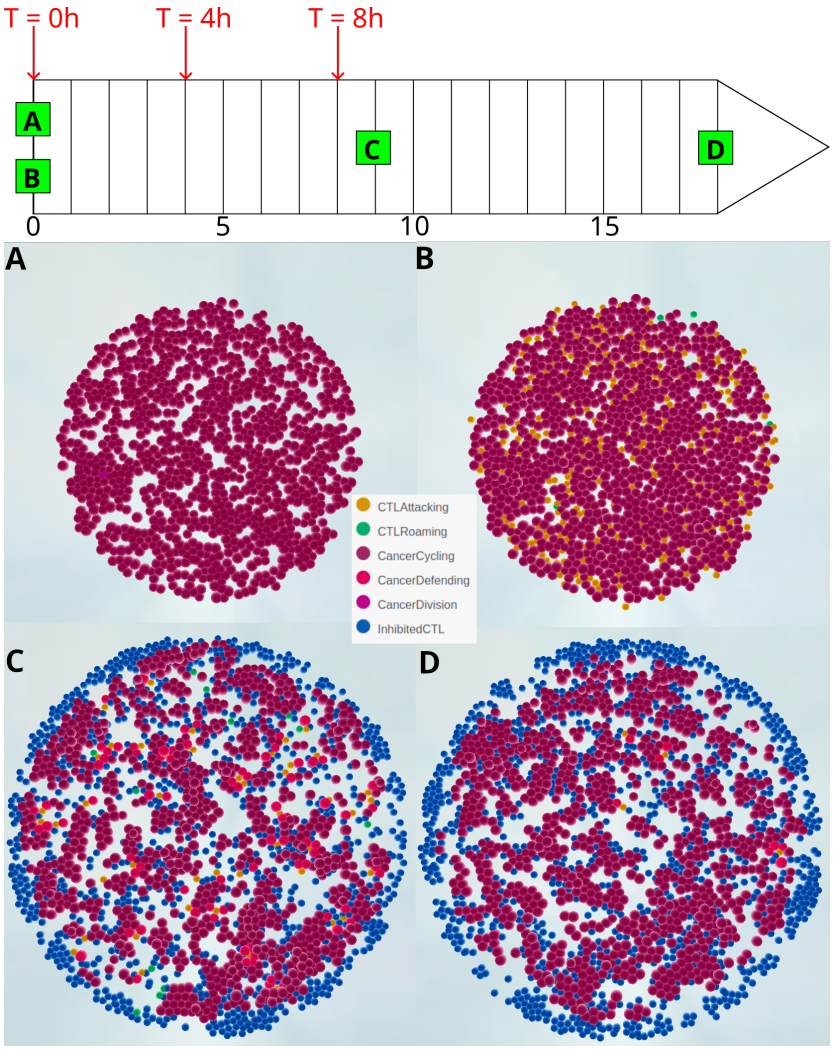


FIGURE 6.9 – **Série de captures d'écran d'une simulation de 18 heures avec 3 injections de LTCs et 1 000 cellules cancéreuses initiales.** Le protocole présenté ci-dessus propose 3 injections à T=0h, T=4h et T=8h pour un total de 1 000 LTCs. (A) montre la disposition initiale des cellules cancéreuses suivie de la première injection de LTCs (B). (C) montre la simulation après 9 heures et trois injections de LTCs et (D) représente l'état de la simulation à la fin.

Hermann, 2006] afin de reproduire les phases précoces du cancer à un plus haut niveau d'abstraction.

Ce modèle s'appuie sur des recherches collaboratives antérieures sur les cel-

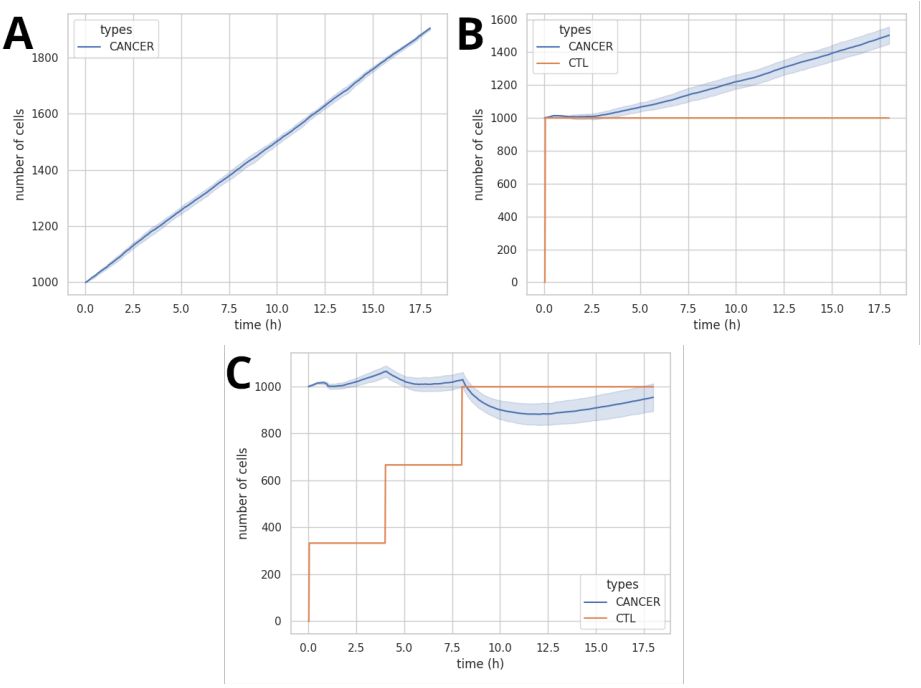


FIGURE 6.10 – **Graphiques des dynamiques de croissance de population générée en fonction du protocole d'injection des LTCs.**
(A) population de contrôle sans LTC.
(B) Protocole d'injection 1 :1 avec une injection à T=0h.
(C) Protocole d'injection 1 :1 avec 3 injections égales.
Les résultats montrent que la division des injections améliore le contrôle de la croissance tumorale.

lules cancéreuses, l'étude des réponses aux agents ciblant les microtubules, et leur impact sur la dynamique du cycle [Pascalie et al., 2010, Pascalie et al., 2011, Laurent et al., 2013, Bernard et al., 2019]. Ce modèle permet de simuler la croissance d'un sphéroïde avec plusieurs protocoles différents.

Ce modèle est issu d'une collaboration avec Pr. Valérie Lobjois à partir des travaux précédents réalisés par Dr. David Bernard, Pr. Bernard Ducommun et Pr. Valérie Lobjois. Il est disponible à l'URL suivant : `https://isicell.irit.fr/app.html?demo=Spheroid`.

6.2.1.1 Contexte biologique

Les sphéroïdes sont des agrégats cellulaires tridimensionnels qui permettent d'étudier des tissus et des populations de cellules au sein d'un volume ce qui permet de se rapprocher de conditions in vivo par rapport à des cultures mo-

nocouches de cellules. Ils sont notamment utilisés pour reproduire des microtumeurs afin d'étudier les effets de la spatialisation et du micro-environement sur les cellules cancéreuses au sein de la tumeur. Ainsi, on pourra observer l'impact du manque d'accès aux nutriments et à l'oxygène des cellules qui sont au centre par rapport à celles en périphérie. Ce phénomène tend à disparaître avec la vascularisation de la tumeur dans un être vivant.

Le cycle cellulaire correspond aux différentes étapes de la vie d'une cellule, de sa naissance jusqu'à sa division. Il se divise généralement en 4 étapes (voir Figure 6.11) [Baserga, 1981, Israels and Israels, 2000] et exclut les processus d'apoptose et de nécrose qui conduisent à la mort d'une cellule sans avoir atteint la mitose. On divise aussi le cycle en interphase (phases G1, S, G2) où la cellule prépare sa division et interagit avec son environnement et les cellules voisines selon ce pourquoi elle a été programmée, et la mitose qui correspond à la phase M et au processus de division cellulaire. On considère parfois une phase de quiescence, plus généralement nommée G0, pour les cellules ne proliférant. Cela correspond à une phase qui peut être permanente où la cellule n'a plus de comportement prolifératif. Une cellule peut entrer dans la phase G0 depuis la phase G1 et reprendre son cycle depuis la phase G1. Les transitions entre les différentes phases du cycle cellulaires sont régulées par des mécanismes moléculaires appelés points de contrôle[Murray, 1994, Johnson and Walker, 1999].

Certaines molécules agissent directement sur la structure interne des cellules perturbant ainsi le déroulement du cycle cellulaire en ralentissant certaines phases du cycle voire en bloquant une cellule dans une des phases. Les cellules cancéreuses étant des cellules très prolifératives, ces molécules sont souvent étudiées voire utilisées pour traiter des tumeurs cancéreuses et limiter leur capacité à se diviser très rapidement voire à déclencher des mécanismes d'apoptose dans certains cas. Le nocodazole, par exemple, a la capacité de bloquer les cellules en phase M [Nagano et al., 2009, Choi et al., 2011], un bloquage prolongé en phase M peut enclencher le processus d'apoptose. Cette molécule est notamment étudiée pour ses propriétés anti-tumorales et utilisée pour synchroniser des populations de cellules. De la même manière, le palbociclib (Ibrance®) développé par les laboratoires Pfizer a la capacité de bloquer les cellules au point de Restriction de la phase G1 et est autorisé et utilisé dans plusieurs pays pour traiter le cancer du sein [Cadoo et al., 2014, Dhillon, 2015, Beaver et al., 2015].

6.2.1.2 Fonctionnement du modèle

L'objectif de ce modèle *in silico* était de parvenir à une représentation réaliste du gradient de prolifération au sein des sphéroïdes et de la dynamique du cycle cellulaire associée. Ces paramètres jouent un rôle crucial dans la réponse aux molécules anti-tumorales.

Le modèle intègre un module de physique masse-ressort [Van Liedekerke et al., 2018] pour simuler la forme du sphéroïde. Chaque cellule est connectée à ses voisines comme s'il y avait un ressort reliant leurs centres. En outre, un module de gradient d'oxygène basé sur un article de Grimes et al. [Grimes et al., 2014] est utilisé pour gérer les quantités d'oxygène à l'intérieur du sphéroïde.

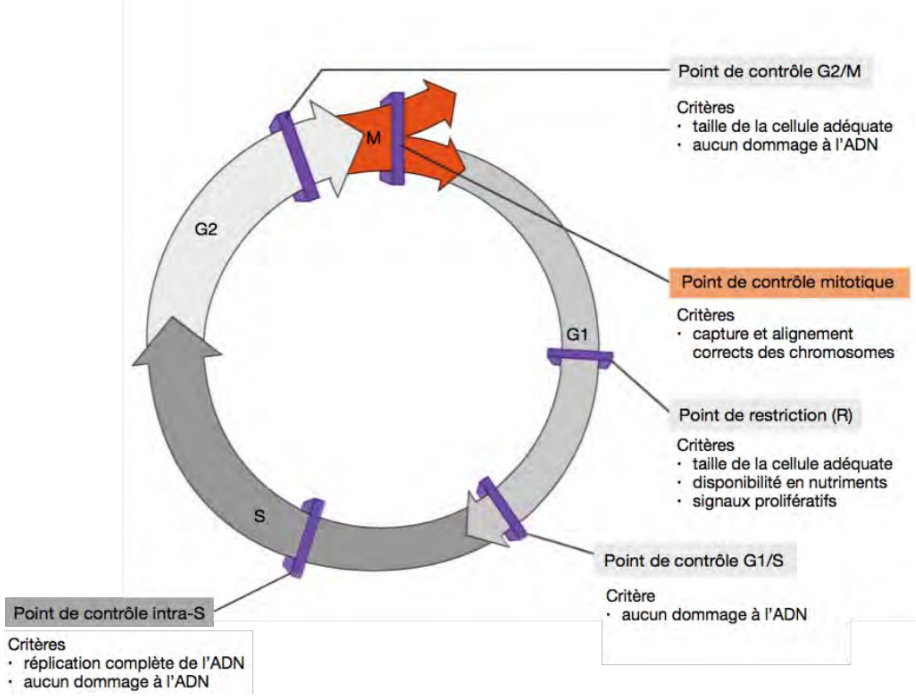


FIGURE 6.11 – **Schéma du cycle cellulaire**
Une cellule qui vient de naître entre directement en phase G1 et se prépare à doubler son ADN. Certains mécanismes internes permettent de déclencher la fin de la phase G1 si les conditions sont réunies symbolisés par le point de restriction. Durant la phase S, la cellule duplique son ADN. De la même manière, des mécanismes contrôlent le bon déroulement de la phase et le déclenchement de sa fin et sont symbolisés par un point de contrôle. La phase G2 prépare à la division cellulaire et le point de contrôle représente les mécanismes internes de contrôle du déclenchement de la mitose. Enfin la phase M correspond à l'étape de division cellulaire et à la naissance de deux nouvelles cellules débutant leur phase G1.
Source : [Andrieu, 2014]

Le comportement est divisé en différentes phases du cycle cellulaire (illustrées dans la Figure 6.12), et les durées de ces phases peuvent être ajustées pour correspondre à des cycles spécifiques en fonction de la lignée étudiée. Cela permet de tester différentes dynamiques de population en modifiant les règles et les paramètres du cycle. Par exemple, l'impact de l'allongement de la phase G1 en fonction de l'oxygénation ou l'effet du taux de quiescence peuvent être explorés. De même, l'inoculation de molécules perturbant le cycle cellulaire peut être testée pour reproduire leurs effets sur la culture.

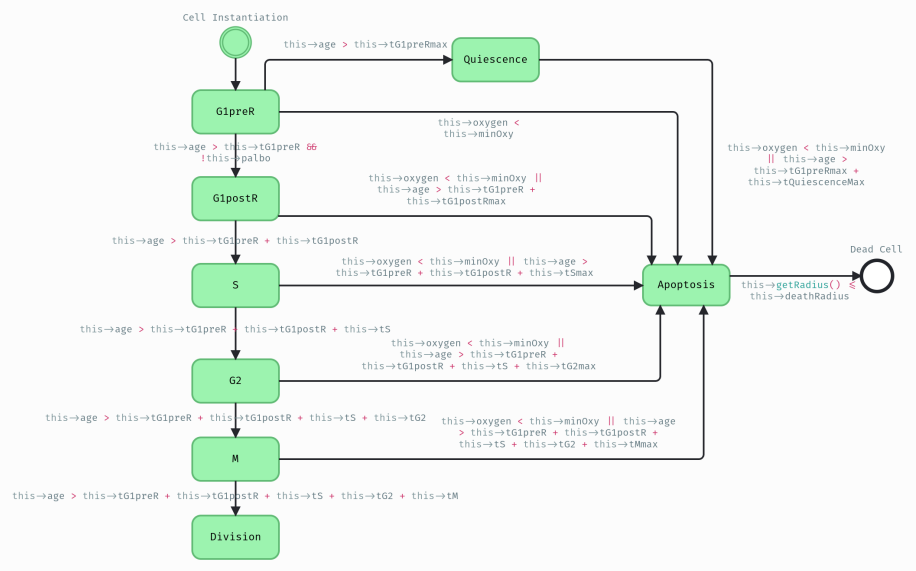


FIGURE 6.12 – **Diagramme états-transitions du modèle de sphéroïde cancéreux.**
Le comportement est réparti entre les différents états du cycle cellulaire. Les cellules peuvent également entrer dans un état de quiescence ou d'apoptose. Les cellules sont retirées de la simulation après avoir rétréci jusqu'à un rayon minimal.

6.2.1.3 Apport du modèle à la méthodologie

Ce modèle a permis pour la première fois d'éprouver notre plateforme et notre méthodologie dans un contexte pour lesquels ils ont été conçus. L'outil d'exploration n'existait pas encore ce qui nous a permis de surtout tester les premières phases de notre approche pour générer un premier prototype.

Notre approche a rendu possible la création rapide d'un modèle initial avec la visualisation instantanée permise par CellViewer (voir Figure 6.13). Cela a permis d'apporter des corrections immédiates à divers paramètres. Un autre aspect remarquable est la représentation graphique des règles du modèle, qui offre aux biologistes un moyen plus intuitif d'interagir avec le modèle par rapport au code traditionnel. Cette représentation graphique est essentielle pour favoriser l'acceptation et l'engagement des biologistes dans le travail interdisciplinaire.

6.2.1.4 Apport du modèle à la plateforme

Ce modèle a permis d'intégrer, d'adapter et de raffiner les modules de physique masse-ressort et de diffusion d'oxygène développés au cours de la thèse de David Bernard [Bernard, 2020], enrichissant ainsi la plateforme de deux nou-

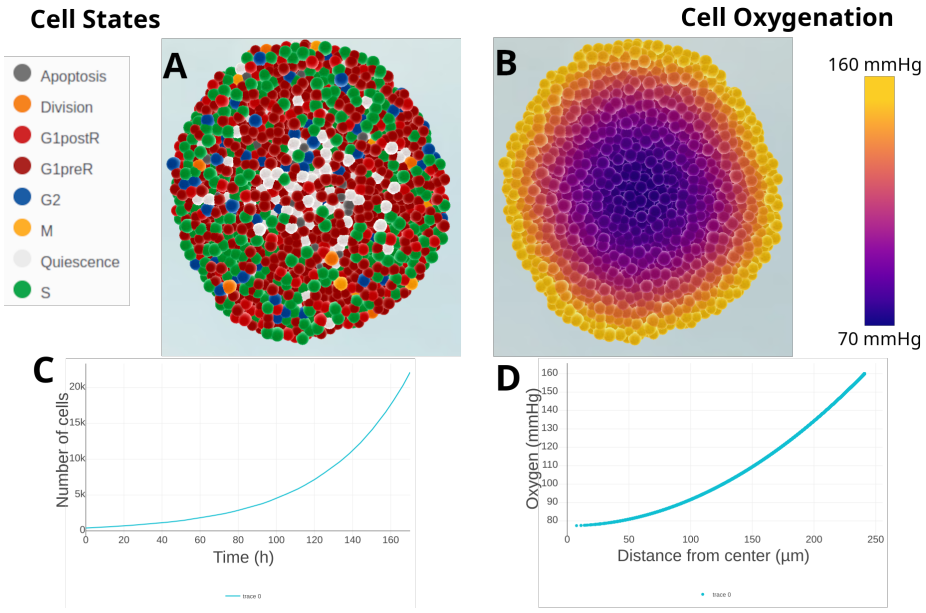


FIGURE 6.13 – **Visualisations en coupes et courbes d'un sphéroïde de plus de 20 000 cellules grâce à ISiCell Viewer.**
(A) Décrit l'état de chaque cellule, les cellules les moins prolifératives se trouvent au centre. (B) Courbe du gradient d'oxygène dans le sphéroïde de près de 250 µm. (C) Décrit la croissance exponentielle du sphéroïde en termes de nombre de cellules. (D) Montre les niveaux d'oxygénation en fonction de la distance par rapport au centre. En l'absence de données expérimentales, les biologistes peuvent toujours valider qualitativement la cohérence du sphéroïde simulé en utilisant l'outil ISiCell Viewer. Ce modèle est disponible à l'adresse suivante : https://isicell.irit.fr/app.html?demo=Spheroid.

veaux modules.

Ce modèle s'est également avéré utile pour évaluer le temps de calcul de la plateforme en fonction du nombre de cellules. Le nombre de cellules croît de manière exponentielle et, par conséquent, le temps de calcul nécessaire pour mettre à jour le comportement de toutes les cellule augmente, en particulier lors du calcul de leurs nouvelles positions avec le module de physique. La Figure 6.14 montre une comparaison du temps de calcul entre les trois modes d'utilisation de la simulation générée. Les résultats montrent que la visualisation et le code C++ interfacé au Python sont respectivement, en moyenne, 2,1 et 1,3 fois plus lents, que l'exécution du code C++ seul.

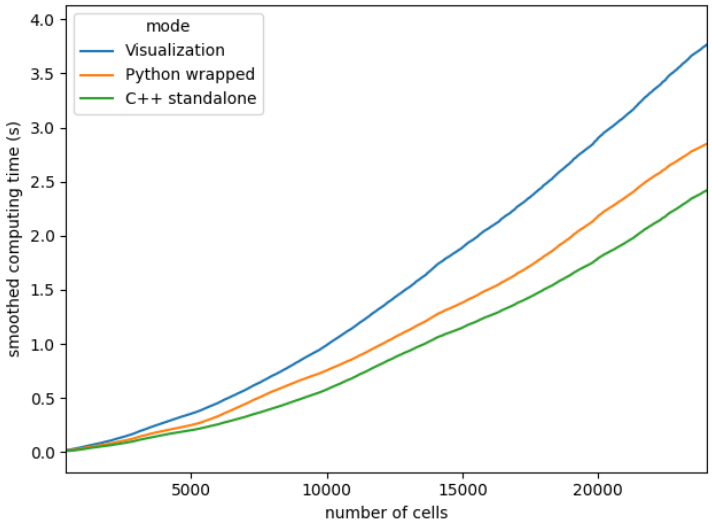


FIGURE 6.14 – **Comparaison des temps de calcul**
Courbes des temps de calcul lissées montrant le temps nécessaire pour calculer une étape de la simulation en fonction du nombre de cellules. Le temps de calcul a été calculé avec la même graine sur une simulation de sphéroïde de 200 heures avec un CPU Intel® Core™ i7-10850H de 2,70 GHz et 32 Go de RAM. Chaque étape de simulation représente 10 minutes dans la simulation. La visualisation et l'exécution interfacé au Python multiplient respectivement le temps de calcul par des facteurs moyens de 2,1 et 1,3.

6.2.2 Modèle de neurogenèse

La mise en place du système nerveux se déroule par un processus appelé neurogenèse, qui permet la différentiation d'une cellule souche neurale en un neurone fonctionnel. L'étude de ce phénomène dans des organismes aussi variées que des drosophiles [Hartenstein and Wodarz, 2013, Fernández-Hernández et al., 2013, Younossi-Hartenstein et al., 1996, Campos-Ortega, 1993], des poissons [Zupanc, 2006, Ganz and Brand, 2016, Zupanc, 2008, Zupanc, 2001] et chez des mammifères [Götz and Huttner, 2005, Gage, 2019, Rakic, 2002, Gould and Gross, 2002, Bonfanti and Peretto, 2011, Lipp and Bonfanti, 2016] a permis au biologistes de développer des modèles et des protocoles *in vitro* et *in vivo* pour étudier les mécanismes de neurogenèse tant lors du développement initial chez l'embryon que lors de la production de nouveaux neurones chez les individus adultes.

Des modèles in silico de neurogenèse ont également été développés pour éprouver les connaissances sur ce phénomène. Ainsi on peut compter les modèles de neurogenèse sous-ventriculaire chez l'adulte de Kurhekar et al. [Kurhekar and Deshpande, 2015, Kurhekar and Deshpande, 2014], d'étude du cortex de souris

et de rats [Gohlke et al., 2004, Postel et al., 2019] ou de l'hippocampe [Weisz and Argibay, 2009, Becker et al., 2009].

Le modèle étudié dans la suite de ce chapitre s'intéresse à la formation des neurones moteurs dans le tube neural, ébauche embryonnaire de la moelle spinale, de l'embryon de poulet. Ce modèle fait suite à de précédents travaux des biologistes avec lesquelles nous avons collaboré [Saade et al., 2013, Azaïs et al., 2019, Molina et al., 2022]. Ce modèle a été réalisée en collaboration avec Dr. Delphine Bretonnière, Pr. Eric Agius et Pr. Valérie Lobjois.

6.2.2.1 Contexte biologique

Durant le développement de l'embryon chez les vertébrés, les neurones se forment au sein du tube neural (voir Figure 6.15). Ce mécanisme est régulé par des gradients de molécules émises depuis la plaque du toit côté dorsal et de la plaque du plancher côté ventral [Stamataki et al., 2005, Balaskas et al., 2012, Rifes et al., 2020].

Les neurones moteurs se forment à partir de cellules progénitrices situées dans la région ventrale du tube neural et font partie des premières cellules à se différentier en neurones fonctionnels au cours du développement. Les cellules de cette région expriment le facteur de transcription OLIG2 [Park et al., 2002, Novitch et al., 2001] et les neurones moteurs le facteur de transcription ISLET1 [Qu et al., 2014, Moreno and Ribera, 2014], ce qui permet de mettre en évidence les deux types cellulaires au sein d'un tissus par un immunomarquage spécifique de chacune de ces protéines.

Chez les Vertébrés, le développement du tube neural suit des règles similaires [Goldstein et al., 2002, Garcia-Lopez et al., 2009]. Ainsi l'usage d'embryons de poulet représente un modèle biologique intéressant pour effectuer des expérimentations *in vivo*. En effet, l'embryon est localisé dans un œuf donc facilement accessible, le développement embryonnaire dure une vingtaine de jours et le processus de neurogenèse spinale débute dès le deuxième et dure 4 jours. Cela permet de facilement étudier les mécanismes de neurogenèse spinale de bout en bout et de disposer d'un nombre conséquent d'individus pour un coût restreint.

6.2.2.2 Première itération du modèle

Ce modèle, contrairement au précédent, n'est pas basé sur une ancienne collaboration ayant déjà abouti à un modèle multi-agents. Ceci nous a permis d'appliquer notre méthodologie dès les premières interactions et d'enrichir itérativement le modèle au fur et à mesure de la collaboration. De plus, un modèle mathématique issu des travaux d'Azaïs et al. [Azaïs et al., 2019] a permis d'orienter certaines directions du modèle développé par la plateforme et d'enrichir en retour le modèle mathématique précédemment étudié.

Définition de l'environnement

Lors de la première itération, nous avons souhaité représenter le mouvement des noyaux associé au cycle cellulaire au sein du tube neural (appelé mouve-

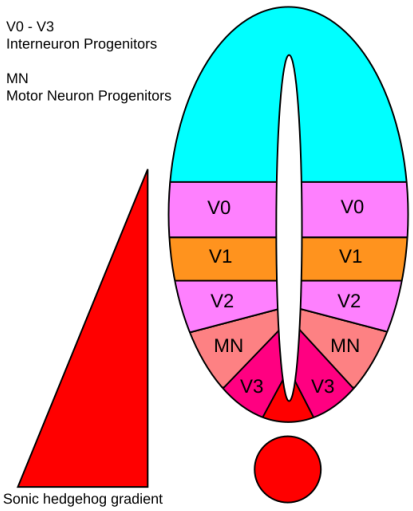


FIGURE 6.15 – **Domaines des cellules progénitrices du tube neural**
Le tube neural est subdivisé de la plaque du plancher côté ventral (en bas) jusqu'à la plaque du toit, côté dorsal (en haut) en différentes régions qui vont produire différents types de neurones. Côté ventral, la chorde produit la molécule Sonic hedgehog (Shh) ce qui permet la formation d'un gradient de cette molécule selon l'axe ventro-dorsale (plus forte concentration côté ventral). C'est ce gradient qui va permettre la spécialisation des cellules progénitrices. Côté ventral, on observe 3 régions de formations d'interneurones (V0 à V3) ainsi que la région de formation des neurones moteurs (MN).
Source : utilisateur wikimedia Fred the Oyster sous la licence Creative Commons Attribution-ShareAlike 4.0 International

ment interkinétique nucléaire, figure 6.15). En effet, dans le neuroépithelium, les noyaux sont localisés de la zone apicale lors de la mitose, puis migrent vers la région basale en phase G1 restent dans la zone basale lors de la phase S puis retournent vers la région apicale en phase G2. Cela nécessite donc de prendre en compte les contraintes physiques et la motilité des noyaux dans le tube. D'un point de vue spatio-temporel, les ordres de grandeur sont les suivants :

— la largeur de la subsection étudiée du tube est d'une quarantaine de microns ;
— les progéniteurs ont une durée de cycle moyenne aux alentours d'une douzaine d'heure ;
— les noyaux des progéniteurs effectuent un aller retour apical-basal-apical durant un cycle.

Ainsi, afin de disposer d'une granularité temporelle suffisante pour décrire les mouvements des noyaux au cours de leur cycle, nous avons pris un pas de temps de 5mn. De plus, pour gérer les forces de mouvements ainsi que l'encombrement, nous avons utilisé le module de physique hertzienne. Les données des biologistes

correspondent à des stades de développement de l'embryon que l'on peut associer à des temps selon une table de correspondance (voir table 6.1) allant des stades 12 à 25.

Stade	Heures post fécondation
12	44.9
13	49.7
14	54.4
15	58.9
16	63.3
17	67.7
18	72.2
19	76.7
20	81.4
21	86.3
22	91.6
23	97.2
24	103.2
25	109.6

TABLE 6.1 – **Table de correspondance stade / heures post fécondation**

Description des agents

Ce modèle fait intervenir deux types d'agents : les Progéniteurs (P) et les Neurones (N) qui disposent des caractéristiques suivantes :

— un âge qui représente leur avancement dans leur cycle en heures ;
— un temps de cycle en heures tiré aléatoirement selon une loi de probabilité à partir d'une moyenne et d'un écart-type ;
— des paramètres de régulation des modes de division ;
— un rayon nucléaire en microns ;
— un vecteur de vitesse de déplacement en µm/s.

Seuls les noyaux des agents sont physiquement représentés dans ce premier prototype.

Comportement des agents

Le schéma état-transition (voir Figure 6.16) utilisé pour ce modèle est basé sur celui du modèle de sphéroïde de la subsection précédente qui décompose le comportement selon les phases du cycle cellulaire.

Les neurones n'ont qu'un seul état (Quiescence) accessible dans lequel leur noyau va se déplacer vers la base à l'extérieur du tube neural jusqu'à s'y empiler. Les noyaux des progéniteurs, en fonction de leur phase, vont se déplacer entre la base et la lumière du tube comme suit :

— en phase G1, ils se dirigent vers l'extérieur du tube ;

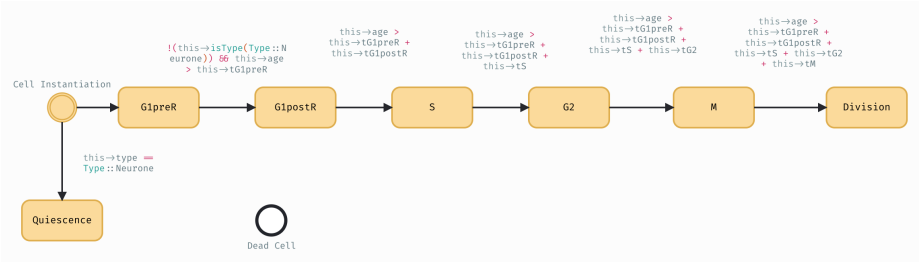


FIGURE 6.16 – **Diagramme état-transition du modèle de Neurogénèse chez l'embryon de poulet.**
Les neurones après l'initialisation entrent directement en quiescence tandis que les progéniteurs entrent en phase G1 correspondant aux stade G1preR et G1postR correspondants à la phase G1 avant et après le point de restriction R et avancent dans leur cycle jusqu'à se diviser dans le stade Division.

— en phase S, les noyaux restent du côté extérieur du tube et s'y déplacent aléatoirement selon l'axe apico-basal ;
— en phase G2 et M, ils se dirigent vers la lumière du tube où la division aura lieu.

Ce déplacement des noyaux est un phénomène connu et décrit en biologie comme illustré dans la Figure 6.17.

Au moment de la division, un progéniteur pourra se diviser selon les 3 modes de division suivants :

— **PP** : les deux cellules filles seront des cellules P ;
— **PN** : une cellule fille sera de type P et l'autre de type N ;
— **NN** : les deux cellules filles seront des cellules N.

Le "choix" du mode de division s'effectue à la fin de la mitose et dépend de 5 paramètres partagés par l'ensemble des cellules reprenant les équations de l'article d'Azaïs et al. (voir équation 6.1).

$$\begin{cases} \alpha_{PP}(t) &= \frac{1}{2}[1 - tanh(\frac{t-\tau_{PP}}{\sigma_{PP}})] \\ \alpha_{NN}(t) &= \frac{1}{2}\alpha_{NN}(t \to \infty)[1 + tanh(\frac{t-\tau_{NN}}{\sigma_{NN}})] \\ \alpha_{PN}(t) &= 1 - \alpha_{PP}(t) - \alpha_{NN}(t) \end{cases} \tag{6.1}$$

où

— $\alpha_{PP}(t), \alpha_{PN}(t), \alpha_{NN}(t)$ correspondent respectivement aux probabilités de faire une division PP, PN ou NN à l'instant t ;
— τ_{PP}, τ_{NN} correspondent respectivement aux points de flexion en heures des courbes d'$\alpha_{PP}(t), \alpha_{NN}(t)$;
— σ_{PP}, σ_{NN} correspondent respectivement aux étalements en heures des courbes d'$\alpha_{PP}(t), \alpha_{NN}(t)$.

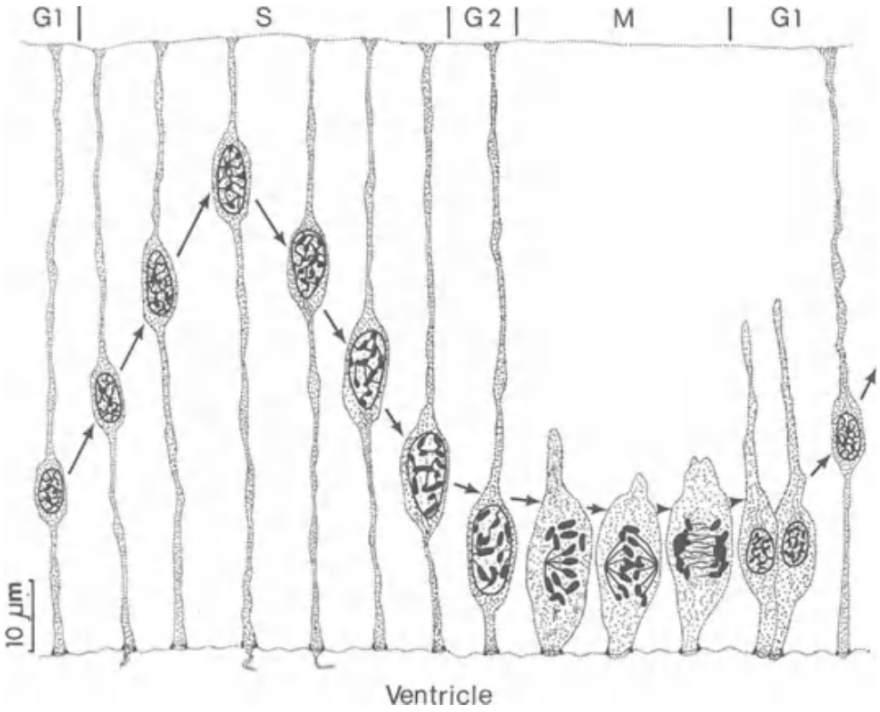


FIGURE 6.17 – **Mouvement des noyaux de cellules progénitrices au sein du tube neural.**
Les noyaux des progéniteurs se déplacent entre la base et la lumière du tube tout au long de leur cycle.
Source : [Jacobson and Jacobson, 1991]

Les valeurs par défaut sont celles issus des travaux d'Azaïs et al. et les courbes correspondantes sont visibles dans la Figure 6.18

Définition du protocole expérimental

Pour initialiser la simulation on dispose d'un nombre aléatoire (tiré selon une loi normale) de progéniteurs disposés spatialement selon leur phase du cycle. Leur avancement dans leur cycle est tiré aléatoirement selon une loi exponentielle de paramètre ln(2) et on obtient son âge en multipliant ce tirage par la durée du cycle.

Dans la première version du modèle, il n'y avait pas de mécanisme à l'échelle de la population implémenté en dehors de la mise en place.

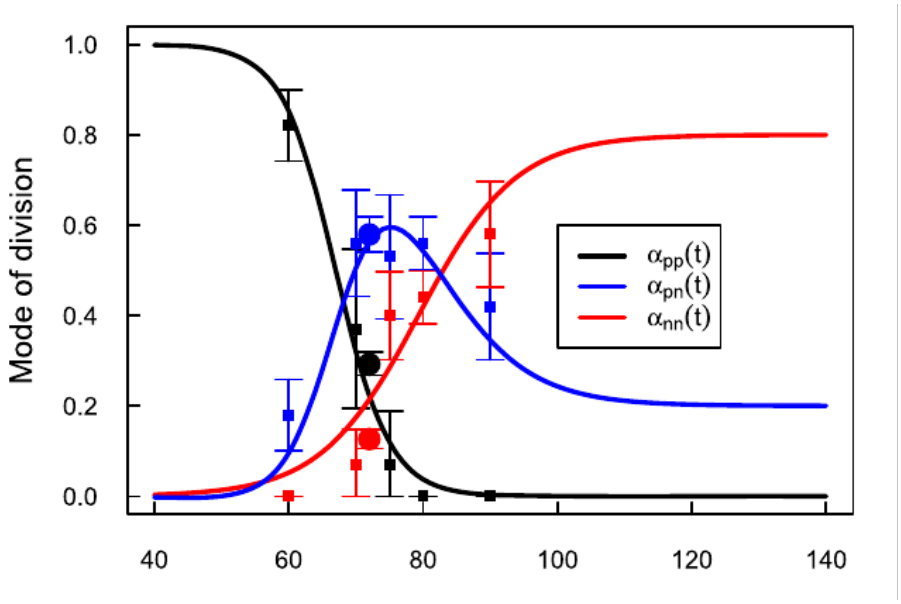


FIGURE 6.18 – **Courbes de probabilité des modes de divisions.** Les courbes correspondent aux données issues des travaux d'Azaïs et al. Source : [Azaïs et al., 2019]

Validation du premier prototype

Afin de valider qualitativement ce prototype, la visualisation (voir Figure 6.19) a permis de vérifier que le comportement des noyaux et que les dynamiques étaient bien celles attendues par les biologistes.

Le modèle permet donc bien de décrire le mouvement des noyaux et la répartition spatiale selon l'axe apico-basal des noyaux selon leur phase ainsi que les dynamiques temporelles de populations entre neurones et progéniteurs.

6.2.2.3 Nouvelles hypothèses

Dans la suite des itérations, le mouvement des noyaux et la physique associée ont été désactivées. En effet, la modélisation de la physique des noyaux n'était pas satisfaisante car nos modèles de physique de ne permettent pas de décrire correctement le déplacement des noyaux longitudinaux au sein de cellules elles-mêmes longitudinales. De plus, ces mécaniques accaparent le plus de ressources computationelles et nous ne disposions pas de données pour calibrer ces mécaniques spatialisées.

Pour calibrer le modèle, nous disposons de :

— données de dynamiques de population (nombre de progéniteurs et de neurones par stade),

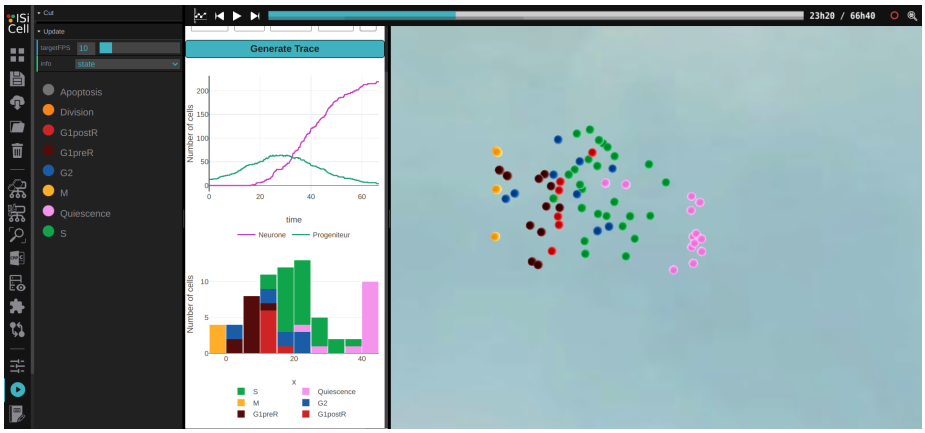


FIGURE 6.19 – **Visualisation avec ISiCell Viewer du premier prototype validé**
Les deux diagrammes et la visualisation des cellules colorées selon leur état (basé sur le cycle cellulaire) permettent de montrer la dynamique temporelle des populations de progéniteurs et de neurones ainsi que la répartition spatiale des noyaux selon leur phase.

— données qualitatives sur les dynamiques de modes de division,
— de matrices de descendances clonales.

Les itérations suivantes du modèle n'étant pas spatialisées, la validation qualitative s'est faite par rapport aux courbes permettant de comparer les résultats de simulation aux données expérimentales grâce aux outils d'exploration et d'analyse de la plateforme.

Descendance clonale

Un des objectifs du travail est de répondre à la question sur le type de progéniteurs présents dans la moelle spinale. Un premier modèle appelé PN postule que le tube neural est constitué par une population homogène de progéniteurs effectuent les 3 modes de division dont les pourcentages évoluent au cours du temps. Un deuxième modèle appelé GAN décrit des progéniteurs hétérogènes divisés en deux groupes avec des progéniteurs jeunes réalisant des divisions PP ou PN (appelés G) et des précurseurs issus d'une division asymétrique PN qui ne seront plus capable d'opérer des divisions symétriques (PP) et effectuant les divisions PN et NN (appelé A).

Le modèle PN peut avoir un comportement similaire au modèle GAN (l'inverse n'est pas possible) si aucun progéniteur issu d'une division PN n'effectue de division PP après cet événement. Nous avons introduit un coefficient de ressemblance GAN (nommé GANlike) qui correspond au pourcentage de simulations avec un comportement GAN sur mille réplicas. Ainsi un jeu de paramètres avec

une valeur de GANlike de 50% signifie que la moitié des réplicas n'ont pas eu de progéniteur issu d'une division PN qui a effectué une division PP ensuite. Cette mesure nous permet de distinguer les jeux de paramètres amenant à des simulations indistinguables de modèle GAN (GANlike = 100%).

Dans le travail d'Azaïs, il a été montré que les arbres de descendance sont différents dans les 2 modèles. Un des moyens pour déterminer le type de progéniteurs existant dans le tubes neural est donc de comparer les profils de descendances clonales des progéniteurs obtenus dans les 2 modèles avec les profils obtenus expérimentalement. Dans le cadre de notre modèle, nous avons implémenté un modèle de type PN.

Ces profils correspondent à des matrices avec en colonnes le nombre de neurones et un ligne le nombre de progéniteurs. Les valeurs contenues dans la matrice correspondent au pourcentage de progéniteurs présentant le profil iP jN (i Progéniteurs et j Neurones issus du même premier progéniteur) à la fin de la période étudiée. Ainsi un progéniteur ayant fait une division PP et dont les deux cellules filles ont également fait une division PP sur la période présentera un profil 4P 0N, si l'une des deux cellules filles ne se divise pas et que l'autre fait une division NN, le profil sera 1P 2N. La Figure 6.20 illustre le format de données que l'on essaie de reproduire avec le modèle.

La premier prototype ne permettait pas de comparer le modèle aux données de descendances clonales. Il a donc fallu développer un module nommé *CellLineManager* propre au modèle pour gérer les mécanismes permettant de générer des matrices comparables à celles obtenues par les biologistes.

Ce module est constitué d'une classe *TreeNode* permettant de gérer les arbres de descendances de chaque cellule. Cette classe contient un identifiant, le pas de simulation de naissance et de division (initialisé à -1) de la cellule, son type et les pointeurs vers les noeuds de la cellule mère et des cellules-filles s'ils existent (initialisés à nullptr). De plus, elle dispose d'une méthode permettant de renvoyer un vecteur d'entier avec toutes les informations afin de faciliter le traitement côté Python (les pointeurs sont remplacés par les identifiants des cellules mères et filles si le pointeur n'est pas nul).

La classe *BodyCellLineManager* contient un pointeur vers le noeud correspondant à la cellule ainsi qu'un autre vers sa cellule soeur si elle existe (initialisé à nullptr) et tous les *getters* et *setters* pour ses deux attributs.

La classe *PluginCellLineManager* dispose, quand à elle, d'un compteur initialisé à zéro pour gérer les identifiants des noeuds, d'un vecteur contenant les noeuds de toutes les cellules présentes à l'initialisation, de méthodes pour ajouter de nouveaux noeuds à ce vecteur, si des cellules sont ajoutées au domaine par d'autres mécanismes que la division, et pour supprimer tous les arbres de lignées. Dans la méthode *onAddCell*, on parcourt l'ensemble des cellules apparues durant le pas de simulation courant, si la cellule a une soeur (elle est issue d'une division, une seule nouvelle cellule est générée, la "mère" étant réinitialisée) alors on met à jour toutes les informations des noeuds pour la cellule, sa soeur et sa mère.

Pour générer les profils, on récupère l'ensemble des progéniteurs présents au départ de l'étude par exemple 48h et on détermine le profil en comptant tous les

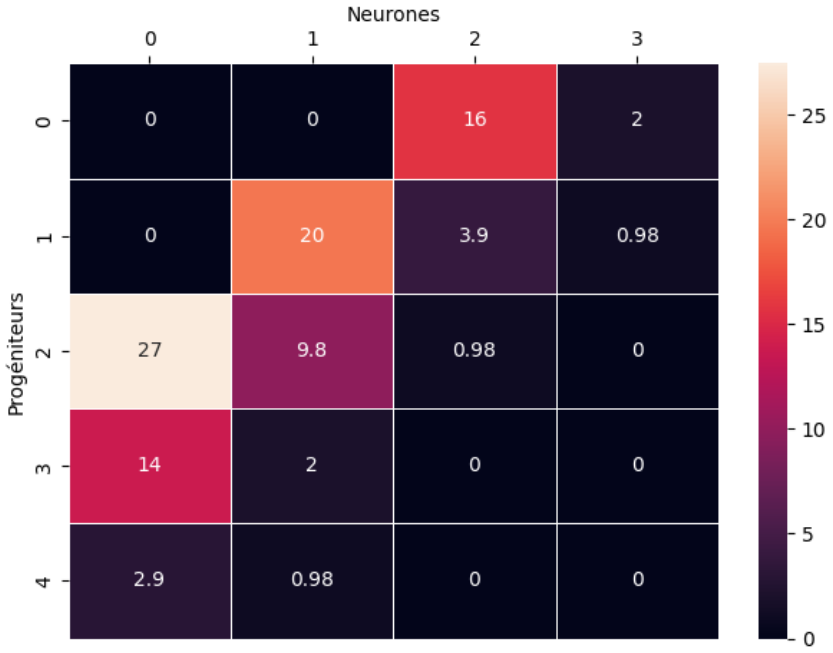


FIGURE 6.20 – **Matrice des profils de descendances clonales sur la période 48h - 72h post fécondation**
Les profils les plus fréquents sur la période sont 2P 0N (27%), 1P 1N (20%), 0P 2N (16%), 3P 0N (14%).

progéniteurs et les neurones issus de cette cellule à la fin de l'étude par exemple à 72h en reconstruisant l'arbre depuis les cellules présentes à la fin (on remonte jusqu'à tomber sur un des progéniteurs de départ ou sur une cellule sans mère). On obtient ensuite le compte de chaque profil, il suffit ensuite de diviser par le nombre total de profil pour obtenir le pourcentage comme dans les données expérimentales.

Plusieurs tentatives d'optimisation à base d'algorithmes génétiques (NSGA-II [Deb et al., 2002, Yusoff et al., 2011, Verma et al., 2021]) n'ont pas permis d'obtenir des résultats concluants. Nous disposions de 3 matrices de profils (48h+24h, 48h+48h, 72h+24h) et notre fonction objectif se composait, entre autres, de 3 objectifs de minimisation de l'erreur carrée moyenne de ces trois matrices par rapport à celles de nos simulations. Les meilleures résultats étaient fortement corrélés à une plus grande erreur sur les dynamiques de population (tendance à produire beaucoup plus de progéniteurs et de neurones). De plus, les meilleurs profils obtenus n'étaient pas satisfaisants car trop éloignés des données expérimentales. Pour ces raisons, ces données n'ont pas été utilisées dans

les itérations suivantes pour se concentrer sur les dynamiques de population. Cela a permis de ne pas complexifier davantage le modèle et diminuer le volume de données à générer et analyser. Dans les itérations suivantes, le module de gestion des lignées cellulaires a été désactivé.

Cycle cellulaire

La gestion du cycle cellulaire a été le sujet de plusieurs itérations. En effet, plusieurs mécanismes ont été mis en place pour gérer la durée du cycle et des phases.

Premièrement, les discussions ont porté sur la gestion de la durée du cycle sur son ensemble ou phase par phase. En effet, dans l'article de Molina et al. [Molina et al., 2022], des durées de phases ont été calculées avec un temps de cycle moyen supérieur à 14h. Cela ouvre donc la possibilité à traiter chaque phase indépendemment et à tirer aléatoirement la durée de chacune à la naissance de chaque cellule. De l'autre côté, les modèles issus des travaux d'Azaïs et al. [Azaïs et al., 2019] utilisent une loi exponentielle pour décrire la durée du cycle avec un temps minimal de 10h, un écart-type de 2h pour une moyenne de 12h. Les deux approches ont été implémentées mais la deuxième solution a été privilégiée pour diminuer le nombre de paramètres du modèles et conséquemment à l'abandon de la spatialisation des noyaux. Nous avons également confronté cette hypothèse à une loi log-normale de moyenne et écart-type similaire pour comparer l'effet de la loi sur les dynamiques mais aucune différence significative n'a pu être mise en évidence. De plus, la loi exponentielle permet de disposer d'un temps de cycle minimum, c'est pour cela que le modèle a gardé cette implémentation du cycle.

Deuxièmement, il a été observé que le cycle des progéniteurs s'allongent au fur et à mesure des divisions successives. Nous avons donc implémenté un mécanisme d'allongement du cycle (principalement au niveau de la phase G1) après chaque division qui dépend du type de division (PP ou PN). En fonction des hypothèses sur la gestion du cycle, l'allongement a été implémenté de différentes manières :

- en considérant les phases indépendemment, 4 paramètres augmentant la durée moyenne et l'écart-type selon le mode de division ;
- avec la loi log-normale, 4 paramètres augmentant la durée moyenne et l'écart-type selon le mode de division ;
- avec la loi exponentielle, 2 paramètres augmentant l'écart-type selon le mode de division.

Le choix de la loi exponentielle a également été favorisé par rapport aux mécanismes d'allongement. En effet, dans une loi exponentielle de paramètre λ l'écart-type et la moyenne sont égaux et valent $1/\lambda$. On peut donc relier l'augmentation de la durée moyenne de cycle à une augmentation de l'hétérogénéité des durées de cycles dans le temps. Nous avons également considéré l'hypothèse que l'allongement n'était pas dépendant de la division précédente mais de la division en cours. Cela impliquait d'ajouter un paramètre pour les divisions NN et de calculer à la naissance les probabilités de modes de division de la cellule en se projetant un temps de cycle moyen allongé plus tard. Cette hypothèse

n'a pas été retenu car elle ajoutait un paramètre au modèle et surtout car elle rendait plus difficile la calibration du modèle principalement à cause du tirage du mode de division avant la durée de cycle.

Source externe

Malgré toutes les hypothèses précédentes, il n'était pas possible avec les premières itérations de ce modèle de reproduire les dynamiques de population des jeux de données expérimentaux sans diminuer drastiquement le temps de cycle d'où les propositions de modifier les lois de probabilité du temps de cycle pour autoriser des temps de cycle plus courts.

Dans l'article de Saade et al. [Saade et al., 2013] une source externe de progéniteurs est ajoutée à leurs équations pour compenser la croissance initiale de la population de progéniteurs. Cette source est constante du stade 12 jusqu'à leur pic expérimental à 73h post fécondation d'une valeur β de 0,9 progéniteurs par heure. Ce phénomène peut s'expliquer par le gradient de Shh qui s'intensifie sur les premiers stade et peut ajouter de nouveaux progéniteurs au domaine en les faisant exprimer le gène OLIG2.

Dans un premier temps, nous avons implémenté un mécanisme de source externe constante entre deux bornes. Cela nous a permis de nous rapprocher des quantités observées expérimentalement et d'obtenir un nombre de neurones finaux similaire aux données. Cependant cela n'était pas suffisant pour suivre plus finement les dynamiques de population.

En implémentant cette nouvelle hypothèse, nous nous sommes posés la question des dynamiques de modes de division des nouveaux progéniteurs intégrés au domaine OLIG2. Le parti pris initial était que le «choix» du mode de division était indépendant des cellules et de leurs divisions précédentes. Ainsi toutes les cellules du domaines partagent la même «horloge» biologique de l'avancement de la neurogenèse et ont les mêmes probabilités de mode de division dépendant uniquement du temps écoulé depuis la fécondation. Ce phénomène pourrait être biologiquement expliqué par la présence de gradient de molécules dans le domaine (comme Shh) permettant de signaler aux cellules l'avancement de la neurogenèse. L'autre hypothèse qui a été implémentée est celle que le «choix» du mode de division est lié à une «horloge» interne aux cellules qui dépend de leur entrée dans le domaine OLIG2. Cette hypothèse s'est avérée infructueuse à reproduire les données expérimentales. En effet, les progéniteurs ajoutés continuent, d'autant plus longtemps qu'ils ont été ajoutés tard, à faire des divisions PP et PN ce qui retarde et augmente le pic de progéniteurs. De plus, cela complique d'autant plus la calibration des probabilités de mode de division car chaque progéniteur aura des probabilités différentes de se diviser en fonction de leur arrivée dans le domaine. Pour ces raisons, cette hypothèse a été mise de côté.

Pour affiner cette mécanique, nous avons décomposé la source externe en fonction des stades de l'embryon. Ainsi, la quantité de progéniteurs ajoutés par heure dépend du stade, ce qui permet d'ajuster la source en fonction du stade et de potentiellement mettre en évidence, pendant la calibration, la prévalence

d'un stade sur les autres vis à vis de la contribution à l'activation de nouveaux progéniteurs. Nous avons ainsi pu obtenir de meilleurs résultats en implémentant cette nouvelle hypothèse.

6.2.2.4 Analyse des résultats

Après avoir obtenu un modèle capable de reproduire les dynamiques de populations attendues, nous avons entamé un processus d'exploration de l'espace des paramètres d'entrée pour chercher les jeux de paramètres les plus pertinents par rapport aux données expérimentales. Le but est de trouver, parmi les meilleurs jeux, quels sont les critères qui permettent d'obtenir les simulations reproduisant au mieux les dynamiques expérimentales.

Dans la suite de cette section, nous détaillerons les différentes étapes des analyses effectuées sur la version la plus récente du modèle vis-à-vis des hypothèses conservées.

Définition de l'espace d'exploration et génération des données

Dans un premier temps, nous avons commencé par déterminer les paramètres sur lesquels nous souhaitions réaliser l'exploration et les plages de valeurs acceptables pour ces différents paramètres. Les ordres de grandeurs sont issus de discussions avec les biologistes pour décider de plages biologiquement cohérentes avec les connaissances pré-établies.

Les paramètres que nous explorons se concentrent sur trois mécaniques du modèle. Dans un premier temps, nous nous intéressons aux valeurs des sources externes entre les stades 12 et 17 inclus. Dans un second temps, nous explorons les valeurs des paramètres qui contrôlent les probabilités de modes de division. Enfin, les deux derniers paramètres que nous explorons sont ceux des allongements des durées de cycle par l'augmentation de la variance des lois de tirage des durées de cycle. Ainsi, notre espace d'exploration comprend treize paramètres dans les plages suivantes :

— $\beta_{s\in[12,17]} \in [0;5]h^{-1}$
— $\alpha_{NN}(t \to \infty) \in [0,5;1]$
— $\sigma_{PP}, \sigma_{NN} \in [0,1;20]h$
— $\tau_{PP} \in [40;80]h$
— $\tau_{PP} \in [70;100]h$
— $\epsilon_{PP}, \epsilon_{NN} \in [0;5]h$

Pour explorer une part significative et représentative de cet espace, nous avons utilisé un échantillonage en hypercube latin pour tirer les différents jeux de paramètres dans notre espace d'entrées. De plus, pour éviter d'évaluer des simulations dont les probabilités de modes de division ne correspondent pas aux attentes des biologistes, nous n'avons pas pris en compte, les jeux de paramètres résultant dans des courbes de modes de division avec au moins une des caractéristiques suivantes :

— des valeurs négatives possibles pour les probabilités de division PN,

— la courbe PN n'est pas croissante puis décroissante sur la durée de la simulation.

A partir de cet espace de paramètres d'entrées, nous pouvons générer un sous-ensemble de jeux de paramètres afin d'explorer une partie de l'ensemble des simulations que permettent d'obtenir cet espace d'entrée. Pour visualiser et comparer nos simulations aux données expérimentales, nous avons générés et stockés les données des simulations. Pour chaque jeu de paramètres, nous avons lancés dix réplicas et avons enregistrés pour chacun les nombres de progéniteurs et de neurones toutes les demi-heures. Au total, après filtrage, notre exploration prend en compte plus de cent mille jeux de paramètres.

Comparaison aux données expérimentales

Nous avons confronté les données issues de nos simulations à deux jeux de données expérimentales obtenues par les biologistes. Le premier est celui correspondant aux dynamiques de population de progéniteurs et de neurones (voir Figure 6.21) et le second correspond aux données de modes de division présenté précédemment (voir Figure 6.18).

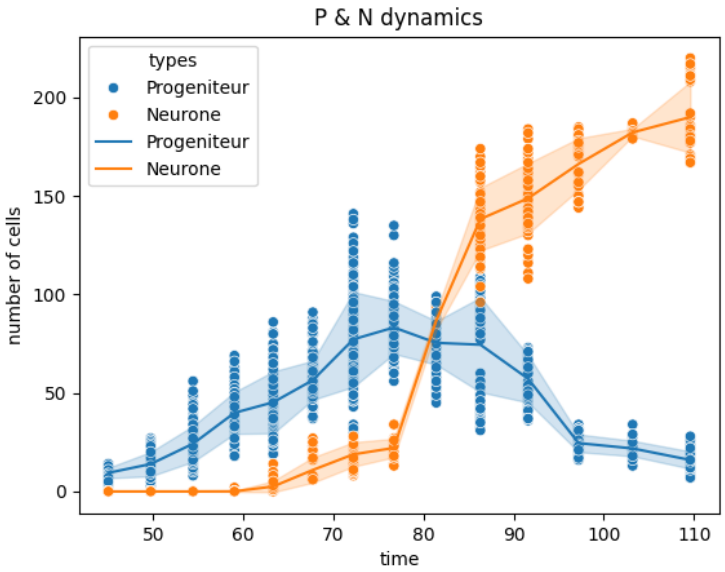


FIGURE 6.21 – **Dynamiques de population expérimentales.** Les courbes ont été générées dans l'outil de notebooks d'ISiCell Workbench. La courbe relie les moyennes des points expérimentaux et le ruban autour correspond à l'écart-type.

Dans les deux cas, nous avons utilisé l'erreur quadratique moyenne pour mesurer la distance de nos simulations aux données expérimentales. Plus pré-

cisément, dans le cas des données de populations, nous avons calculé l'erreur avec l'ensemble des points expérimentaux. Dans celui des données de modes de division, nous avons pris en compte la distance à l'intervalle de confiance pour calculer l'erreur. Ainsi si une courbe de mode de division simulée passe dans l'intervalle de confiance du point expérimental, l'erreur sera considérée comme nulle. De plus, nous avons agrégé les erreur pour les trois modes de division en une seule erreur moyenne.

Nous avons ainsi évaluer nos simulations selon quatre critères : l'erreur en nombre de progéniteurs, l'erreur en nombre de neurones, l'erreur moyenne sur les trois modes de division et la moyenne de source externe entre les stades 12 et 17 inclus. Ce dernier critère d'évaluation traduit la volonté d'éviter de faire reposer les dynamiques de population trop fortement sur l'apport de la source externe. En effet, cela permet aussi de garder des valeurs pour la source externe plus proche de celle de l'article de Saad et al.

Enfin, afin de limiter le nombre de simulations à étudier, nous avons étudié uniquement les simulations dont l'erreur en nombre de progéniteurs et de neurones est inférieure à 25. Cette valeur a été choisie empiriquement pour favoriser les simulations les plus proches des dynamiques de populations. En effet, il s'agit des deux métriques dont nous souhaitons prioriser la minimisation parmi les quatre par rapport aux attentes des biologistes vis-à-vis du modèle.

Avec ces restrictions, le front de Pareto de nos meilleures simulations comptent généralement un peu plus d'une centaine de jeux de paramètres. C'est sur ce front que portent nos analyses.

Résultats d'analyse

La première étape de notre analyse consiste à tracer les boîtes à moustaches des différents paramètres des jeux de notre front de Pareto pour mieux connaître la distribution des paramètres de nos meilleures simulations. Cela permet, en effet, de mettre en évidence des tendances qui amènent aux meilleures solutions. Dans la Figure 6.22, on peut relever :

- Les valeurs trop faibles de σ ne sont pas retenues, cela peut être dû au filtrage en amont ;
- Les valeurs de τ_{PP} et τ_{NN} prennent des ensembles de valeurs plus restreints que l'espace d'entrées, cela peut aussi résulter du filtrage en amont ;
- Les valeurs de β restent faibles (ce qui est attendu par la métrique de minimisation de la source externe) et semblent augmenter en moyenne avec la tardiveté du stade.

En regardant, plus spécifiquement les valeurs de τ_{PP} et τ_{NN} et des β (voir Figure 6.23), on remarque que :

- Les valeurs de τ_{PP} se concentrent entre 65 et 70h, ce qui correspond aux données d'Azaïs et al. avec $\tau_{PP} = 67h$;
- Les valeurs de τ_{NN} se concentrent entre 77 et 83h, ce qui correspond aux données d'Azaïs et al. avec $\tau_{NN} = 79,3h$;
- Les valeurs des β se concentrent aux alentours de 1 progéniteur par heure, ce qui est un peu plus que les données de l'article de Saad et al. avec

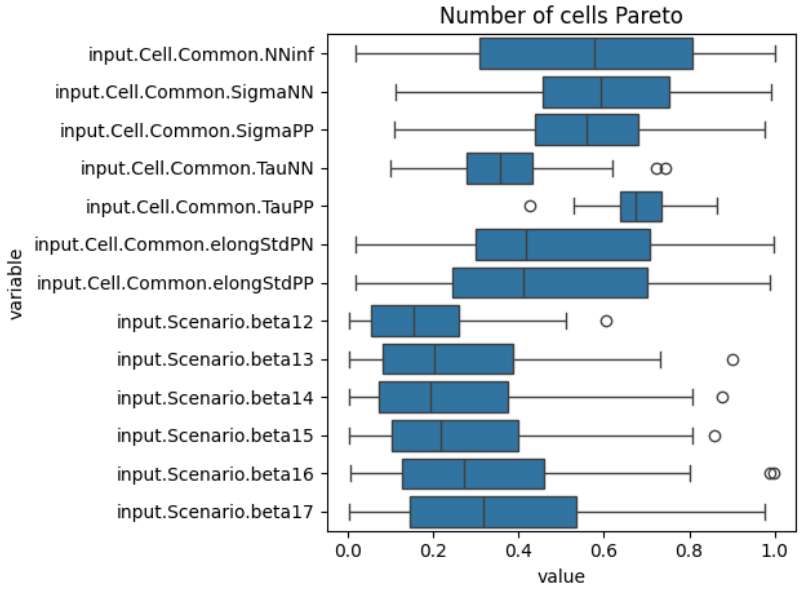


FIGURE 6.22 – **Boîtes à moustaches des différents paramètres d'entrée.** Pour chacun des paramètres, les valeurs ont été normalisées entre les plages de valeurs possibles de l'échantillonnage en hypercube latin.

$$\beta = 0,9h^{-1}$$

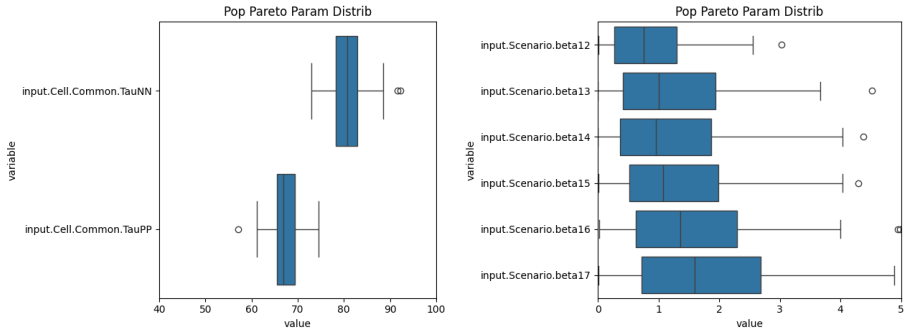


FIGURE 6.23 – **Boîtes à moustaches des valeurs des τ et β.** A gauche les boîtes à moustaches des valeurs prises par les deux paramètres τ des modes de division en heure. A droite, les boîtes à moustaches des valeurs des paramètres β en progéniteurs par heure.

La suite de notre analyse porte sur les corrélations entre les différents pa-

ramètres, les erreurs et mesures pour détecter des dynamiques corrélées. Cela permet de mettre en évidence certains mécanismes du modèle et des effets combinés entre certaines variables. La matrice de corrélation de la Figure 6.24 permet de distinguer deux corrélations à plus de 0,75 en valeur absolue : la première de 0,93 entre l'erreur sur les modes de division moyen et l'erreur sur la courbe de probabilité de division PN et la seconde de -0.88 entre le coefficient de ressemblance GAN et σ_{PP}.

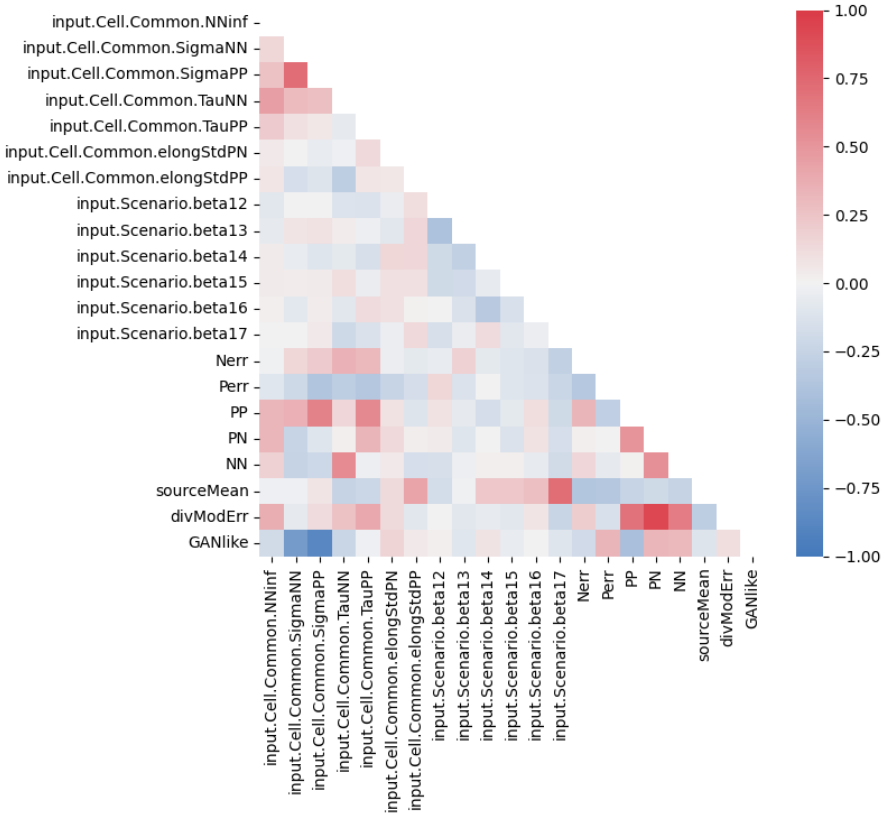


FIGURE 6.24 – **Matrices de corrélation des paramètres, mesures et erreur des jeux de paramètres du front de Pareto.**

La Figure 6.25 place les différents jeux de paramètres selon les deux corrélations précédemment mentionnées. Le paramètre σ_{PP} peut être compris comme la durée de passage d'un régime majoritairement PP à un régime majoritairement PN. Ainsi plus σ_{PP} est grand et plus la transition est longue et donc la probabilité de faire des divisions PP après une PN grande. Inversement, une valeur de σ_{PP} faible plus la transition sera courte, limitant la possibilité pour une cellule isue d'une division PN d'effectuer une division PP un cycle cellulaire

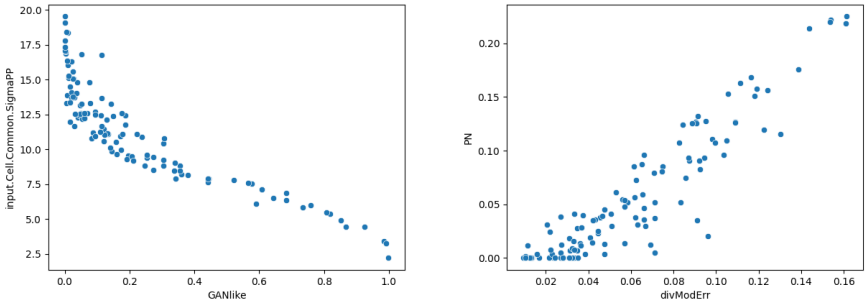


FIGURE 6.25 – **Graphes des jeux de paramètres du front de Pareto selon les plus grandes corrélations.**
A gauche, la valeur de σ_{PP} des différents jeux du front de Pareto en fonction de la valeur de GANlike. A droite, l'erreur moyenne sur les modes de division des différents jeux du front de Pareto en fonction de leur erreur sur la courbe de probabilité PN.

plus tard. Pour la seconde corrélation, cela indique une absence de compensation de l'erreur moyenne sur les modes de division quand les courbes s'éloignent des point expérimentaux pour les probabilités de division PN.

Pour vérifier avec les biologistes que les dynamiques de mode de division et de prolifération correspondent à leurs attentes, nous traçons les courbes correspondantes à nos jeux de paramètre de notre front de Pareto. La Figure 6.26 est un exemple de courbes que l'on obtient en utilisant le jeu de paramètre de notre front de Pareto qui minimise le plus l'erreur sur les modes de division. Le modèle est capable de reproduire ces deux dynamiques avec plusieurs jeux de paramètres différents.

L'ensemble des jeux de paramètres du front de Pareto, ne permet pas de mettre en évidence des paramètres spécifiquement satisfaisant pour les biologistes. Nous avons donc voulu rassembler les dynamiques similaires selon les modes de division et les sources externes pour essayer de discerner des sous-ensembles de paramètres amenant à des dynamiques spécifiques. L'objectif était ensuite de sélectionner qualitativement ces sous-ensemble avec les biologistes par rapport à leurs connaissances du système biologique. Pour partitionner nos jeux de paramètres, nous avons utilisé la méthode des k-moyennes couplée à une méthode du coude [Bholowalia and Kumar, 2014, Cui et al., 2020, Humaira and Rasyidah, 2020] et pour visualiser la distribution de nos partitions, nous avons utilisé une projection UMAP (Uniform Manifold Approximation and Projection ou Approximation et Projection uniformes multi-dimensionnelles en français) [McInnes et al., 2018, Becht et al., 2019, Stolarek et al., 2022] qui permet de projeter en deux dimensions l'ensemble de nos points avec leur partition. La Figure 6.27 montre le résultat de la méthode du coude et la visualisation de

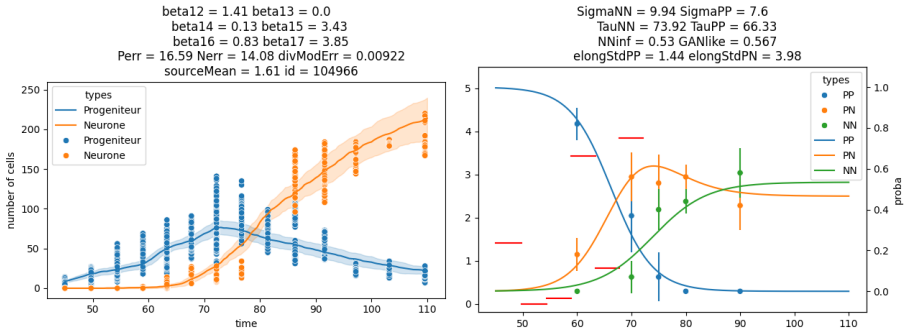


FIGURE 6.26 – **Graphes de prolifération et de probabilités de mode de division d'un jeu de paramètre du front de Pareto.**
A gauche, la courbe de prolifération des progéniteurs et des neurones par rapport aux données expérimentales, le ruban correspond à l'écart-type. A droite, la courbe des probabilités des modes de division par rapport aux données expérimentales, les barres correspondent à l'intervalle de confiance. Sur ce même graphique, on trouve en rouge les valeurs de source externe en progéniteurs par heure. Au dessus de chaque graphe, on retrouve l'ensemble des paramètres, mesures et erreurs pour ce jeu de paramètres.

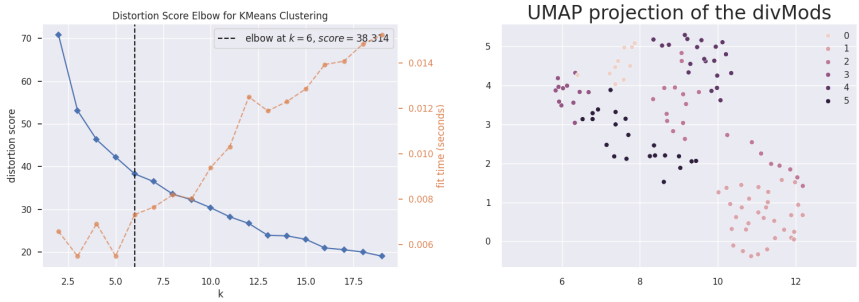


FIGURE 6.27 – **Graphes de la méthode du coude et de projection UMAP du partitionnement des jeux de paramètres selon leurs dynamiques de modes de divisions.**
A gauche, les courbes du score de distorsion et du temps d'ajustement en fonction du nombre de partitionnements. La barre verticale correspond au nombre de clusters optimal selon la méthode du coude. A droite, la projection UMAP des différents jeux de paramètres du front de Pareto colorés selon la partition à laquelle ils ont été affectés.

ces partitions avec la projection UMAP. La projection semble indiquer que les différentes partitions ne se distinguent pas fortement les unes des autres. En traçant les courbes correspondant aux six partitions proposées par la méthode du coude, on observe une grande hétérogénéité au sein de chaque chacune, ce qui laisse entendre que l'ensemble de nos jeux de paramètres diffèrent grandement les uns des autres. Les conclusions sont similaires lorsque l'on essaie de partitionner nos jeux de paramètres selon les valeurs de leur sources externes entre les stade 12 et 17 inclus. Cela est probablement dû au fait que ces jeux sont issus d'un échantillonnage en hypercube latin.

Le partitionnement de nos simulations ne semblent pas permettre de conclure quant à l'existence de sous-ensembles de jeux de paramètres qualitativement plus intéressants du point de vue des biologistes. La réintroduction des données de profils de descendances clonales nous semblent à présent pertinentes à réintroduire pour discriminer parmi tous les différents jeux de paramètres permettant de minimiser l'erreur sur les dynamiques de prolifération et de modes de division.

Nous avons appliqué les mêmes analyses en nous intéressant uniquement aux jeux de paramètres dont la valeur de GANlike est de 100% pour observer les potentielles différences que cela pouvait engendrer. Avec les mêmes critères que pour l'analyse précédente, le nombre de jeux de paramètres du front de Pareto n'est plus que d'une trentaine, ce qui limite la pertinence des résultats de l'analyse étant donné le nombre de dimensions de notre espace d'entrées. Nous avons pu cependant confirmer un résultat déjà attendu, l'ensemble des jeux du front de Pareto des simulations GAN-compatibles présentent une valeur de σ_{PP} faible comme le montre la Figure 6.28. De plus, on observe que les valeurs de σ_{NN} sont également faibles. Cela permet probablement d'obtenir une augmentation rapide du nombre de neurones que le régime PN ne permet pas.

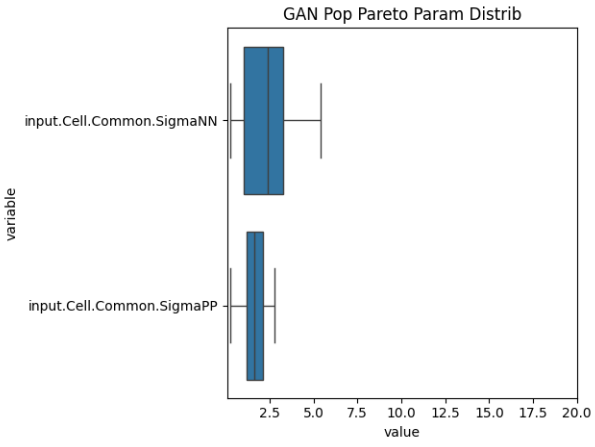


FIGURE 6.28 – **Boîtes à moustaches des valeurs de σ pour les jeux de paramètres GAN-compatibles.**

6.2.2.5 Apport du modèle à la méthodologie

Ce modèle a permis de mettre en place notre méthodologie dans un nouveau contexte biologique et sur un terme plus long que le modèle de sphéroïde. Nous avons pu, ainsi, confirmer l'attrait des biologistes pour notre approche et itérer notre méthodologie avec plusieurs nouvelles hypothèses pour enrichir ou simplifier le modèle au fur et à mesure des itérations.

Nous avons également pu remarquer l'intérêt de la multiplication des expériences de modélisation dans différents contextes biologiques car cela nous a permis, dans le cas de ce modèle, de réemployer un diagramme états-transition déjà utilisé pour un autre modèle. En l'occurrence il s'agissait du diagramme du modèle de sphéroïde dont les états correspondent aux phases du cycle cellulaire.

Notre méthodologie s'est montrée particulièrement efficace pour implémenter et tester différentes hypothèses rapidement. Cela nous a permis d'enrichir ou de simplifier le modèle au fur et à mesure des hypothèses conservées ou rejetées. De plus, la collaboration est, à ce jour, toujours d'actualité, ce qui témoigne de la capacité de notre méthodologie à maximiser l'implication des biologistes et des modélisateurs dans le développement du modèle.

La durée de ce projet a permis d'éprouver le caractère itératif de notre méthodologie. En effet, comme mentionné précédemment, nous avons pu implémenter et tester plusieurs hypothèses. Chacune d'entre elles a correspondu à une nouvelle itération de la méthodologie pour modifier le modèle pour correspondre à l'hypothèse que l'on voulait tester.

6.2.2.6 Apport du modèle à la plateforme

Ce modèle a permis d'utiliser ISiCell dans un nouveau contexte biologique et sur un plus long terme que pour les modèles précédemment présentés. De plus, nous avons pu utilisé l'ensemble des outils de la plateforme avec les biologistes et constaté que ceux-ci favorisent et facilitent l'interaction et le dialogue. Côté modélisateurs, nous avons également pu constater un gain de temps dans le développement du modèle.

Les itérations successives sur le modèle pour implémenter les différentes hypothèses ont permis d'assurer que les modifications du modèles peuvent être facilement implémentées. Notamment l'outil de recherche permet de rapidement retrouver et accéder aux parties du modèle que l'on veut modifier.

ISiCell Explorer s'est révélé particulièrement utile pour rapidement tester de nouvelles hypothèses, notamment sur les sources externes, en visualisant leur impact sur les dynamiques du modèle. De la même manière, nous avons pu exploiter l'ensemble des outils d'ISiCell Workbench pour explorer et analyser le modèle. L'outil de notebook, nous a particulièrement permis de facilement mettre en place un flux d'analyses avec ses différents graphiques qui nous ont permis de faciliter la présentation des résultats aux biologistes entre chaque réunion.

En plus d'avoir enrichi la plateforme d'un nouveau modèle de lignée cellulaire, cette expérience nous a permis de tester et d'améliorer les différents outils

de la plateforme dans un contexte de collaboration. Le modèle n'ayant pas encore donné lieu à une publication, il n'est actuellement pas encore disponible en ligne mais pourra devenir, à terme, un autre modèle de démonstration de la plateforme.

6.3 Retours d'expériences

Dans ce chapitre, nous avons présentés cinq modèles développés selon notre méthodologie à l'aide de notre plateforme ISiCell. Dans cette section, nous allons faire la synthèse des résultats mis en évidence par nos retours d'expériences sur le développement de ces différents modèles vis-à-vis, dans un premier temps, de notre méthodologie et, dans un second temps, de notre plateforme.

6.3.1 Intérêt de notre méthodologie

Au cours du développement de ces cinq modèles, nous avons pu confronter et appliquer notre méthodologie à différents contextes biologiques que ce soit avec ou sans biologiste. Cela nous a permis de constater sa capacité à répondre aux problématiques pour lesquelles nous l'avons pensée mais nous avons également pu découvrir d'autres intérêts auxquels nous n'avions pas pensé en amont de sa conception.

La reproduction de modèles issus de la littérature nous a permis de mettre en avant un nouvel intérêt de notre méthodologie résultant de l'utilisation de diagrammes UML pour représenter nos modèles. En effet, en plus de faciliter la compréhension du modèle par les biologistes et d'accélérer le développement du modèle, nous avons constaté que ces diagrammes répondaient également à un besoin de réplicabilité des modèles de la littérature. Ainsi, les diagrammes produits par notre méthodologie peuvent aisément être ajoutés aux articles pour servir de support à l'explication du modèle mais également pour permettre à d'autres modélisateurs de reproduire le modèle dans le langage de son choix.

Quant aux modèles issus de collaborations, nous avons pu constater l'efficacité de notre méthodologie pour impliquer les biologistes dans le développement et faciliter l'intercompréhension et le dialogue. Une meilleure compréhension du fonctionnement du modèle par les biologistes leur a également permis de plus facilement corriger des incompréhensions et de plus aisément proposer des modifications au modèle. La durée de notre collaboration sur le modèle de neurogenèse nous rassure sur la capacité de notre méthodologie à maintenir l'intérêt des biologistes et des modélisateurs pour le modèle.

Chacun des modèles développés a été l'occasion d'ajuster et de raffiner notre méthodologie en fonction des contextes auxquels nous avons été confrontés. Cela nous a permis de confirmer l'intérêt de notre méthodologie vis-à-vis des objectifs qui lui ont été donné et de découvrir une nouvelle utilité de celle-ci dans la facilitation de la reproductibilité des modèles publiés.

6.3.2 Intérêt d'ISiCell

De la même manière que pour notre méthodologie, nous avons pu, avec ces cinq modèles, utiliser notre plateforme dans différents contextes biologiques. Ceci nous a permis de tester notre plateforme dans des conditions d'utilisation réelle et de l'améliorer au fur et à mesure de son utilisation.

Nous avons pu tester l'ensemble des modules auparavant développés mais également en développer de nouveaux. Notre plateforme a ainsi pu s'enrichir de nouveaux modules qui pourront être utilisés dans de nouveaux contextes pour de nouveaux modèles. De plus, chacun des modèles, à l'exception du modèle de neurogenèse, constituent, à présent, des modèles démonstratifs de notre plateforme accessibles sur l'instance en ligne [1].

En interagissant avec des biologistes, nous avons constaté que l'interface d'ISiCell Builder facilitait les interactions grâce à l'utilisation des diagrammes UML pour représenter le comportement de nos agents cellulaires. En effet, l'utilisation des diagrammes limitent la quantité de code à écrire pour les différentes briques comportementales ce qui permet de ne pas perdre les biologistes lors de ces courtes périodes et leur permet également de mieux comprendre les quelques lignes de code auxquels ils seront confrontés. De plus, la possibilité de compiler à la volée le modèle contribue grandement à accélérer le développement du modèle. Cet outil confirme donc encore l'intérêt de notre méthodologie et d'autant plus lorsqu'elle est couplée à notre plateforme.

L'utilisation d'ISiCell Viewer nous a permis, pour les modèles issus de la littérature, de vérifier que les modèles avaient des comportements cohérents avec ce que les articles décrivaient. De plus, dans le cadre des collaborations avec les biologistes, cela a permis aux biologistes de facilement identifier les comportements incohérents du modèle qui pourraient échapper aux modélisateurs. L'outil semble donc pertinent pour valider qualitativement le bon fonctionnement du modèle.

ISiCell Explorer et ISiCell Workbench se sont avérés des outils très puissants pour étudier les dynamiques de notre modèle. D'un côté ISiCell Explorer permet de disposer des mêmes avantages qu'ISiCell Viewer pour valider qualitativement le fonctionnement du modèle par rapport à ses dynamiques tout en impliquant le biologiste dans le processus. De l'autre côté, ISiCell Workbench a constitué un outil polyvalent pour interagir avec nos simulations avec l'ensemble des librairies disponibles en Python pour analyser, explorer ou optimiser nos modèles. Ces deux outils nous ont permis d'approfondir les connaissances de nos modèles et ont pu servir de base pour recommencer des itérations de notre méthodologie.

Les modèles présentés dans ce chapitre ont permis de tester et d'améliorer et d'améliorer ISiCell au fur et à mesure de leur développement. Nous avons pu ainsi confronter l'utilisation des différents outils de la plateforme à des conditions d'utilisation réelles, de les affiner en fonction des contextes et de confirmer leur intérêt pour appuyer notre méthodologie et ses objectifs.

1. URL de notre instance d'ISiCell : https://isicell.irit.fr

Chapitre 7

Conclusion et perspectives

Dans ce dernier chapitre, nous allons tirer les conclusions des différents aspects des travaux menés dans le cadre de cette thèse, détailler les perspectives envisagées et envisageables sur le court et moyen terme pour poursuivre les travaux précédemment présenter, pour enfin dresser un bilan plus général de ce manuscrit dans son contexte scientifique.

7.1 Conclusion

Au début de ce manuscrit, nous avons décrit une méthodologie participative pour développer des MMA basés cellule dans le cadre de collaborations entre biologistes et modélisateurs. Cette méthodologie a pour but de faciliter le dialogue entre participants appartenant à des domaines d'expertise différents, d'améliorer l'intercompréhension et de minimiser les frustrations lors du développement de modèles. Cette méthodologie repose sur l'usage de diagrammes et la décomposition atomique du code afin de rendre aussi accessible que possible la compréhension du fonctionnement du modèle aux participants n'ayant pas ou peu d'expérience de modélisation informatique et d'inciter ces participants à être plus pro-actifs dans le développement.

Pour appuyer cette méthodologie, nous avons développé, en parallèle, une plateforme de développement facilitant la mise en place de notre méthodologie et accélérant le développement du prototype par le biais de l'automatisation de la génération du code et l'utilisation d'outils encourageant l'interaction et facilitant la compréhension du modèle. Cette plateforme dispose de trois outils principaux répondant à des objectifs différents. Ainsi ISiCell Builder, l'interface principale, vise à accélérer le développement du modèle et à faciliter le dialogue et la compréhension entre biologistes et modélisateurs. De son côté, ISiCell Viewer permet aux biologistes d'apporter une approbation qualitative du comportement du modèle et de vérifier que son comportement est celui attendu dans un cadre donné. Enfin, les outils d'exploration et d'analyse intégrés constituent un moyen pour les modélisateurs de facilement étudier, explorer et

analyser le modèle.

Le fonctionnement interne de cette plateforme est le résultat d'une réflexion plus générale sur la nature de nos modèles et de leur architecture. Ainsi par un processus de généralisation de cette dernière, nous avons pu assez facilement dégager un métamodèle au sein duquel la génération et l'injection de code pour générer nos MMAs pouvaient être standardisées et automatisées. Conséquemment, cette standardisation de la structure de nos modèles a permis la généralisation de la communication entre nos simulations et la base de données utilisée par le Viewer. De manière similaire, nous avons pu automatisé l'interfaçage du code C++ des simulations avec du Python pour les outils d'analyse et d'exploration.

Enfin, nous avons confronté et éprouvé notre plateforme et la méthodologie qui la sous-tend au développement concrets de modèles basés cellules. Ainsi, nous avons démontré la capacité de la plateforme à reproduire des modèles issus de la littérature et à fournir des schémas plus explicites du fonctionnement interne des modèles facilitant, de la sorte, la reproductibilité. Au travers de collaboration avec des biologistes, nous avons pu confirmer l'intérêt de notre méthodologie pour impliquer les biologistes dans le développement de modèle et faciliter le dialogue et l'intercompréhension entre les différents participants.

L'ensemble de ce manuscrit tend à démontrer l'efficacité de notre méthodologie couplée à la plateforme ISiCell pour développer des MMA basés cellules. En effet, notre architecture standardisée permet la génération et l'injection de code, ce qui accélère le développement. Et de l'autre côté, notre méthodologie basée sur l'usage de diagrammes UML facilite l'intercompréhension et le dialogue, ce qui minimise la frustration des modélisateurs et des biologistes au cours de la collaboration. Enfin, cette méthodologie semble également un bon moyen pour favoriser la réplicabilité des modèles par d'autres modélisateurs et encourage à employer des architectures similaires à celle de notre métamodèle.

7.2 Perspectives

Dans cette section, nous allons aborder des perspectives sur les court et moyen termes de poursuite de ces travaux. Nous allons aborder, dans un premier temps, des perspectives pour faciliter l'accès à la plateforme pour des biologistes qui n'ont pas ou peu de compétences en modélisation informatique. Nous poursuivrons en abordant des perspectives pour agrandir la communauté d'utilisateurs d'ISiCell. Enfin, nous traiterons la constitution de protocoles pour évaluer l'efficacité de notre méthodologie auprès des biologistes qui ont collaboré au développement de modèles.

7.2.1 Accessibilité de la plateforme

ISiCell, dans sa forme actuelle, nécessite un minimum de connaissances et compétences en C++ et en Python pour être utilisé. Ainsi un biologiste avec peu ou pas de compétences en modélisation informatique aura beaucoup de

difficultés à développer, modifier et analyser un modèle sans l'aide d'un modélisateur plus expérimenté. Ainsi, nous souhaiterions explorer quelques pistes pour faciliter l'accès de la plateforme en abaissant le minimum de compétences nécessaires à son emploi.

Une des premières idées qui nous paraît envisageable est le développement d'un métalangage à l'image de la plateforme gamma et de son GAML [Taillandier et al., 2019], qui pourrait simplifier le codage pour les non-informaticiens, rendant la plateforme plus accessible pour les biologistes. En effet, cela pourrait simplifier l'écriture du code des briques de comportement et du protocole en fournissant un langage intermédiaire au C++ avec une syntaxe plus abordable pour des biologistes avec une faible expérience en programmation.

De la même manière, l'encapsulation et la constitution de librairies Python pour faciliter et automatiser l'utilisation d'algorithmes d'exploration, d'analyse et d'optimisation des simulations permettraient à un plus large éventail de biologistes de s'approprier les outils d'exploration de manière autonome. Le fonctionnement pourrait se rapprocher de solutions déjà existantes comme OpenMole [Reuillon et al., 2013] utilisées pour l'analyse de modèles.

Enfin, de manière assez générale, il est envisagé de rédiger un ensemble de tutoriels et d'exemples de code pour faciliter la prise en main par de nouveaux utilisateurs. Cela permettra dans un premier temps de fournir des ébauches de codes prêtes à l'emploi pour commencer à utiliser la plateforme plus facilement.

7.2.2 Fidéliser une communauté

ISiCell ne compte, à ce jour, comme utilisateurs que les personnes ayant participé aux travaux détaillés dans ce manuscrit. La plateforme n'a en effet pas encore fidéliser de communauté en dehors du microcosme de ses premiers utilisateurs. C'est pourquoi l'un de nos objectifs sur le moyen terme est d'agrandir le nombre d'utilisateurs et de cas d'utilisation de la plateforme.

Pour assurer la pérennité d'ISiCell, il nous paraît vital de commencer par augmenter le nombre d'utilisateurs de la plateforme. En effet, la constitution d'une communauté permettrait d'augmenter le nombre de contributeurs au projet qui pourront participer au processus d'amélioration continue de la plateforme au travers du dépôt gitlab [1]. Ainsi, l'augmentation du nombre de contributeurs permettrait d'enrichir la plateforme en nouvelles fonctionnalités et de potentiellement découvrir et corriger des problèmes qui n'avaient pas été identifiés. Cela permettrait aussi d'augmenter le nombre de collaborations et de conséquemment augmenter le nombre de modèles développés avec notre plateforme et sa méthodologie, ce qui améliorera, en retour, la visibilité de la plateforme en la faisant découvrir à de nouveaux utilisateurs dans un cercle vertueux d'amélioration continue et de croissance de la communauté de modélisateurs utilisant ISiCell.

De manière similaire, nous souhaiterions développer un système de gestion des dépôts officiels nommé ISiCell Hub permettant de centraliser l'ensemble des

1. Adresse du repo gitlab d'ISiCell : https://gitlab.com/isicell-irit/isicell

modules validés et des modèles (sous forme de fichiers isidiag) par la communauté afin de faciliter le partage et la réutilisation de modules et de modèles au sein de la communauté d'utilisateurs. Cela permettrait d'ajouter de nouvelles fonctionnalités et mécanismes aux modèles que d'autres utilisateurs ont pu développer pour d'autres modèles, mais aussi de reprendre des modèles déjà éprouvés pour servir de base à de nouveaux modèles ou juste étudier/utiliser ce même modèle. L'objectif d'ISiCell Hub sera de faciliter le partage et la réplicabilité de résultats publiés ainsi que d'enrichir toujours plus la plateforme en briques comportementales et en modules permettant de gérer de nouvelles mécaniques environnementales ou de physique.

Enfin, nous souhaiterions augmenter le nombre de collaboration avec des biologistes pour appliquer notre méthodologie à toujours plus de contextes et ainsi toujours plus enrichir la plateforme en nouveaux modules et modèles. En effet, chaque nouveau contexte biologique est l'occasion de développer de nouveaux modules pour répondre à de nouveaux besoins en briques de modélisation. De plus, augmenter ces collaborations est l'occasion de continuer à raffiner notre méthodologie en recueillant les avis des biologiste. Cela permettra de proposer des ajustements et de nouveaux outils pour améliorer le dialogue et l'intercompréhension entre les modélisateurs et les biologistes.

7.2.3 Evaluer notre méthodologie

Enfin, le dernier axe de perspectives sur les travaux de ce manuscrit est la création de protocoles d'évaluation de notre méthodologie adressés aux modélisateurs et aux biologistes. Ces protocoles s'inscrivent dans notre volonté d'amélioration continue de la plateforme et de notre méthodologie.

Côté modélisateurs, les informations que nous souhaiterions recueillir sont les suivantes :

- l'ergonomie de la plateforme (trouvabilité, efficacité, efficience,etc.),
- la facilité de prise en main,
- la pertinence de la plateforme/méthodologie par rapport à leurs autres expériences de modélisation,
- la frustration ressentie lors de l'utilisation,
- l'évaluation de l'interaction avec les biologistes,
- des pistes/idées d'amélioration.

L'objectif est d'identifier les points forts et points faibles de la plateforme et de la méthodologie qui la sous-tend afin de distinguer des axes d'amélioration de l'expérience des modélisateurs lors de collaborations.

Côté biologistes, nous voudrions recueillir les informations suivantes :

- le niveau de compréhension du fonctionnement du modèle,
- la pertinence de la méthodologie par rapport à leurs autres expériences de modélisation,
- la frustration ressentie lors de la collaboration,
- l'évaluation de l'interaction avec les modélisateurs,
- des pistes/idées d'amélioration.

Nous visons, ainsi, à mettre en lumière les aspects de notre méthodologie qui peuvent être améliorer dans l'objectif d'augmenter toujours plus l'interaction entre biologistes et modélisateurs et faciliter le dialogue et l'intercompréhension entre eux.

Ces protocoles d'évaluation nécessiteront une réflexion, en amont, sur leur mise en place et sur comment évaluer/quantifier les différents aspects qui nous intéressent. L'objectif de ces protocoles est de permettre de mettre en évidence les axes d'amélioration de notre méthodologie et de sa plateforme à un instant donné. Cela nous permettra également de témoigner de leur évolution au fur et à mesure de la prise en compte des différentes remarques.

7.3 Bilan

Les travaux réalisés dans le cadre de cette thèse ont permis l'émergence d'une méthodologie participative pour la modélisation de MMA basés cellules et d'une plateforme web à son service. Cette combinaison méthodologie et plateforme est favorisée et rendue possible par la standardisation de l'architecture des MMA basés cellules développés. ISiCell a pu être employée pour développer différents modèles de biologie cellulaire démontrant la grande variété d'applications de la plateforme. Nous estimons que cette méthodologie accompagnée de sa plateforme répondent au besoins suivants :

- de facilitation du dialogue entre biologistes et modélisateurs,
- d'intercompréhension entre ces derniers,
- de minimisation de la frustration lors de collaboration,
- de reproductibilité des modèles,
- de standardisation des architectures des MMA basés cellules,
- de centralisation des outils de développements et d'analyse/exploration/optimisation,
- d'accélération du développement de prototypes fonctionnels.

Les perspectives pour la plateforme et la méthodologie qu'elle sert comprennent une amélioration de l'accessibilité d'ISiCell à des biologistes avec peu d'expérience de programmation/modélisation, l'agrandissement de la communauté de modélisateurs et de biologistes utilisant la plateforme et la mise en place de protocoles d'évaluation dans le cadre de l'amélioration continue de notre méthodologie et des outils qui la servent. Ainsi, sur les court et moyen termes, nous prévoyons d'agir sur les différents axes d'amélioration dégagés afin de poursuivre les efforts qui ont mené aux résultats présentés dans ce manuscrit.

La modélisation multi-agents et plus généralement *in silico* permet une grande variété d'applications en biologie. En effet, les possibilités offertes par l'informatique autorisent de conceptualiser un ensemble large de mécanismes et de phénomènes biologiques. Cela permet aux biologistes de disposer d'outils de prédiction, d'infirmer/confirmer des hypothèses et de prendre de la hauteur vis-à-vis de leurs connaissances. Ainsi, de la même manière que l'informatique s'enrichit des algorithmes inspiré du vivant, les biologistes peuvent retirer de nouveaux outils et de nouvelles informations de la modélisation *in silico*. La

collaboration entre modélisateurs et biologistes constituent, à notre sens, une perspective fructueuse de dégager de nouvelles connaissances.

De manière plus générale, ces travaux témoignent de l'intérêt croissant pour l'interdisciplinarité dans le cadre de la recherche scientifique. De plus, la modélisation *in silico* se prête plus particulièrement à l'émergence de collaborations entre modélisateurs et des experts de différents domaines. En effet, l'informatique permet un niveau d'abstraction suffisant à la représentation d'un panel très large de phénomènes et dispose de plusieurs paradigmes qui autorisent d'autant plus d'angles d'approches. Ainsi, nous estimons nous inscrire dans cette dynamique de collaborations pluridisciplinaires et plus particulièrement dans le contexte des méthodologies participatives appliquées à la modélisation du vivant. Notre expérience de l'approche participative peut, à notre avis, constituer une inspiration pour d'autres tentatives de formalisation de modélisation collaborative.

Annexe A

Structure du JSON de génération

```
{
    "moduleSelected": ["moduleNameA", "moduleNameB",
        ...],
    "customModules": {
        "customNameA": {
            "customBodyA.hpp":{
                "type": "Body",
                "code": "#ifndef ..."
            },
            "customPluginA.hpp":{
                "type": "Plugin",
                "code": "#ifndef ..."
            },
            "customFile.hpp":{
                "type": "",
                "code": "#ifndef ..."
            },
            ...
        },
        "customNameB": {
            ...
        },
        ...
    }
    "enums": {
        "enumA": ["valueA", "valueB", ...],
        "enumB": [...],
        ...
```

```
28      }
29      "cellTypes": ["typeNameA", "typeNameB", ...],
30      "scenario": {
31          "attributes": [
32              {"type" :  "int", "value": "varA"},
33              {"type" :  "double", "value": "varB"},
34              ...
35          ],
36          "codeInit": "methodA(params);\nmethodB(params)
                ;\n...",
37          "codeLoop": "methodC(params);\nmethodD(params)
                ;\n...",
38          "functions":[
39              "hashA": "void methodA(params)...",
40              "hashB": "...",
41              ...
42          ]
43      },
44      "cell": {
45          "attributes": [
46              {"type" :  "int", "value": "varA"},
47              {"type" :  "double", "value": "varB"},
48              ...
49          ],
50          "initializationCode": "methodA(params);\
                nmethodB(params);\n...",
51          "initializationTransitions": [
52              "from": "cellInstantiation",
53              "destinations":[
54                  {"to": "stateNameA", "condition": "
                        boolExpression"},
55                  {"to": "stateNameB", "condition":
                        "..."},
56                  ...
57              ]
58          ],
59          "States": [
60              {"name": "stateNameA", "code": "
                    commonBehavior(w);\n..."},
61              {"name": "stateNameB", "code": "
                    commonBehavior(w);\n..."},
62              ...
63          ],
64          "Transitions": [
65              "from": "stateNameA",
66              "destinations": [
```

```
                {"to": "stateNameB", "condition": "
                    boolExpression"},
                {"to": "stateNameC", "condition":
                    "..."},
                ...
            ],
            "from": "stateNameB",
            "destinations": [
                {"to": "stateNameA", "condition": "
                    boolExpression"},
                {"to": "stateNameC", "condition":
                    "..."},
                ...
            ],
            ...
        ],
        "functions": [
            "commonBehavior": "template<typename
                world_t>\n..."
            "hashA": "void methodA(params)...",
            "hashB": "...",
            ...
        ]
    },
    "paramInputs": {
        "Cell": [
            {"type": "int", "name": "varA"},
            {"type": "double", "name": "varB"},
            ...
        ],
        "Scenario": [
            {"type": "int", "name": "varC"},
            {"type": "double", "name": "varD"},
            ...
        ]
    },
    "CellAtributesRecorded": [
        {"type": "int", "name": "varA"},
        {"type": "double", "name": "varB"},
        ...
    ]
}
```

Bibliographie

[Abrahams and Gurtovoy, 2004] Abrahams, D. and Gurtovoy, A. (2004). *C++ template metaprogramming : concepts, tools, and techniques from Boost and beyond*. Pearson Education.

[Adamatzky, 2010] Adamatzky, A. (2010). *Game of life cellular automata*, volume 1. Springer.

[Aguilera-Venegas et al., 2019] Aguilera-Venegas, G., Galán-García, J. L., Egea-Guerrero, R., Galán-García, M. Á., Rodríguez-Cielos, P., Padilla-Domínguez, Y., and Galán-Luque, M. (2019). A probabilistic extension to conway's game of life. *Advances in Computational Mathematics*, 45 :2111–2121.

[Albers and Sonsalla, 1995] Albers, D. S. and Sonsalla, P. K. (1995). Methamphetamine-induced hyperthermia and dopaminergic neurotoxicity in mice : pharmacological profile of protective and nonprotective agents. *Journal of Pharmacology and Experimental Therapeutics*, 275(3) :1104–1114.

[Almet et al., 2020] Almet, A. A., Maini, P. K., Moulton, D. E., and Byrne, H. M. (2020). Modeling perspectives on the intestinal crypt, a canonical system for growth, mechanics, and remodeling. *Current Opinion in Biomedical Engineering*, 15 :32–39.

[Ameden et al., 2009] Ameden, H. A., Boxall, P. C., Cash, S. B., and Vickers, D. A. (2009). An agent-based model of border enforcement for invasive species management. *Canadian Journal of Agricultural Economics/Revue canadienne d'agroeconomie*, 57(4) :481–496.

[Amereh et al., 2021] Amereh, M., Edwards, R., Akbari, M., and Nadler, B. (2021). In-silico modeling of tumor spheroid formation and growth. *Micromachines*, 12(7) :749.

[An et al., 2009] An, G., Mi, Q., Dutta-Moscato, J., and Vodovotz, Y. (2009). Agent-based models in translational systems biology. *Wiley Interdisciplinary Reviews : Systems Biology and Medicine*, 1(2) :159–171.

[An and Faeder, 2009] An, G. C. and Faeder, J. R. (2009). Detailed qualitative dynamic knowledge representation using a bionetgen model of tlr-4 signaling and preconditioning. *Mathematical biosciences*, 217(1) :53–63.

[Anckaert et al., 2007] Anckaert, B., Madou, M., and De Bosschere, K. (2007). A model for self-modifying code. In *Information Hiding : 8th International*

Workshop, IH 2006, Alexandria, VA, USA, July 10-12, 2006. Revised Selcted Papers 8, pages 232–248. Springer.

[Andrieu, 2014] Andrieu, G. (2014). *Rôle de la voie sphingosine kinase/sphingosine 1-phosphate dans le contrôle de la division cellulaire/ Theses. fr*. PhD thesis, Toulouse 3.

[Apeke et al., 2017] Apeke, S., Gaubert, L., Boussion, N., Lambin, P., Visvikis, D., Rodin, V., and Redou, P. (2017). Multi-scale modeling and oxygen impact on tumor temporal evolution : Application on rectal cancer during radiotherapy. *IEEE Transactions on Medical Imaging*, 37(4) :871–880.

[Apeke et al., 2022] Apeke, S., Gaubert, L., Boussion, N., Visvikis, D., Saut, O., Colin, T., Lambin, P., Rodin, V., and Redou, P. (2022). An hybrid model for rectal tumour response prediction during radiotherapy. *Open Journal of Biophysics*, 12(4) :245–264.

[Arató, 2003] Arató, M. (2003). A famous nonlinear stochastic equation (lotka-volterra model with diffusion). *Mathematical and Computer Modelling*, 38(7-9) :709–726.

[Ari and Ustazhanov, 2014] Ari, N. and Ustazhanov, M. (2014). Matplotlib in python. In *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, pages 1–6. IEEE.

[Arora et al., 2017] Arora, V., Bhatia, R., and Singh, M. (2017). Synthesizing test scenarios in uml activity diagram using a bio-inspired approach. *Computer Languages, Systems & Structures*, 50 :1–19.

[Assini et al., 2009] Assini, F. L., Duzzioni, M., and Takahashi, R. N. (2009). Object location memory in mice : pharmacological validation and further evidence of hippocampal ca1 participation. *Behavioural brain research*, 204(1) :206–211.

[Axelrod, 1997] Axelrod, R. (1997). *The Complexity of Cooperation : Agent-Based Models of Competition and Collaboration : Agent-Based Models of Competition and Collaboration*. Princeton university press.

[Axelrod and Hamilton, 1981] Axelrod, R. and Hamilton, W. D. (1981). The evolution of cooperation. *science*, 211(4489) :1390–1396.

[Ayachit, 2015] Ayachit, U. (2015). *The paraview guide : a parallel visualization application*. Kitware, Inc.

[Azaïs et al., 2019] Azaïs, M., Agius, E., Blanco, S., Molina, A., Pituello, F., Tregan, J.-M., Vallet, A., and Gautrais, J. (2019). Timing the spinal cord development with neural progenitor cells losing their proliferative capacity : a theoretical analysis. *Neural development*, 14 :1–19.

[Badino, 2015] Badino, M. (2015). *The bumpy road : Max Planck from radiation theory to the quantum (1896-1906)*. Springer.

[Bahar and Mao, 2004] Bahar, A. and Mao, X. (2004). Stochastic delay lotka–volterra model. *Journal of Mathematical Analysis and Applications*, 292(2) :364–380.

[Bai et al., 2008] Bai, L., Eyiyurekli, M., and Breen, D. E. (2008). Automated shape composition based on cell biology and distributed genetic programming. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1179–1186.

[Bailer-Jones, 2002] Bailer-Jones, D. M. (2002). Scientists' thoughts on scientific models. *Perspectives on science*, 10(3) :275–301.

[Bailer-Jones, 2009] Bailer-Jones, D. M. (2009). *Scientific models in philosophy of science*. University of Pittsburgh Pre.

[Balaskas et al., 2012] Balaskas, N., Ribeiro, A., Panovska, J., Dessaud, E., Sasai, N., Page, K. M., Briscoe, J., and Ribes, V. (2012). Gene regulatory logic for reading the sonic hedgehog signaling gradient in the vertebrate neural tube. *Cell*, 148(1) :273–284.

[Ballesteros Hernando et al., 2019] Ballesteros Hernando, J., Ramos Gómez, M., and Díaz Lantada, A. (2019). Modeling living cells within microfluidic systems using cellular automata models. *Scientific Reports*, 9(1) :14886.

[Ballet et al., 2017] Ballet, P., Rivière, J., Pothet, A., Theron, M., Pichavant, K., Abautret, F., Fronville, A., and Rodin, V. (2017). Modelling and simulating complex systems in biology : Introducing netbiodyn–a pedagogical and intuitive agent-based software. In *Multi-Agent-Based Simulations Applied to Biological and Environmental Systems*, pages 128–158. IGI Global.

[Balter et al., 2007] Balter, A., Merks, R. M., Popławski, N. J., Swat, M., and Glazier, J. A. (2007). The glazier-graner-hogeweg model : extensions, future directions, and opportunities for further study. *Single-Cell-Based Models in Biology and Medicine*, pages 151–167.

[Bao et al., 2011] Bao, J., Mao, X., Yin, G., and Yuan, C. (2011). Competitive lotka–volterra population dynamics with jumps. *Nonlinear Analysis : Theory, Methods & Applications*, 74(17) :6601–6616.

[Barbier et al., 2024] Barbier, J.-M., Tardivo, C., Delmotte, S., Cittadini, R., Hossard, L., and Le Page, C. (2024). How to intensify collaboration in a participatory modelling process to collectively design and evaluate new farming systems. *Italian Journal of Agronomy*, 18(4).

[Barreteau et al., 2003] Barreteau, O., Antona, M., D'Aquino, P., Aubert, S., Boissau, S., Bousquet, F., Daré, W., Etienne, M., Le Page, C., Mathevet, R., et al. (2003). Our companion modelling approach. *Journal of Artificial Societies and Social Simulation*, 6(1).

[Barricelli, 1962] Barricelli, N. A. (1962). Numerical testing of evolution theories : part i theoretical introduction and basic tests. *Acta Biotheoretica*, 16(1-2) :69–98.

[Barricelli et al., 1954] Barricelli, N. A. et al. (1954). Esempi numerici di processi di evoluzione. *Methodos*, 6(21-22) :45–68.

[Basco-Carrera et al., 2017] Basco-Carrera, L., Warren, A., van Beek, E., Jonoski, A., and Giardino, A. (2017). Collaborative modelling or participatory

modelling ? a framework for water resources management. *Environmental Modelling & Software*, 91 :95–110.

[Baserga, 1981] Baserga, R. (1981). The cell cycle. *New England Journal of Medicine*, 304(8) :453–459.

[Bauerstätter, 2022] Bauerstätter, P. (2022). Compartment modeling of overweight in toddler age : Modeling and simulating a diets effect with copasi. *Simul. Notes Eur.*, 32(4) :221–224.

[Baxevanis et al., 2020] Baxevanis, A. D., Bader, G. D., and Wishart, D. S. (2020). *Bioinformatics*. John Wiley & Sons.

[Bays, 2010] Bays, C. (2010). Introduction to cellular automata and conway's game of life. In *Game of Life Cellular Automata*, pages 1–7. Springer.

[Beaver et al., 2015] Beaver, J. A., Amiri-Kordestani, L., Charlab, R., Chen, W., Palmby, T., Tilley, A., Zirkelbach, J. F., Yu, J., Liu, Q., Zhao, L., et al. (2015). Fda approval : palbociclib for the treatment of postmenopausal patients with estrogen receptor–positive, her2-negative metastatic breast cancer. *Clinical Cancer Research*, 21(21) :4760–4766.

[Beazley, 1998] Beazley, D. M. (1998). Interfacing c/c++ and python with swig. In *7th International Python Conference, SWIG Tutorial*.

[Beazley et al., 1996] Beazley, D. M. et al. (1996). Swig : An easy to use tool for integrating scripting languages with c and c++. In *Tcl/Tk Workshop*, volume 43, page 74.

[Becht et al., 2019] Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I. W., Ng, L. G., Ginhoux, F., and Newell, E. W. (2019). Dimensionality reduction for visualizing single-cell data using umap. *Nature biotechnology*, 37(1) :38–44.

[Becker et al., 2009] Becker, S., MacQueen, G., and Wojtowicz, J. M. (2009). Computational modeling and empirical studies of hippocampal neurogenesis-dependent memory : Effects of interference, stress and depression. *Brain research*, 1299 :45–54.

[Beckert and Masquida, 2011] Beckert, B. and Masquida, B. (2011). Synthesis of rna by in vitro transcription. *RNA : Methods and protocols*, pages 29–41.

[Benov, 2016] Benov, D. M. (2016). The manhattan project, the first electronic computer and the monte carlo method. *Monte Carlo Methods and Applications*, 22(1) :73–79.

[Benzekry et al., 2012] Benzekry, S., Chapuisat, G., Ciccolini, J., Erlinger, A., and Hubert, F. (2012). A new mathematical model for optimizing the combination between antiangiogenic and cytotoxic drugs in oncology. *Comptes Rendus Mathematique*, 350(1-2) :23–28.

[Bernard, 2020] Bernard, D. (2020). *De la cellule au sphéroïde : modélisation et simulation-agents de la prolifération cellulaire*. PhD thesis, Université Toulouse I - Capitole.

[Bernard et al., 2019] Bernard, D., Mondesert, O., Gomes, A., Duthen, Y., Lobjois, V., Cussat-Blanc, S., and Ducommun, B. (2019). A checkpoint-oriented cell cycle simulation model. *Cell Cycle*, 18(8) :795–808.

[Berryman, 2004] Berryman, S. (2004). Democritus.

[Bholowalia and Kumar, 2014] Bholowalia, P. and Kumar, A. (2014). Ebk-means : A clustering technique based on elbow method and k-means in wsn. *International Journal of Computer Applications*, 105(9).

[Blackwell, 1966] Blackwell, R. J. (1966). Descartes' laws of motion. *Isis*, 57(2) :220–234.

[Blikstein and Wilensky, 2010] Blikstein, P. and Wilensky, U. (2010). Materialsim : A constructionist agent-based modeling approach to engineering education. *Designs for learning environments of the future : International perspectives from the learning sciences*, pages 17–60.

[Bodine et al., 2020] Bodine, E. N., Panoff, R. M., Voit, E. O., and Weisstein, A. E. (2020). Agent-based modeling and simulation in mathematics and biology education. *Bulletin of Mathematical Biology*, 82 :1–19.

[Bonfanti and Peretto, 2011] Bonfanti, L. and Peretto, P. (2011). Adult neurogenesis in mammals–a theme with many variations. *European Journal of Neuroscience*, 34(6) :930–950.

[Bostock et al., 2011] Bostock, M., Ogievetsky, V., and Heer, J. (2011). D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12) :2301–2309.

[Bouligand, 1986] Bouligand, Y. (1986). Fibroblasts, morphogenesis and cellular automata. In *Disordered Systems and Biological Organization*, pages 367–379. Springer.

[Bousquet and Le Page, 2004] Bousquet, F. and Le Page, C. (2004). Multi-agent simulations and ecosystem management : a review. *Ecological modelling*, 176(3-4) :313–332.

[Box, 1976] Box, G. E. (1976). Science and statistics. *Journal of the American Statistical Association*, 71(356) :791–799.

[Box, 1979] Box, G. E. (1979). Robustness in the strategy of scientific model building. In *Robustness in statistics*, pages 201–236. Elsevier.

[Brady et al., 2020] Brady, K., Moon, S., Nguyen, T., and Coffman, J. (2020). Docker container security in cloud computing. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0975–0980. IEEE.

[Bravo and Axelrod, 2013] Bravo, R. and Axelrod, D. E. (2013). A calibrated agent-based computer model of stochastic cell dynamics in normal human colon crypts useful for in silico experiments. *Theoretical Biology and Medical Modelling*, 10 :1–24.

[Breitling et al., 2008] Breitling, R., Gilbert, D., Heiner, M., and Orton, R. (2008). A structured approach for the engineering of biochemical network

models, illustrated for signalling pathways. *Briefings in bioinformatics*, 9(5) :404–421.

[Brigandt and Love, 2008] Brigandt, I. and Love, A. (2008). Reductionism in biology.

[Browning et al., 2020] Browning, A. P., Warne, D. J., Burrage, K., Baker, R. E., and Simpson, M. J. (2020). Identifiability analysis for stochastic differential equation models in systems biology. *Journal of the Royal Society Interface*, 17(173) :20200652.

[Bruggeman and Westerhoff, 2007] Bruggeman, F. J. and Westerhoff, H. V. (2007). The nature of systems biology. *TRENDS in Microbiology*, 15(1) :45–50.

[Bull et al., 2020] Bull, J. A., Mech, F., Quaiser, T., Waters, S. L., and Byrne, H. M. (2020). Mathematical modelling reveals cellular dynamics within tumour spheroids. *PLoS computational biology*, 16(8) :e1007961.

[Buske et al., 2011] Buske, P., Galle, J., Barker, N., Aust, G., Clevers, H., and Loeffler, M. (2011). A comprehensive model of the spatio-temporal stem cell and tissue organisation in the intestinal crypt. *PLoS computational biology*, 7(1) :e1001045.

[Butner et al., 2016] Butner, J. D., Chuang, Y.-L., Simbawa, E., Al-Fhaid, A., Mahmoud, S., Cristini, V., and Wang, Z. (2016). A hybrid agent-based model of the developing mammary terminal end bud. *Journal of theoretical biology*, 407 :259–270.

[Cadoo et al., 2014] Cadoo, K. A., Gucalp, A., and Traina, T. A. (2014). Palbociclib : an evidence-based review of its potential in the treatment of breast cancer. *Breast Cancer : Targets and Therapy*, pages 123–133.

[Campos-Ortega, 1993] Campos-Ortega, J. A. (1993). Mechanisms of early neurogenesis in drosophila melanogaster. *Journal of neurobiology*, 24(10) :1305–1327.

[Carbo et al., 2013] Carbo, A., Bassaganya-Riera, J., Pedragosa, M., Viladomiu, M., Marathe, M., Eubank, S., Wendelsdorf, K., Bisset, K., Hoops, S., Deng, X., et al. (2013). Predictive computational modeling of the mucosal immune responses during helicobacter pylori infection. *PloS one*, 8(9) :e73365.

[Carr, 1997] Carr, M. (1997). Visualization with opengl : 3d made easy. *IEEE Antennas and Propagation Magazine*, 39(4) :116–120.

[Castelletti and Soncini-Sessa, 2007] Castelletti, A. and Soncini-Sessa, R. (2007). Bayesian networks and participatory modelling in water resource management. *Environmental Modelling & Software*, 22(8) :1075–1088.

[Caudill and Lynch, 2018] Caudill, L. and Lynch, F. (2018). A mathematical model of the inflammatory response to pathogen challenge. *Bulletin of Mathematical Biology*, 80 :2242–2271.

[Cerenius et al., 2010] Cerenius, L., Babu, R., Söderhäll, K., and Jiravanichpaisal, P. (2010). In vitro effects on bacterial growth of phenoloxidase reaction products. *Journal of invertebrate pathology*, 103(1) :21–23.

[Chan, 2018] Chan, B. W.-C. (2018). Lenia-biology of artificial life. *arXiv preprint arXiv :1812.05433*.

[Chan, 2020] Chan, B. W.-C. (2020). Lenia and expanded universe. In *Artificial Life Conference Proceedings 32*, pages 221–229. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . ..

[Che et al., 2011] Che, D., Liu, Q., Rasheed, K., and Tao, X. (2011). Decision tree and ensemble learning algorithms with their applications in bioinformatics. *Software tools and algorithms for biological systems*, pages 191–199.

[Chiacchio et al., 2014] Chiacchio, F., Pennisi, M., Russo, G., Motta, S., and Pappalardo, F. (2014). Agent-based modeling of the immune system : Netlogo, a promising framework. *BioMed research international*, 2014(1) :907171.

[Chirife et al., 1983] Chirife, J., Herszage, L., Joseph, A., and Kohn, E. S. (1983). In vitro study of bacterial growth inhibition in concentrated sugar solutions : microbiological basis for the use of sugar in treating infected wounds. *Antimicrobial Agents and Chemotherapy*, 23(5) :766–773.

[Choi et al., 2011] Choi, H. J., Fukui, M., and Zhu, B. T. (2011). Role of cyclin b1/cdc2 up-regulation in the development of mitotic prometaphase arrest in human breast cancer cells treated with nocodazole. *PloS one*, 6(8) :e24312.

[Choi et al., 2018] Choi, K., Medley, J. K., König, M., Stocking, K., Smith, L., Gu, S., and Sauro, H. M. (2018). Tellurium : an extensible python-based modeling environment for systems and synthetic biology. *Biosystems*, 171 :74–79.

[Chow, 2013] Chow, T. L. (2013). *Classical mechanics*. CRC press.

[Chuang et al., 2010] Chuang, H.-Y., Hofree, M., and Ideker, T. (2010). A decade of systems biology. *Annual review of cell and developmental biology*, 26 :721–744.

[Clenet et al., 2023] Clenet, M., Massol, F., and Najim, J. (2023). Equilibrium and surviving species in a large lotka–volterra system of differential equations. *Journal of Mathematical Biology*, 87(1) :13.

[Clevers, 2013] Clevers, H. (2013). The intestinal crypt, a prototype stem cell compartment. *Cell*, 154(2) :274–284.

[Cogoni et al., 2024] Cogoni, F., Bernard, D., Kazhen, R., Valitutti, S., Lobjois, V., and Cussat-Blanc, S. (2024). Isicell : A participatory methodology and platform for collaborative agent-based modeling in cell biology. In *ALIFE 2024 : Proceedings of the 2024 Artificial Life Conference*. MIT Press.

[Cooper et al., 2020] Cooper, F. R., Baker, R. E., Bernabeu, M. O., Bordas, R., Bowler, L., Bueno-Orovio, A., Byrne, H. M., Carapella, V., Cardone-Noott, L., Jonatha, C., et al. (2020). Chaste : cancer, heart and soft tissue environment. *Journal of Open Source Software*, 5(47).

[Cordy and Shukla, 1992] Cordy, J. R. and Shukla, M. (1992). *Practical metaprogramming*. Queen's University of Kingston. Department of Computing and Information Science.

[Corrias et al., 2013] Corrias, A., Du, P., and Buist, M. L. (2013). Modelling tissue electrophysiology in the gi tract : past, present and future. *New Advances in Gastrointestinal Motility Research*, pages 167–195.

[Corrias et al., 2012] Corrias, A., Pathmanathan, P., Gavaghan, D. J., and Buist, M. L. (2012). Modelling tissue electrophysiology with multiple cell types : applications of the extended bidomain framework. *Integrative biology*, 4(2) :192–201.

[Cottom, 2003] Cottom, T. L. (2003). Using swig to bind c++ to python. *Computing in Science & Engineering*, 5(2) :88–97.

[Cresson and Sonner, 2018] Cresson, J. and Sonner, S. (2018). A note on a derivation method for sde models : Applications in biology and viability criteria. *Stochastic Analysis and Applications*, 36(2) :224–239.

[Cryan and Mombereau, 2004] Cryan, J. F. and Mombereau, C. (2004). In search of a depressed mouse : utility of models for studying depression-related behavior in genetically modified mice. *Molecular psychiatry*, 9(4) :326–357.

[Cuevas, 2020] Cuevas, E. (2020). An agent-based model to evaluate the covid-19 transmission risks in facilities. *Computers in biology and medicine*, 121 :103827.

[Cui et al., 2020] Cui, M. et al. (2020). Introduction to the k-means clustering algorithm based on the elbow method. *Accounting, Auditing and Finance*, 1(1) :5–8.

[Cussat-Blanc, 2020] Cussat-Blanc, S. (2020). When artificial life inspires (computational) biology.

[Dalle Nogare and Chitnis, 2020] Dalle Nogare, D. and Chitnis, A. B. (2020). Netlogo agent-based models as tools for understanding the self-organization of cell fate, morphogenesis and collective migration of the zebrafish posterior lateral line primordium. In *Seminars in cell & developmental biology*, volume 100, pages 186–198. Elsevier.

[Danchilla, 2012] Danchilla, B. (2012). Three. js framework. *Beginning WebGL for HTML5*, pages 173–203.

[Darsey et al., 2015] Darsey, J. A., Griffin, W. O., Joginipelli, S., and Melapu, V. K. (2015). Architecture and biological applications of artificial neural networks : a tuberculosis perspective. *Artificial neural networks*, pages 269–283.

[Darwin, 1859] Darwin, C. (1859). On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life. *London : Murray*.

[Darwin, 1872a] Darwin, C. (1872a). *The descent of man, and selection in relation to sex*, volume 2. D. Appleton.

[Darwin, 1872b] Darwin, C. (1872b). On the origin of species.

[Davey and MacLean, 2006] Davey, R. A. and MacLean, H. E. (2006). Current and future approaches using genetically modified mice in endocrine

research. *American Journal of Physiology-Endocrinology and Metabolism*, 291(3) :E429–E438.

[Davidson, 1970] Davidson, D. (1970). Mental events. reprinted in d. davidson (1980). *Essays on actions and events*, pages 201–224.

[De Rainville et al., 2012] De Rainville, F.-M., Fortin, F.-A., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). Deap : A python framework for evolutionary algorithms. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 85–92.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2) :182–197.

[Dethlefsen et al., 1968] Dethlefsen, L. A., Prewitt, J., and Mendelsohn, M. (1968). Analysis of tumor growth curves. *Journal of the National Cancer Institute*, 40(2) :389–405.

[Deutsch et al., 2021] Deutsch, A., Nava-Sedeño, J. M., Syga, S., and Hatzikirou, H. (2021). Bio-lgca : a cellular automaton modelling class for analysing collective cell migration. *PLoS computational biology*, 17(6) :e1009066.

[Dewar et al., 2010] Dewar, M. A., Kadirkamanathan, V., Opper, M., and Sanguinetti, G. (2010). Parameter estimation and inference for stochastic reaction-diffusion systems : application to morphogenesis in d. melanogaster. *BMC Systems Biology*, 4 :1–9.

[Dhillon, 2015] Dhillon, S. (2015). Palbociclib : first global approval. *Drugs*, 75 :543–551.

[Diambra et al., 2015] Diambra, L., Senthivel, V. R., Menendez, D. B., and Isalan, M. (2015). Cooperativity to increase turing pattern space for synthetic biology. *ACS synthetic biology*, 4(2) :177–186.

[Dijksterhuis et al., 1959] Dijksterhuis, E. J. et al. (1959). The origins of classical mechanics from aristotle to newton. *Critical problems in the history of science*, pages 163–90.

[Donovan, 1988] Donovan, A. (1988). Lavoisier and the origins of modern chemistry. *Osiris*, 4 :214–231.

[Dray et al., 2006] Dray, A., Perez, P., LePage, C., D'Aquino, P., and White, I. (2006). 12. atollgame : a companion modelling experience in the pacific. *Complex Science for a Complex World*, page 255.

[Dray et al., 2021] Dray, K. E., Edelstein, H. I., Dreyer, K. S., and Leonard, J. N. (2021). Control of mammalian cell-based devices with genetic programming. *Current opinion in systems biology*, 28 :100372.

[Duke, 2002] Duke, D. W. (2002). Hipparchus' coordinate system. *Archive for history of exact sciences*, 56(5) :427–433.

[Dumas and Ter Hofstede, 2001] Dumas, M. and Ter Hofstede, A. H. (2001). Uml activity diagrams as a workflow specification language. In *International conference on the unified modeling language*, pages 76–90. Springer.

[Edwards and Cottage, 2003] Edwards, Y. J. and Cottage, A. (2003). Bioinformatics methods to predict protein structure and function : A practical approach. *Molecular biotechnology*, 23 :139–166.

[Einstein, 1905] Einstein, A. (1905). Does the inertia of a body depend upon its energy-content. *Annalen der physik*, 18(13) :639–641.

[Einstein, 1915a] Einstein, A. (1915a). Die feldgleichungen der gravitation. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften*, pages 844–847.

[Einstein, 1915b] Einstein, A. (1915b). Erklärung der perihelbewegung des merkur aus der allgemeinen relativitätstheorie. *Sitzungsberichte der königlich preussischen Akademie der Wissenschaften*, pages 831–839.

[Einstein, 2003] Einstein, A. (2003). *The meaning of relativity*. Routledge.

[Epstein and Axtell, 1996] Epstein, J. M. and Axtell, R. (1996). *Growing artificial societies : social science from the bottom up*. Brookings Institution Press.

[Esposito and Picchiami, 2021] Esposito, M. and Picchiami, L. (2021). A comparative study of ai search methods for personalised cancer therapy synthesis in copasi. In *International Conference of the Italian Association for Artificial Intelligence*, pages 638–654. Springer.

[Étienne, 2013] Étienne, M. (2013). *Companion modelling : a participatory approach to support sustainable development*. Springer.

[Feldberg et al., 1988] Feldberg, R. S., Chang, S., Kotik, A., Nadler, M., Neuwirth, Z., Sundstrom, D., and Thompson, N. (1988). In vitro mechanism of inhibition of bacterial cell growth by allicin. *Antimicrobial agents and chemotherapy*, 32(12) :1763–1768.

[Fernández-Hernández et al., 2013] Fernández-Hernández, I., Rhiner, C., and Moreno, E. (2013). Adult neurogenesis in drosophila. *Cell reports*, 3(6) :1857–1865.

[Fletcher et al., 2013] Fletcher, A. G., Osborne, J. M., Maini, P. K., and Gavaghan, D. J. (2013). Implementing vertex dynamics models of cell populations in biology within a consistent computational framework. *Progress in biophysics and molecular biology*, 113(2) :299–326.

[Fletcher et al., 2014] Fletcher, A. G., Osterfield, M., Baker, R. E., and Shvartsman, S. Y. (2014). Vertex models of epithelial morphogenesis. *Biophysical journal*, 106(11) :2291–2304.

[Fletcher et al., 2006] Fletcher, P. S., Elliott, J., Grivel, J.-C., Margolis, Leonid ; Anton, P., McGowan, I., and Shattock, R. J. (2006). Ex vivo culture of human colorectal tissue for the evaluation of candidate microbicides. *AIDS*, 20(9) :1237–1245.

[Fodor, 1980] Fodor, J. A. (1980). Special sciences, or the disunity of science as a working hypothesis. In *The Language and Thought Series*, pages 120–133. Harvard University Press.

[Fofonjka and Milinkovitch, 2021] Fofonjka, A. and Milinkovitch, M. C. (2021). Reaction-diffusion in a growing 3d domain of skin scales generates a discrete cellular automaton. *Nature Communications*, 12(1) :2433.

[Fogel, 2006] Fogel, D. (2006). Nils barricelli - artificial life, coevolution, self-adaptation. *IEEE Computational Intelligence Magazine*, 1(1) :41–45.

[Fortuna et al., 2020] Fortuna, I., Perrone, G. C., Krug, M. S., Susin, E., Belmonte, J. M., Thomas, G. L., Glazier, J. A., and de Almeida, R. M. (2020). Compucell3d simulations reproduce mesenchymal cell migration on flat substrates. *Biophysical Journal*, 118(11) :2801–2815.

[Fresnel, 1868] Fresnel, A. (1868). *Oeuvres complètes*, volume 2. Imprimerie impériale.

[Fuentes et al., 2005] Fuentes, G., Talavera, C., Oropeza, C., Desjardins, Y., and Santamaria, J. M. (2005). Exogenous sucrose can decrease in vitro photosynthesis but improve field survival and growth of coconut (cocos nucifera l.) in vitro plantlets. *In Vitro Cellular Developmental Biology-Plant*, 41 :69–76.

[Fujita et al., 2015] Fujita, T., Nishikori, D., Nakano, K., and Ito, Y. (2015). Efficient gpu implementations for the conway's game of life. In *2015 Third International Symposium on Computing and Networking (CANDAR)*, pages 11–20. IEEE.

[Funahashi et al., 2008] Funahashi, A., Matsuoka, Y., Jouraku, A., Morohashi, M., Kikuchi, N., and Kitano, H. (2008). Celldesigner 3.5 : a versatile modeling tool for biochemical networks. *Proceedings of the IEEE*, 96(8) :1254–1265.

[Funahashi et al., 2003] Funahashi, A., Morohashi, M., Kitano, H., and Tanimura, N. (2003). Celldesigner : a process diagram editor for gene-regulatory and biochemical networks. *Biosilico*, 1(5) :159–162.

[Gage, 2019] Gage, F. H. (2019). Adult neurogenesis in mammals. *Science*, 364(6443) :827–828.

[Ganz and Brand, 2016] Ganz, J. and Brand, M. (2016). Adult neurogenesis in fish. *Cold Spring Harbor perspectives in biology*, 8(7) :a019018.

[Garberg et al., 2005] Garberg, P., Ball, M., Borg, N., Cecchelli, R., Fenart, L., Hurst, R., Lindmark, T., Mabondzo, A., Nilsson, J., Raub, T., et al. (2005). In vitro models for the blood–brain barrier. *Toxicology in vitro*, 19(3) :299–334.

[Garcia et al., 2020] Garcia, C. A., Vendé, J., Konerira, N., Kalla, J., Nay, M., Dray, A., Delay, M., Waeber, P. O., Stoudmann, N., Bose, A., et al. (2020). Coffee, farmers, and trees—shifting rights accelerates changing landscapes. *Forests*, 11(4) :480.

[Garcia-Lopez et al., 2009] Garcia-Lopez, R., Pombero, A., and Martinez, S. (2009). Fate map of the chick embryo neural tube. *Development, growth & differentiation*, 51(3) :145–165.

[Gardner, 1970] Gardner, M. (1970). Mathematical games-the fantastic combinations of john conway's new solitaire game, life, 1970. *Scientific American, October*, pages 120–123.

[Gáspári, 2020] Gáspári, Z. (2020). *Structural Bioinformatics*. Springer.

[Geurts et al., 2009] Geurts, P., Irrthum, A., and Wehenkel, L. (2009). Supervised learning with decision tree-based methods in computational and systems biology. *Molecular Biosystems*, 5(12) :1593–1605.

[Ghaffarizadeh et al., 2018] Ghaffarizadeh, A., Heiland, R., Friedman, S. H., Mumenthaler, S. M., and Macklin, P. (2018). Physicell : An open source physics-based cell simulator for 3-d multicellular systems. *PLoS computational biology*, 14(2) :e1005991.

[Ghiasi and Zendehboudi, 2021] Ghiasi, M. M. and Zendehboudi, S. (2021). Application of decision tree-based ensemble learning in the classification of breast cancer. *Computers in biology and medicine*, 128 :104089.

[Ghosh et al., 2007] Ghosh, P., Ghosh, S., Basu, K., and Das, S. (2007). Modeling protein-dna binding time in stochastic discrete event simulation of biological processes. In *2007 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, pages 439–446. IEEE.

[Gingerich, 1973] Gingerich, O. (1973). From copernicus to kepler : Heliocentrism as model and as reality. *Proceedings of the American Philosophical Society*, 117(6) :513–522.

[Gohlke et al., 2004] Gohlke, J. M., Griffith, W. C., and Faustman, E. M. (2004). The role of cell death during neocortical neurogenesis and synaptogenesis : implications from a computational model for the rat and mouse. *Developmental brain research*, 151(1-2) :43–54.

[Goldsmith, 2004] Goldsmith, P. (2004). Zebrafish as a pharmacological tool : the how, why and when. *Current opinion in pharmacology*, 4(5) :504–512.

[Goldstein et al., 2002] Goldstein, R. S., Drukker, M., Reubinoff, B. E., and Benvenisty, N. (2002). Integration and differentiation of human embryonic stem cells transplanted to the chick embryo. *Developmental dynamics : an official publication of the American Association of Anatomists*, 225(1) :80–86.

[Goodwin and Trainor, 1985] Goodwin, B. C. and Trainor, L. E. (1985). Tip and whorl morphogenesis in acetabularia by calcium-regulated strain fields. *Journal of theoretical biology*, 117(1) :79–106.

[Gopalakrishnan et al., 2013] Gopalakrishnan, V., Kim, M., and An, G. (2013). Using an agent-based model to examine the role of dynamic bacterial virulence potential in the pathogenesis of surgical site infection. *Advances in wound care*, 2(9) :510–526.

[Götz and Huttner, 2005] Götz, M. and Huttner, W. B. (2005). The cell biology of neurogenesis. *Nature reviews Molecular cell biology*, 6(10) :777–788.

[Gould and Gross, 2002] Gould, E. and Gross, C. G. (2002). Neurogenesis in adult mammals : some progress and problems. *Journal of Neuroscience*, 22(3) :619–623.

[Grand, 1975] Grand, H. L. (1975). The “conversion” of c.-l. berthollet to lavoisiers’ chemistry. *Ambix*, 22(1) :58–70.

[Grapí and Izquierdo, 1997] Grapí, P. and Izquierdo, M. (1997). Berthollet's conception of chemical change in context. *Ambix*, 44(3) :113–130.

[Green et al., 2021] Green, R., Lo Vercio, L., Dauter, A., Guo, S. H., Robertson, S., Marchini, M., Vidal Garcia, M., Zhao, X., Marcucio, R., Forkert, N., et al. (2021). Integration of cellular dynamics and morphology to understand mouse facial development. *The FASEB Journal*, 35.

[Grimes et al., 2014] Grimes, D. R., Kelly, C., Bloch, K., and Partridge, M. (2014). A method for estimating the oxygen consumption rate in multicellular tumour spheroids. *Journal of The Royal Society Interface*, 11(92) :20131124.

[Grogan et al., 2017] Grogan, J. A., Connor, A. J., Markelc, B., Muschel, R. J., Maini, P. K., Byrne, H. M., and Pitt-Francis, J. M. (2017). Microvessel chaste : an open library for spatial modeling of vascularized tissues. *Biophysical Journal*, 112(9) :1767–1772.

[Gu and Bourne, 2009] Gu, J. and Bourne, P. E. (2009). *Structural bioinformatics*, volume 44. John Wiley & Sons.

[Guillouzo, 1998] Guillouzo, A. (1998). Liver cell models in in vitro toxicology. *Environmental health perspectives*, 106(suppl 2) :511–532.

[Guo et al., 2008] Guo, Z., Sloot, P. M., and Tay, J. C. (2008). A hybrid agent-based approach for modeling microbiological systems. *Journal of theoretical biology*, 255(2) :163–175.

[Gurumurthy et al., 2020] Gurumurthy, C. B., Quadros, R. M., Richardson, G. P., Poluektova, L. Y., Mansour, S. L., and Ohtsuka, M. (2020). Genetically modified mouse models to help fight covid-19. *Nature protocols*, 15(12) :3777–3787.

[Gurung et al., 2022] Gurung, T. R., Le Page, C., and Trébuil, G. (2022). Collaborative modeling and simulation to mitigate high-elevation rangeland degradation in eastern bhutan. *Mountain Research and Development*, 42(4) :D14–D24.

[Hamilton, 1998] Hamilton, G. (1998). Multicellular spheroids as an in vitro tumor model. *Cancer letters*, 131(1) :29–34.

[Hare et al., 2003] Hare, M., Letcher, R. A., and Jakeman, A. J. (2003). Participatory modelling in natural resource management : a comparison of four case studies. *Integrated Assessment*, 4(2) :62–72.

[Harel, 1987] Harel, D. (1987). Statecharts : A visual formalism for complex systems. *Science of computer programming*, 8(3) :231–274.

[Hartenstein and Wodarz, 2013] Hartenstein, V. and Wodarz, A. (2013). Initial neurogenesis in drosophila. *Wiley Interdisciplinary Reviews : Developmental Biology*, 2(5) :701–721.

[Harvatine and Allen, 2006] Harvatine, K. and Allen, M. (2006). Effects of fatty acid supplements on milk yield and energy balance of lactating dairy cows. *Journal of Dairy Science*, 89(3) :1081–1091.

[Heiland et al., 2023] Heiland, R., Bergman, D., Lyons, B., Cass, J., Rocha, H. L., Ruscone, M., Noël, V., and Macklin, P. (2023). Physicell studio : a graphical tool to make agent-based modeling more accessible. *bioRxiv*.

[Helton and Davis, 2003] Helton, J. C. and Davis, F. J. (2003). Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System Safety*, 81(1) :23–69.

[Herman and Usher, 2017] Herman, J. and Usher, W. (2017). Salib : An open-source python library for sensitivity analysis. *Journal of Open Source Software*, 2(9) :97.

[Hesper and Hogeweg, 1970] Hesper, B. and Hogeweg, P. (1970). Bioinformatica : een werkconcept. *Kameleon*, 1(6) :28–29.

[Hogeweg, 1978] Hogeweg, P. (1978). Simulating the growth of cellular forms. *Simulation*, 31(3) :90–96.

[Hogeweg and Hesper, 1979] Hogeweg, P. and Hesper, B. (1979). Heterarchical, selfstructuring simulation systems : concepts and applications in biology. *Methodologies in systems modelling and simulation*, pages 221–231.

[Hogeweg and Hesper, 1983] Hogeweg, P. and Hesper, B. (1983). The ontogeny of the interaction structure in bumble bee colonies : a mirror model. *Behavioral Ecology and Sociobiology*, 12 :271–283.

[Hogeweg and Hesper, 1985] Hogeweg, P. and Hesper, B. (1985). Socioinformatic processes : Mirror modelling methodology. *Journal of theoretical Biology*, 113(2) :311–330.

[Holland and Miller, 1991] Holland, J. H. and Miller, J. H. (1991). Artificial adaptive agents in economic theory. *The American economic review*, 81(2) :365–370.

[Holmes, 1985] Holmes, F. L. (1985). *Lavoisier and the chemistry of life : An exploration of scientific creativity*. Number 4. Univ of Wisconsin Press.

[Hoops et al., 2006] Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., and Kummer, U. (2006). Copasi—a complex pathway simulator. *Bioinformatics*, 22(24) :3067–3074.

[Horner et al., 2010] Horner, D. S., Pavesi, G., Castrignano, T., De Meo, P. D., Liuni, S., Sammeth, M., Picardi, E., and Pesole, G. (2010). Bioinformatics approaches for genomics and post genomics applications of next-generation sequencing. *Briefings in bioinformatics*, 11(2) :181–197.

[Hossard et al., 2022] Hossard, L., Tardivo, C., Barbier, J.-M., Cittadini, R., Delmotte, S., and Le Page, C. (2022). Embedding the integrated assessment of agricultural systems in a companion modeling process to debate and enhance their sustainability. *Agronomy for Sustainable Development*, 42(1) :11.

[Howe et al., 2013] Howe, K., Clark, M. D., Torroja, C. F., Torrance, J., Berthelot, C., Muffato, M., Collins, J. E., Humphray, S., McLaren, K., Matthews, L., et al. (2013). The zebrafish reference genome sequence and its relationship to the human genome. *Nature*, 496(7446) :498–503.

[Hristov and Broderick, 1996] Hristov, A. N. and Broderick, G. A. (1996). Synthesis of microbial protein in ruminally cannulated cows fed alfalfa silage, alfalfa hay, or corn silage1. *Journal of Dairy Science*, 79(9) :1627–1637.

[Hubbert, 1937] Hubbert, M. K. (1937). Theory of scale models as applied to the study of geologic structures. *Bulletin of the Geological Society of America*, 48(10) :1459–1520.

[Hucka et al., 2003] Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., et al. (2003). The systems biology markup language (sbml) : a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4) :524–531.

[Humaira and Rasyidah, 2020] Humaira, H. and Rasyidah, R. (2020). Determining the appropiate cluster number using elbow method for k-means algorithm. In *Proceedings of the 2nd Workshop on Multidisciplinary and Applications (WMA) 2018, 24-25 January 2018, Padang, Indonesia*.

[HUSSEY, 1976] HUSSEY, G. (1976). In vitro release of axillary shoots from apical dominance in monocotyledonous plantlets. *Annals of Botany*, 40(170) :1323–1325.

[Israels and Israels, 2000] Israels, E. and Israels, L. (2000). The cell cycle. *The oncologist*, 5(6) :510–513.

[Jacob, 1985] Jacob, R. J. K. (1985). A state transition diagram language for visual programming. *Computer*, 18(08) :51–59.

[Jacobson and Jacobson, 1991] Jacobson, M. and Jacobson, M. (1991). Histogenesis and morphogenesis of cortical structures. *Developmental neurobiology*, pages 401–451.

[Jahel et al., 2023] Jahel, C., Bourgeois, R., Bourgoin, J., De Lattre-Gasquet, M., Delay, E., Dumas, P., Le Page, C., Piraux, M., Prudhomme, R., et al. (2023). The future of social-ecological systems at the crossroads of quantitative and qualitative methods. *Technological Forecasting and Social Change*, 193 :122624.

[Jain et al., 2023] Jain, S., Shrestha, A., and Nichele, S. (2023). Capturing emerging complexity in lenia. In *Italian Workshop on Artificial Life and Evolutionary Computation*, pages 41–53. Springer.

[Jalajakumari et al., 2022] Jalajakumari, S., Devaraj, S., Manivel, T., and Ramasamy, S. (2022). Analysis of cell proliferation and apoptosis in virtual model. In *Proceedings of the Bulgarian Academy of Sciences*, volume 75, pages 1483–1490.

[Johnson and Walker, 1999] Johnson, D. G. and Walker, C. L. (1999). Cyclins and cell cycle checkpoints. *Annual review of pharmacology and toxicology*, 39(1) :295–312.

[Jones, 2019] Jones, D. T. (2019). Setting the standards for machine learning in biology. *Nature Reviews Molecular Cell Biology*, 20(11) :659–660.

[Kačeniauskas et al., 2010] Kačeniauskas, A., Pacevič, R., Bugajev, A., and Katkevičius, T. (2010). Efficient visualization by using paraview software on balticgrid. *Information Technology and Control*, 39(2).

[Kerr et al., 2021] Kerr, C. C., Stuart, R. M., Mistry, D., Abeysuriya, R. G., Rosenfeld, K., Hart, G. R., Núñez, R. C., Cohen, J. A., Selvaraj, P., Hagedorn, B., et al. (2021). Covasim : an agent-based model of covid-19 dynamics and interventions. *PLOS Computational Biology*, 17(7) :e1009149.

[Khazen et al., 2019] Khazen, R., Müller, S., Lafouresse, F., Valitutti, S., and Cussat-Blanc, S. (2019). Sequential adjustment of cytotoxic t lymphocyte densities improves efficacy in controlling tumor growth. *Scientific reports*, 9(1) :1–11.

[Kim, 2008] Kim, J. (2008). Reduction and reductive explanation : Is one possible without the other. *Being reduced*, pages 93–114.

[Kim et al., 2023] Kim, J., Park, S., and Kim, M. (2023). Safety map : Disaster management road network for urban resilience. *Sustainable Cities and Society*, 96 :104650.

[Kincaid and Lavoie, 1989] Kincaid, C. B. and Lavoie, K. H. (1989). Inhibition of bacterial growth in vitro following stimulation with high voltage, monophasic, pulsed current. *Physical Therapy*, 69(8) :651–655.

[Kishigami et al., 2006] Kishigami, S., Wakayama, S., Van Thuan, N., Ohta, H., Mizutani, E., Hikichi, T., Bui, H.-T., Balbach, S., Ogura, A., Boiani, M., et al. (2006). Production of cloned mice by somatic cellnuclear transfer. *Nature protocols*, 1(1) :125–138.

[Kitano, 2002] Kitano, H. (2002). Computational systems biology. *Nature*, 420(6912) :206–210.

[Koyré, 2013] Koyré, A. (2013). *The Astronomical Revolution : Copernicus-Kepler-Borelli*. Routledge.

[Krieg and Melton, 1987] Krieg, P. A. and Melton, D. (1987). [25] in vitro rna synthesis with sp6 rna polymerase. In *Methods in enzymology*, volume 155, pages 397–415. Elsevier.

[Kuhn, 1992] Kuhn, T. S. (1992). *The Copernican revolution : Planetary astronomy in the development of Western thought*. Harvard University Press.

[Kumaria and Tolias, 2008] Kumaria, A. and Tolias, C. (2008). In vitro models of neurotrauma. *British journal of neurosurgery*, 22(2) :200–206.

[KUNZ-SCHUGHART et al., 1998] KUNZ-SCHUGHART, L. A., Kreutz, M., and Knuechel, R. (1998). Multicellular spheroids : a three-dimensional in vitro culture system to study tumour biology. *International journal of experimental pathology*, 79(1) :1–23.

[Kurhekar and Deshpande, 2014] Kurhekar, M. and Deshpande, U. (2014). A deterministic model of the adult subventricular neurogenesis. In *Proceedings of the 8th International Conference on Bioinspired Information and Communications Technologies*, pages 71–74.

[Kurhekar and Deshpande, 2015] Kurhekar, M. P. and Deshpande, U. A. (2015). Agent based deterministic model of the adult subventricular neurogenesis. *Nano Communication Networks*, 6(3) :124–132.

[Kusumoto and Yuasa, 2019] Kusumoto, D. and Yuasa, S. (2019). The application of convolutional neural network to stem cell biology. *Inflammation and regeneration*, 39(1) :1–7.

[Langton, 1987] Langton, C. (1987). Artificial life. Technical report, Los Alamos National Lab., NM (USA).

[Langton, 1986] Langton, C. G. (1986). Studying artificial life with cellular automata. *Physica D : Nonlinear Phenomena*, 22(1-3) :120–149.

[Lardon et al., 2011] Lardon, L. A., Merkey, B. V., Martins, S., Dötsch, A., Picioreanu, C., Kreft, J.-U., and Smets, B. F. (2011). idynomics : next-generation individual-based modelling of biofilms. *Environmental microbiology*, 13(9) :2416–2434.

[Laurent et al., 2013] Laurent, J., Frongia, C., Cazales, M., Mondesert, O., Ducommun, B., and Lobjois, V. (2013). Multicellular tumor spheroid models to explore cell cycle checkpoints in 3d. *BMC cancer*, 13(1) :1–12.

[Lebiedz and Maurer, 2004] Lebiedz, D. and Maurer, H. (2004). External optimal control of self-organisation dynamics in a chemotaxis reaction diffusion system. *Systems Biology*, 1(2) :222–229.

[Lehr et al., 1993] Lehr, H. A., Leunig, M., Menger, M. D., Nolte, D., and Messmer, K. (1993). Dorsal skinfold chamber technique for intravital microscopy in nude mice. *The American journal of pathology*, 143(4) :1055–1062.

[Lemay and Oesper, 1946] Lemay, P. and Oesper, R. E. (1946). Claude louis berthollet (1748-1822). *Journal of Chemical Education*, 23(4) :158.

[Leung and Salga, 2010] Leung, C. and Salga, A. (2010). Enabling webgl. In *Proceedings of the 19th international conference on World wide web*, pages 1369–1370.

[Li et al., 2008] Li, N. Y., Verdolini, K., Clermont, G., Mi, Q., Rubinstein, E. N., Hebda, P. A., and Vodovotz, Y. (2008). A patient-specific in silico model of inflammation and healing tested in acute vocal fold injury. *PloS one*, 3(7) :e2789.

[Lilis and Savidis, 2019] Lilis, Y. and Savidis, A. (2019). A survey of metaprogramming languages. *ACM Computing Surveys (CSUR)*, 52(6) :1–39.

[Lindberg, 1997] Lindberg, D. C. (1997). Roger bacon on light, vision, and the universal emanation of force. In *Roger Bacon and the sciences*, pages 243–275. Brill.

[Lindenmayer, 1968a] Lindenmayer, A. (1968a). Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of theoretical biology*, 18(3) :280–299.

[Lindenmayer, 1968b] Lindenmayer, A. (1968b). Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of theoretical biology*, 18(3) :300–315.

[Lipp and Bonfanti, 2016] Lipp, H.-P. and Bonfanti, L. (2016). Adult neurogenesis in mammals : variations and confusions. *Brain, behavior and evolution*, 87(3) :205–221.

[Lirola et al., 2017] Lirola, J. M., Castañeda, E., Lauret, B., and Khayet, M. (2017). A review on experimental research using scale models for buildings : Application and methodologies. *Energy and Buildings*, 142 :72–110.

[Lisle, 2020] Lisle, R. J. (2020). *Geological structures and maps : A practical guide*. Butterworth-Heinemann.

[Liu et al., 2016] Liu, F., Zhang, S.-W., Guo, W.-F., Wei, Z.-G., and Chen, L. (2016). Inference of gene regulatory network based on local bayesian networks. *PLoS computational biology*, 12(8) :e1005024.

[Liu et al., 2022] Liu, L., Hong, F., Liu, H., Zhou, X., Jiang, S., Šulc, P., Jiang, J.-H., and Yan, H. (2022). A localized dna finite-state machine with temporal resolution. *Science advances*, 8(12) :eabm9530.

[Livingstone, 2008] Livingstone, D. J. (2008). *Artificial neural networks : methods and applications*. Springer.

[Lloyd, 1961] Lloyd, G. E. R. (1961). The development of aristotle's theory of the classification of animals. *Phronesis*, pages 59–81.

[Loew and Schaff, 2001] Loew, L. M. and Schaff, J. C. (2001). The virtual cell : a software environment for computational cell biology. *TRENDS in Biotechnology*, 19(10) :401–406.

[Loh, 1996] Loh, W.-L. (1996). On latin hypercube sampling. *The annals of statistics*, 24(5) :2058–2080.

[Lotka, 1925] Lotka, A. J. (1925). *Elements of physical biology*. Williams & Wilkins.

[Lytinen and Railsback, 2012] Lytinen, S. L. and Railsback, S. F. (2012). The evolution of agent-based simulation platforms : a review of netlogo 5.0 and relogo. In *Proceedings of the fourth international symposium on agent-based modeling and simulation*.

[Macnamara, 2021] Macnamara, C. K. (2021). Biomechanical modelling of cancer : agent-based force-based models of solid tumours within the context of the tumour microenvironment. *Computational and Systems Oncology*, 1(2) :e1018.

[MacRae and Peterson, 2023] MacRae, C. A. and Peterson, R. T. (2023). Zebrafish as a mainstream model for in vivo systems pharmacology and toxicology. *Annual Review of Pharmacology and Toxicology*, 63 :43–64.

[Maini et al., 2012] Maini, P. K., Woolley, T. E., Baker, R. E., Gaffney, E. A., and Lee, S. S. (2012). Turing's model for biological pattern formation and the robustness problem. *Interface focus*, 2(4) :487–496.

[Makris et al., 2021] Makris, A., Tserpes, K., Spiliopoulos, G., Zissis, D., and Anagnostopoulos, D. (2021). Mongodb vs postgresql : A comparative study on performance aspects. *GeoInformatica*, 25 :243–268.

[Maltman, 2012] Maltman, A. (2012). *Geological maps : an introduction*. Springer Science & Business Media.

[Manukyan et al., 2017] Manukyan, L., Montandon, S. A., Fofonjka, A., Smirnov, S., and Milinkovitch, M. C. (2017). A living mesoscopic cellular automaton made of skin scales. *Nature*, 544(7649) :173–179.

[Mao et al., 2018] Mao, X., McManaway, S., Jaiswal, J. K., Patel, P. B., Wilson, W. R., Hicks, K. O., and Bogle, G. (2018). An agent-based model for drug-radiation interactions in the tumour microenvironment : Hypoxia-activated prodrug sn30000 in multicellular tumour spheroids. *PLoS computational biology*, 14(10) :e1006469.

[Mariani et al., 2014] Mariani, E., Filardo, G., Canella, V., Berlingeri, A., Bielli, A., Cattini, L., Landini, M. P., Kon, E., Marcacci, M., and Facchini, A. (2014). Platelet-rich plasma affects bacterial growth in vitro. *Cytotherapy*, 16(9) :1294–1304.

[Martinez et al., 2013] Martinez, G. J., McIntosh, H. V., and Mora, J. C. S. T. (2013). Cellular automata.

[Martínez-Pérez et al., 2007] Martínez-Pérez, I. M., Zhang, G., Ignatova, Z., and Zimmermann, K.-H. (2007). Computational genes : a tool for molecular diagnosis and therapy of aberrant mutational phenotype. *BMC bioinformatics*, 8(1) :1–9.

[Martins et al., 2021] Martins, P., Tomé, P., Wanzeller, C., Sá, F., and Abbasi, M. (2021). Comparing oracle and postgresql, performance and optimization. In *Trends and Applications in Information Systems and Technologies : Volume 2 9*, pages 481–490. Springer.

[Mazzocchi et al., 2011] Mazzocchi, F. et al. (2011). The limits of reductionism in biology : what alternatives. *E-LOGOS : Electron J Philos*, 11 :1–19.

[McInnes et al., 2018] McInnes, L., Healy, J., and Melville, J. (2018). Umap : Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv :1802.03426*.

[McKenzie et al., 1998] McKenzie, F. E., Wong, R. C., and Bossert, W. H. (1998). Discrete-event simulation models of plasmodium falciparum malaria. *Simulation*, 71(4) :250–261.

[McKinney, 2012] McKinney, W. (2012). *Python for data analysis : Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.".

[McKinney et al., 2011] McKinney, W. et al. (2011). pandas : a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9) :1–9.

[McLane et al., 2011] McLane, A. J., Semeniuk, C., McDermid, G. J., and Marceau, D. J. (2011). The role of agent-based models in wildlife ecology and management. *Ecological modelling*, 222(8) :1544–1556.

[McLean et al., 2018] McLean, I. C., Schwerdtfeger, L. A., Tobet, S. A., and Henry, C. S. (2018). Powering ex vivo tissue models in microfluidic systems. *Lab Chip*, 18 :1399–1410.

[Medley et al., 2018] Medley, J. K., Choi, K., König, M., Smith, L., Gu, S., Hellerstein, J., Sealfon, S. C., and Sauro, H. M. (2018). Tellurium notebooks—an environment for reproducible dynamical modeling in systems biology. *PLoS computational biology*, 14(6) :e1006220.

[Meier-Schellersheim et al., 2009] Meier-Schellersheim, M., Fraser, I. D., and Klauschen, F. (2009). Multiscale modeling for biologists. *Wiley Interdisciplinary Reviews : Systems Biology and Medicine*, 1(1) :4–14.

[Mendeleev, 2013] Mendeleev, D. I. (2013). *Mendeleev on the periodic law : Selected writings, 1869-1905*. Courier Corporation.

[Merbah et al., 2011] Merbah, M., Introini, A., Fitzgerald, W., Grivel, J. C., Lisco, A., Vanpouille, C., and Margolis, L. (2011). Cervico-vaginal tissue ex vivo as a model to study early events in hiv-1 infection. *American Journal of Reproductive Immunology*, 65(3) :268–278.

[Metzcar et al., 2019] Metzcar, J., Wang, Y., Heiland, R., and Macklin, P. (2019). A review of cell-based computational modeling in cancer biology. *JCO clinical cancer informatics*, 2 :1–13.

[Meurisse et al., 2022] Meurisse, N., Marcot, B. G., Woodberry, O., Barratt, B. I., and Todd, J. H. (2022). Risk analysis frameworks used in biological control and introduction of a novel bayesian network tool. *Risk Analysis*, 42(6) :1255–1276.

[Mirams et al., 2013] Mirams, G. R., Arthurs, C. J., Bernabeu, M. O., Bordas, R., Cooper, J., Corrias, A., Davit, Y., Dunn, S.-J., Fletcher, A. G., Harvey, D. G., et al. (2013). Chaste : an open source c++ library for computational physiology and biology. *PLoS computational biology*, 9(3) :e1002970.

[Molina et al., 2022] Molina, A., Bonnet, F., Pignolet, J., Lobjois, V., Bel-Vialar, S., Gautrais, J., Pituello, F., and Agius, E. (2022). Single-cell imaging of the cell cycle reveals cdc25b-induced heterogeneity of g1 phase length in neural progenitor cells. *Development*, 149(11) :dev199660.

[Moore, 1956] Moore, E. F. (1956). Artificial living plants. *Scientific American*, 195(4) :118–127.

[Moreno and Ribera, 2014] Moreno, R. L. and Ribera, A. B. (2014). Spinal neurons require islet1 for subtype-specific differentiation of electrical excitability. *Neural development*, 9 :1–18.

[Morrison III et al., 2011] Morrison III, B., Elkin, B. S., Dollé, J.-P., and Yarmush, M. L. (2011). In vitro models of traumatic brain injury. *Annual review of biomedical engineering*, 13 :91–126.

[Mörters and Peres, 2010] Mörters, P. and Peres, Y. (2010). *Brownian motion*, volume 30. Cambridge University Press.

[Movilla et al., 2022] Movilla, N., Gonçalves, I. G., Borau, C., and García-Aznar, J. M. (2022). A novel integrated experimental and computational approach to unravel fibroblast motility in response to chemical gradients in 3d collagen matrices. *Integrative Biology*, 14(8-12) :212–227.

[Murray, 1994] Murray, A. (1994). Cell cycle checkpoints. *Current opinion in cell biology*, 6(6) :872–876.

[Nagano et al., 2009] Nagano, K., Shinkawa, T., Mutoh, H., Kondoh, O., Morimoto, S., Inomata, N., Ashihara, M., Ishii, N., Aoki, Y., and Haramura, M. (2009). Phosphoproteomic analysis of distinct tumor cell lines in response to nocodazole treatment. *Proteomics*, 9(10) :2861–2874.

[Nagel, 1949] Nagel, E. (1949). In defense of logic without metaphysics. *The Philosophical Review*, 58(1) :26–34.

[Nagel, 1961] Nagel, E. (1961). The structure of science : Problems in the logic of scientific explanation.

[Needham et al., 2007] Needham, C. J., Bradford, J. R., Bulpitt, A. J., and Westhead, D. R. (2007). A primer on learning in bayesian networks for computational biology. *PLoS computational biology*, 3(8) :e129.

[Neumann, 1948] Neumann, V. (1948). The general and logical theory of automata. In *Lecture at Hixon Symposium 1948*.

[Newell, 1982] Newell, A. (1982). The knowledge level. *Artificial intelligence*, 18(1) :87–127.

[Nguyen et al., 2021] Nguyen, J., Powers, S. T., Urquhart, N., Farrenkopf, T., and Guckert, M. (2021). An overview of agent-based traffic simulators. *Transportation research interdisciplinary perspectives*, 12 :100486.

[Novitch et al., 2001] Novitch, B. G., Chen, A. I., and Jessell, T. M. (2001). Coordinate regulation of motor neuron subtype identity and pan-neuronal properties by the bhlh repressor olig2. *Neuron*, 31(5) :773–789.

[Obaidullah, 2007] Obaidullah, M. (2007). Medical science and islam : an analysis of the contributions of the medieval muslim scholars. *Erişim adresi : https ://www. researchgate. net/publication/270215891*.

[Oda et al., 2004] Oda, K., Kimura, T., Matsuoka, Y., Funahashi, A., Muramatsu, M., Kitano, H., et al. (2004). Molecular interaction map of a macrophage. *AfCS Research Reports*, 2(14) :1–12.

[Oda et al., 2005] Oda, K., Matsuoka, Y., Funahashi, A., and Kitano, H. (2005). A comprehensive pathway map of epidermal growth factor receptor signaling. *Molecular systems biology*, 1(1) :2005–0010.

[O'grady and Pennington, 1966] O'grady, F. and Pennington, J. (1966). Bacterial growth in an in vitro system simulating conditions in the urinary bladder. *British Journal of Experimental Pathology*, 47(2) :152.

[Oh and Oh, 2011] Oh, P. S. and Oh, S. J. (2011). What teachers of science need to know about models : An overview. *International Journal of Science Education*, 33(8) :1109–1130.

[Oishi and Klavins, 2014] Oishi, K. and Klavins, E. (2014). Framework for engineering finite state machines in gene regulatory networks. *ACS synthetic biology*, 3(9) :652–665.

[Okayama and Murase, 2002] Okayama, T. and Murase, H. (2002). Leaf cellular automata. *J. Soc. of High Tech. in Agr*, (14) :3.

[Oppenheim and Putnam, 1958] Oppenheim, P. and Putnam, H. (1958). Unity of science as a working hypothesis.

[Otsuji et al., 2007] Otsuji, M., Ishihara, S., Co, C., Kaibuchi, K., Mochizuki, A., and Kuroda, S. (2007). A mass conserved reaction–diffusion system captures properties of cell polarity. *PLoS computational biology*, 3(6) :e108.

[Ozik et al., 2018] Ozik, J., Collier, N., Wozniak, J. M., Macal, C., Cockrell, C., Friedman, S. H., Ghaffarizadeh, A., Heiland, R., An, G., and Macklin, P. (2018). High-throughput cancer hypothesis testing with an integrated physicell-emews workflow. *BMC bioinformatics*, 19(18) :483.

[Pappalardo et al., 2010] Pappalardo, F., Pennisi, M., Castiglione, F., and Motta, S. (2010). Vaccine protocols optimization : in silico experiences. *Biotechnology Advances*, 28(1) :82–93.

[Park et al., 2002] Park, H.-C., Mehta, A., Richardson, J. S., and Appel, B. (2002). olig2 is required for zebrafish primary motor neuron and oligodendrocyte development. *Developmental biology*, 248(2) :356–368.

[Pascalie et al., 2011] Pascalie, J., Lobjois, V., Luga, H., Ducommun, B., and Duthen, Y. (2011). A checkpoint-orientated model to simulate unconstrained proliferation of cells. In *ECAL*, pages 630–637. Citeseer.

[Pascalie et al., 2010] Pascalie, J., Luga, H., Lobjois, V., Ducommun, B., and Duthen, Y. (2010). A checkpoint-orientated modelling for cell cycle simulation. In *International Conference on Bio-Inspired Models of Network, Information, and Computing Systems*, pages 40–47. Springer.

[Patel and Goyal, 2007] Patel, J. L. and Goyal, R. K. (2007). Applications of artificial neural networks in medical science. *Current clinical pharmacology*, 2(3) :217–226.

[Peak et al., 2023] Peak, D., Hogan, M. T., and Mott, K. A. (2023). Stomatal patchiness and cellular computing. *Proceedings of the National Academy of Sciences*, 120(14) :e2220270120.

[Peak et al., 2004] Peak, D., West, J. D., Messinger, S. M., and Mott, K. A. (2004). Evidence for complex, collective dynamics and emergent, distributed computation in plants. *Proceedings of the National Academy of Sciences*, 101(4) :918–922.

[Penlou et al., 2023a] Penlou, B., Roche, O., Manga, M., and van den Wildenberg, S. (2023a). Experimental measurement of enhanced and hindered particle settling in turbulent gas-particle suspensions, and geophysical implications. *Journal of Geophysical Research : Solid Earth*, 128(3) :e2022JB025809.

[Penlou et al., 2023b] Penlou, B., Roche, O., and van den Wildenberg, S. (2023b). Experimental study of the generation of pore gas pressure in pyroclastic density currents resulting from eruptive fountain collapse. *Journal of Geophysical Research : Solid Earth*, 128(12) :e2023JB027510.

[Pérez and Dragicevic, 2010] Pérez, L. and Dragicevic, S. (2010). Exploring forest management practices using an agent-based model of forest insect infestations.

[Perkel, 2018] Perkel, J. M. (2018). Why jupyter is data scientists' computational notebook of choice. *Nature*, 563(7732) :145–147.

[Pevsner, 2015] Pevsner, J. (2015). *Bioinformatics and functional genomics*. John Wiley & Sons.

[Picaut and Simon, 2001] Picaut, J. and Simon, L. (2001). A scale model experiment for the study of sound propagation in urban areas. *Applied Acoustics*, 62(3) :327–340.

[Pimentel et al., 2019] Pimentel, J. F., Murta, L., Braganholo, V., and Freire, J. (2019). A large-scale study about quality and reproducibility of jupyter notebooks. In *2019 IEEE/ACM 16th international conference on mining software repositories (MSR)*, pages 507–517. IEEE.

[Pinto et al., 2020] Pinto, B., Henriques, A. C., Silva, P. M., and Bousbaa, H. (2020). Three-dimensional spheroids as in vitro preclinical models for cancer research. *Pharmaceutics*, 12(12) :1186.

[Pinto and Clevers, 2005] Pinto, D. and Clevers, H. (2005). Wnt control of stem cells and differentiation in the intestinal epithelium. *Experimental cell research*, 306(2) :357–363.

[Planck, 1901] Planck, M. (1901). On the law of distribution of energy in the normal spectrum. *Annalen der physik*, 4(553) :1.

[Planck, 1914] Planck, M. (1914). *The theory of heat radiation*. Blakiston.

[Popławski et al., 2008] Popławski, N. J., Shirinifard, A., Swat, M., and Glazier, J. A. (2008). Simulation of single-species bacterial-biofilm growth using the glazier-graner-hogeweg model and the compucell3d modeling environment. *Mathematical biosciences and engineering : MBE*, 5(2) :355.

[Postel et al., 2019] Postel, M., Karam, A., Pézeron, G., Schneider-Maunoury, S., and Clément, F. (2019). A multiscale mathematical model of cell dynamics during neurogenesis in the mouse cerebral cortex. *BMC bioinformatics*, 20 :1–24.

[Potdar et al., 2020] Potdar, A. M., Narayan, D., Kengond, S., and Mulla, M. M. (2020). Performance evaluation of docker container and virtual machine. *Procedia Computer Science*, 171 :1419–1428.

[Preen et al., 2019] Preen, R. J., Bull, L., and Adamatzky, A. (2019). Towards an evolvable cancer treatment simulator. *Biosystems*, 182 :1–7.

[Qi et al., 2009] Qi, H., Li, H., and Mynett, A. (2009). Exploring multi-agent systems in aquatic population dynamics modelling. In *7th ISE & 8th HIC, Conception, Chile*.

[Qu et al., 2014] Qu, Q., Li, D., Louis, K. R., Li, X., Yang, H., Sun, Q., Crandall, S. R., Tsang, S., Zhou, J., Cox, C. L., et al. (2014). High-efficiency motor neuron differentiation from human pluripotent stem cells and the function of islet-1. *Nature communications*, 5(1) :3449.

[Rakic, 2002] Rakic, P. (2002). Adult neurogenesis in mammals : an identity crisis. *Journal of Neuroscience*, 22(3) :614–618.

[Randles et al., 2017] Randles, B. M., Pasquetto, I. V., Golshan, M. S., and Borgman, C. L. (2017). Using the jupyter notebook as a tool for open science : An empirical study. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 1–2. IEEE.

[Rejniak and Anderson, 2011] Rejniak, K. A. and Anderson, A. R. (2011). Hybrid models of tumor growth. *Wiley Interdisciplinary Reviews : Systems Biology and Medicine*, 3(1) :115–125.

[Rendell, 2011] Rendell, P. (2011). A universal turing machine in conway's game of life. In *2011 International Conference on High Performance Computing & Simulation*, pages 764–772. IEEE.

[Reuillon et al., 2013] Reuillon, R., Leclaire, M., and Rey-Coyrehourcq, S. (2013). Openmole, a workflow engine specifically tailored for the distributed exploration of simulation models. *Future Generation Computer Systems*, 29(8) :1981–1990.

[Reynolds, 1987] Reynolds, C. W. (1987). Flocks, herds and schools : A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34.

[Rifes et al., 2020] Rifes, P., Isaksson, M., Rathore, G. S., Aldrin-Kirk, P., Møller, O. K., Barzaghi, G., Lee, J., Egerod, K. L., Rausch, D. M., Parmar, M., et al. (2020). Modeling neural tube development by differentiation of human embryonic stem cells in a microfluidic wnt gradient. *Nature biotechnology*, 38(11) :1265–1273.

[Rivière et al., 2016] Rivière, J., Ballet, P., and Rodin, V. (2016). Netbiodyn, a smart agent-based software to intuitively model and simulate complex biological systems. In *Conference on Complex Systems (CCS)*.

[Rodin et al., 2022] Rodin, V., Ballet, P., and Pottier, B. (2022). Application of artificial intelligent on agriculture : a cellular automata like multi-agents engine for agroecosystem simulation. In *IOP Conference Series : Earth and Environmental Science*, volume 1063, page 012060. IOP Publishing.

[Roques, 2013] Roques, L. (2013). Modèles de réaction-diffusion pour l'écologie spatiale : Avec exercices dirigés.

[Rosenberg, 1978] Rosenberg, A. (1978). The supervenience of biological concepts. *Philosophy of Science*, 45(3) :368–386.

[Rosenberg, 1985] Rosenberg, A. (1985). *The structure of biological science*. Cambridge University Press.

[Rosenblatt, 1957] Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386.

[Roy et al., 1983] Roy, P. K., Mamun, A., and Ahmed, G. (1983). In vitro acclimatization of aseptically cultured plantlets to humidity. *Plant Tissue Culture*, 14(2) :149–154.

[Rucker, 2005] Rucker, R. (2005). The lifebox, the seashell, and the soul. *Thunder's Mouth Press, New York*.

[Saade et al., 2013] Saade, M., Gutiérrez-Vallejo, I., Le Dreau, G., Rabadan, M. A., Miguez, D. G., Buceta, J., and Martí, E. (2013). Sonic hedgehog signaling switches the mode of division in the developing nervous system. *Cell reports*, 4(3) :492–503.

[Sarkar, 2000] Sarkar, P. (2000). A brief history of cellular automata. *Acm computing surveys (csur)*, 32(1) :80–107.

[Sarkar, 1992] Sarkar, S. (1992). Models of reduction and categories of reductionism. *Synthese*, 91 :167–194.

[Sayer, 2010] Sayer, A. (2010). Reductionism in social science. *Questioning nineteenth-century assumptions about knowledge II : Reductionism*, pages 5–56.

[Sayfullin et al., 2018] Sayfullin, S., Akhmetov, F., Mazzara, M., Mustafin, R., and Rivera, V. (2018). Gene expression for simulation of biological tissue. In *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 232–239. IEEE.

[Schaffner, 1967] Schaffner, K. F. (1967). Approaches to reduction. *Philosophy of science*, 34(2) :137–147.

[Schaller and Meyer-Hermann, 2006] Schaller, G. and Meyer-Hermann, M. (2006). Continuum versus discrete model : a comparison for multicellular tumour spheroids. *Philosophical Transactions of the Royal Society A : Mathematical, Physical and Engineering Sciences*, 364(1843) :1443–1464.

[Schelling, 1971] Schelling, T. C. (1971). Dynamic models of segregation. *Journal of mathematical sociology*, 1(2) :143–186.

[Schreiter et al., 2011] Schreiter, J., Meyer, S., Schmidt, C., Schulz, R. M., and Langer, S. (2011). Dorsal skinfold chamber models in mice. *GMS Interdisciplinary plastic and reconstructive surgery DGPW*, 6.

[Schroeder et al., 1996] Schroeder, W. J., Martin, K. M., and Lorensen, W. E. (1996). The design and implementation of an object-oriented toolkit for 3d graphics and visualization. In *Proceedings of Seventh Annual IEEE Visualization'96*, pages 93–100. IEEE.

[Schulman and Seiden, 1978] Schulman, L. and Seiden, P. (1978). Statistical mechanics of a dynamical system based on conway's game of life. *Journal of Statistical Physics*, 19 :293–314.

[Scianna and Preziosi, 2013] Scianna, M. and Preziosi, L. (2013). *Cellular potts models : multiscale extensions and biological applications*. CRC Press.

[Scianna et al., 2012] Scianna, M., Preziosi, L., and Wolf, K. (2012). A cellular potts model simulating cell migration on and in matrix environments. *Mathematical Biosciences & Engineering*, 10(1) :235–261.

[Sego et al., 2023a] Sego, T., Comlekoglu, T., Peirce, S. M., Desimone, D. W., and Glazier, J. A. (2023a). General, open-source vertex modeling in biological applications using tissue forge. *Scientific Reports*, 13(1) :17886.

[Sego et al., 2023b] Sego, T., Sluka, J. P., Sauro, H. M., and Glazier, J. A. (2023b). Tissue forge : interactive biological and biophysics simulation environment. *PLOS Computational Biology*, 19(10) :e1010768.

[Sengupta and Sarkar, 2003] Sengupta, D. L. and Sarkar, T. K. (2003). Maxwell, hertz, the maxwellians, and the early history of electromagnetic waves. *IEEE Antennas and Propagation Magazine*, 45(2) :13–19.

[Shamil et al., 2021] Shamil, M. S., Farheen, F., Ibtehaz, N., Khan, I. M., and Rahman, M. S. (2021). An agent-based modeling of covid-19 : validation, analysis, and recommendations. *Cognitive computation*, pages 1–12.

[Shazadi, 2021] Shazadi, K. (2021). Decision tree in biology. *European Journal of Biology*, 6(1) :1–15.

[Shechter et al., 2005] Shechter, S. M., Bryce, C. L., Alagoz, O., Kreke, J. E., Stahl, J. E., Schaefer, A. J., Angus, D. C., and Roberts, M. S. (2005). A clinically based discrete-event simulation of end-stage liver disease and the organ allocation process. *Medical Decision Making*, 25(2) :199–209.

[Sherwood and Toliver-Kinsky, 2004] Sherwood, E. R. and Toliver-Kinsky, T. (2004). Mechanisms of the inflammatory response. *Best Practice & Research Clinical Anaesthesiology*, 18(3) :385–405.

[Shinde and Kurhekar, 2020] Shinde, S. B. and Kurhekar, M. P. (2020). Agent-based modeling of the adaptive immune system using netlogo simulation tool. In *Soft Computing for Problem Solving : SocProS 2018, Volume 2*, pages 463–474. Springer.

[Silva et al., 2020] Silva, P. C., Batista, P. V., Lima, H. S., Alves, M. A., Guimarães, F. G., and Silva, R. C. (2020). Covid-abs : An agent-based model of covid-19 epidemic to simulate health and economic effects of social distancing interventions. *Chaos, Solitons & Fractals*, 139 :110088.

[Sindhu et al., 2020] Sindhu, V., Nivedha, S., and Prakash, M. (2020). An empirical science research on bioinformatics in machine learning. *Journal of Mechanics of Continua and Mathematical Sciences*, 7 :86–94.

[Slepchenko and Loew, 2010] Slepchenko, B. M. and Loew, L. M. (2010). Use of virtual cell in studies of cellular dynamics. *International review of cell and molecular biology*, 283 :1–56.

[Smith et al., 1970] Smith, D. W., Schaller, H. E., and Bonhoeffer, F. J. (1970). Dna synthesis in vitro. *Nature*, 226(5247) :711–713.

[Smith, 2005] Smith, S. E. (2005). Topographic mapping. *Environmental soil-landscape modeling : geographic information technologies and pedometrics*, 1 :155–182.

[Soldatow et al., 2013] Soldatow, V. Y., LeCluyse, E. L., Griffith, L. G., and Rusyn, I. (2013). In vitro models for liver toxicity testing. *Toxicology research*, 2(1) :23–39.

[Spector and Willoughby, 1963] Spector, W. and Willoughby, D. (1963). The inflammatory response. *Bacteriological Reviews*, 27(2) :117–154.

[Stamataki et al., 2005] Stamataki, D., Ulloa, F., Tsoni, S. V., Mynett, A., and Briscoe, J. (2005). A gradient of gli activity mediates graded sonic hedgehog signaling in the neural tube. *Genes & development*, 19(5) :626–641.

[Stanojevic et al., 2018] Stanojevic, R., Abbar, S., Thirumuruganathan, S., De Francisci Morales, G., Chawla, S., Filali, F., and Aleimat, A. (2018). Road network fusion for incremental map updates. In *Progress in Location Based Services 2018 14*, pages 91–109. Springer.

[Starruß et al., 2014] Starruß, J., De Back, W., Brusch, L., and Deutsch, A. (2014). Morpheus : a user-friendly modeling environment for multiscale and multicellular systems biology. *Bioinformatics*, 30(9) :1331–1332.

[Stolarek et al., 2022] Stolarek, I., Samelak-Czajka, A., Figlerowicz, M., and Jackowiak, P. (2022). Dimensionality reduction by umap for visualizing and aiding in classification of imaging flow cytometry data. *Iscience*, 25(10).

[Sütterlin et al., 2009] Sütterlin, T., Huber, S., Dickhaus, H., and Grabe, N. (2009). Modeling multi-cellular behavior in epidermal tissue homeostasis via finite state machines in multi-agent systems. *Bioinformatics*, 25(16) :2057–2063.

[Suzuki, 2011] Suzuki, K. (2011). *Artificial neural networks : methodological advances and biomedical applications*. BoD–Books on Demand.

[Swat et al., 2009] Swat, M. H., Hester, S. D., Balter, A. I., Heiland, R. W., Zaitlen, B. L., and Glazier, J. A. (2009). Multicell simulations of development and disease using the compucell3d simulation environment. *Systems Biology*, pages 361–428.

[Swat et al., 2012] Swat, M. H., Thomas, G. L., Belmonte, J. M., Shirinifard, A., Hmeljak, D., and Glazier, J. A. (2012). Chapter 13 - Multi-Scale Modeling of Tissues Using CompuCell3D. In Asthagiri, A. R. and Arkin, A. P., editors, *Methods in Cell Biology*, volume 110 of *Computational Methods in Cell Biology*, pages 325–366. Academic Press. ZSCC : NoCitationData[s1].

[Swierniak et al., 2009] Swierniak, A., Kimmel, M., and Smieja, J. (2009). Mathematical modeling as a tool for planning anticancer therapy. *European journal of pharmacology*, 625(1-3) :108–121.

[Szabó and Merks, 2013] Szabó, A. and Merks, R. M. (2013). Cellular potts modeling of tumor growth, tumor invasion, and tumor evolution. *Frontiers in oncology*, 3 :87.

[Taillandier et al., 2019] Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.-N., Marilleau, N., Caillou, P., Philippon, D., and Drogoul, A. (2019). Building, composing and experimenting complex spatial models with the gama platform. *GeoInformatica*, 23 :299–322.

[Tamashiro et al., 2003] Tamashiro, K. L., Wakayama, T., Yamazaki, Y., Akutsu, H., Woods, S. C., Kondo, S., Yanagimachi, R., and Sakai, R. R.

(2003). Phenotype of cloned mice : development, behavior, and physiology. *Experimental Biology and Medicine*, 228(10) :1193–1200.

[Tarca et al., 2007] Tarca, A. L., Carey, V. J., Chen, X.-w., Romero, R., and Drăghici, S. (2007). Machine learning and its applications to biology. *PLoS computational biology*, 3(6) :e116.

[Thanachareonkit et al., 2005] Thanachareonkit, A., Scartezzini, J.-L., and Andersen, M. (2005). Comparing daylighting performance assessment of buildings in scale models and test modules. *Solar Energy*, 79(2) :168–182.

[Thomas, 1765] Thomas, B. (1765). An essay towards solving a problem in the doctrine of chances. *Biometrika*, 45 :296–315.

[Thomson, 2023] Thomson, T. (2023). *The history of chemistry*. Good Press.

[Thorne et al., 2007] Thorne, B. C., Bailey, A. M., and Peirce, S. M. (2007). Combining experiments with multi-cell agent-based modeling to study biological tissue patterning. *Briefings in bioinformatics*, 8(4) :245–257.

[Tisue and Wilensky, 2004] Tisue, S. and Wilensky, U. (2004). Netlogo : A simple environment for modeling complexity. In *International conference on complex systems*, volume 21, pages 16–21. Citeseer.

[Travers et al., 1970] Travers, A., Kamen, R., and Cashel, M. (1970). The in vitro synthesis of ribosomal rna. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 35, pages 415–418. Cold Spring Harbor Laboratory Press.

[Turing, 1952] Turing, A. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London Series B*, 237(641) :37–72.

[Turnbull, 1996] Turnbull, D. (1996). Cartography and science in early modern europe : Mapping the construction of knowledge spaces. *Imago mundi*, 48(1) :5–24.

[Uhlenbeck and Ornstein, 1930] Uhlenbeck, G. E. and Ornstein, L. S. (1930). On the theory of the brownian motion. *Physical review*, 36(5) :823.

[Ulanowicz, 2001] Ulanowicz, R. E. (2001). The organic in ecology. *Ludus Vitalis*, 9(15) :183–204.

[Vadillo, 2019] Vadillo, F. (2019). Comparing stochastic lotka–volterra predator-prey models. *Applied Mathematics and Computation*, 360 :181–189.

[van den Buuse, 2010] van den Buuse, M. (2010). Modeling the positive symptoms of schizophrenia in genetically modified mice : pharmacology and methodology aspects. *Schizophrenia bulletin*, 36(2) :246–270.

[Van Liedekerke, 2019] Van Liedekerke, P. (2019). *Quantitative modeling of cell and tissue mechanics with agent-based models*. PhD thesis, Inria Paris, Sobonne Université.

[Van Liedekerke et al., 2018] Van Liedekerke, P., Buttenschön, A., and Drasdo, D. (2018). Off-lattice agent-based models for cell and tumor growth : numerical methods, implementation, and applications. In *Numerical methods and advanced simulation in biomechanics and biological processes*, pages 245–267. Elsevier.

[Van Liedekerke et al., 2015] Van Liedekerke, P., Palm, M., Jagiella, N., and Drasdo, D. (2015). Simulating tissue mechanics with agent-based models : concepts, perspectives and some novel results. *Computational particle mechanics*, 2(4) :401–444.

[van Wijk et al., 2016] van Wijk, R. C., Krekels, E. H., Hankemeier, T., Spaink, H. P., and van der Graaf, P. H. (2016). Systems pharmacology of hepatic metabolism in zebrafish larvae. *Drug Discovery Today : Disease Models*, 22 :27–34.

[VanDussen et al., 2012] VanDussen, K. L., Carulli, A. J., Keeley, T. M., Patel, S. R., Puthoff, B. J., Magness, S. T., Tran, I. T., Maillard, I., Siebel, C., Kolterud, Å., et al. (2012). Notch signaling modulates proliferation and differentiation of intestinal crypt base columnar stem cells. *Development*, 139(3) :488–497.

[Verma et al., 2021] Verma, S., Pant, M., and Snasel, V. (2021). A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems. *IEEE access*, 9 :57757–57791.

[Verschuren, 2001] Verschuren, P. J. (2001). Holism versus reductionism in modern social science research. *Quality and Quantity*, 35 :389–405.

[Vittadello et al., 2021] Vittadello, S. T., Leyshon, T., Schnoerr, D., and Stumpf, M. P. (2021). Turing pattern design principles and their robustness. *Philosophical Transactions of the Royal Society A*, 379(2213) :20200272.

[Vodovotz et al., 2006] Vodovotz, Y., Chow, C. C., Bartels, J., Lagoa, C., Prince, J. M., Levy, R. M., Kumar, R., Day, J., Rubin, J., Constantine, G., et al. (2006). In silico models of acute inflammation in animals. *Shock*, 26(3) :235–244.

[Vodovotz et al., 2004] Vodovotz, Y., Clermont, G., Chow, C., and An, G. (2004). Mathematical models of the acute inflammatory response. *Current opinion in critical care*, 10(5) :383–390.

[Voinov et al., 2018] Voinov, A., Jenni, K., Gray, S., Kolagani, N., Glynn, P. D., Bommel, P., Prell, C., Zellner, M., Paolisso, M., Jordan, R., et al. (2018). Tools and methods in participatory modeling : Selecting the right tool for the job. *Environmental Modelling & Software*, 109 :232–255.

[Volterra, 1926] Volterra, V. (1926). Fluctuations in the abundance of a species considered mathematically. *Nature*, 118(2972) :558–560.

[von Neumann, 1951] von Neumann, J. (1951). The general and logical theory of automata.

[VP, 1956] VP, C. (1956). Observations on growth rates of human tumors. *AJR*, 76 :988–1000.

[Wakayama et al., 1999] Wakayama, T., Rodriguez, I., Perry, A. C., Yanagimachi, R., and Mombaerts, P. (1999). Mice cloned from embryonic stem cells. *Proceedings of the National Academy of Sciences*, 96(26) :14984–14989.

[Walsh et al., 2021] Walsh, I., Fishman, D., Garcia-Gasulla, D., Titma, T., Pollastri, G., Harrow, J., Psomopoulos, F. E., and Tosatto, S. C. (2021). Dome :

recommendations for supervised machine learning validation in biology. *Nature methods*, 18(10) :1122–1127.

[Wang and Uhlenbeck, 1945] Wang, M. C. and Uhlenbeck, G. E. (1945). On the theory of the brownian motion ii. *Reviews of modern physics*, 17(2-3) :323.

[Wang et al., 2015] Wang, Z., Butner, J. D., Kerketta, R., Cristini, V., and Deisboeck, T. S. (2015). Simulating cancer growth with multiscale agent-based modeling. In *Seminars in cancer biology*, volume 30, pages 70–78. Elsevier.

[Wanjek, 2011] Wanjek, C. (2011). Systems biology as defined by nih : an intellectual resource for integrative biology. *The NIH Catalyst*, 19(6) :1–12.

[Wardle et al., 2004] Wardle, K., Dobbs, E. B., and Short, K. C. (2004). In vitro plantlets regeneration of rose. *Journal of the American Society for Horticultural Science*, 108(3) :386–389.

[Waskom, 2021] Waskom, M. L. (2021). Seaborn : statistical data visualization. *Journal of Open Source Software*, 6(60) :3021.

[Watson and Crick, 1953] Watson, J. D. and Crick, F. H. (1953). The structure of dna. In *Cold Spring Harbor symposia on quantitative biology*, volume 18, pages 123–131. Cold Spring Harbor Laboratory Press.

[Weisz and Argibay, 2009] Weisz, V. I. and Argibay, P. F. (2009). A putative role for neurogenesis in neurocomputational terms : inferences from a hippocampal model. *Cognition*, 112(2) :229–240.

[Westine et al., 2012] Westine, P. S., Dodge, F. T., and Baker, W. (2012). *Similarity methods in engineering dynamics : theory and practice of scale modeling*. Elsevier.

[Weymouth and Loeb, 1978] Weymouth, L. A. and Loeb, L. A. (1978). Mutagenesis during in vitro dna synthesis. *Proceedings of the National Academy of Sciences*, 75(4) :1924–1928.

[Wilensky and Rand, 2015] Wilensky, U. and Rand, W. (2015). *An introduction to agent-based modeling : modeling natural, social, and engineered complex systems with NetLogo*. Mit Press.

[Wilhelm et al., 2011] Wilhelm, I., Fazakas, C., and Krizbai, I. (2011). In vitro models of the blood-brain barrier. *Acta neurobiologiae experimentalis*, 71(1) :113–128.

[Wimsatt, 1976] Wimsatt, W. C. (1976). Reductionism, levels of organization, and the mind-body problem. In *Consciousness and the brain : A scientific and philosophical inquiry*, pages 205–267. Springer.

[Wolfram, 2020] Wolfram, C. (2020). An agent-based model of covid-19. *Complex Systems*, 29(1).

[Woo et al., 2010] Woo, P. C., Huang, Y., Lau, S. K., and Yuen, K.-Y. (2010). Coronavirus genomics and bioinformatics analysis. *viruses*, 2(8) :1804–1820.

[Worzel et al., 2009] Worzel, W. P., Yu, J., Almal, A. A., and Chinnaiyan, A. M. (2009). Applications of genetic programming in cancer research. *The international journal of biochemistry & cell biology*, 41(2) :405–413.

[Xing et al., 2017] Xing, L., Guo, M., Liu, X., Wang, C., Wang, L., and Zhang, Y. (2017). An improved bayesian network method for reconstructing gene regulatory network based on candidate auto selection. *BMC genomics*, 18 :17–30.

[Yeol et al., 2005] Yeol, J., Barjis, I., Ryu, Y., and Barjis, J. (2005). Discrete-event based simulation conceptual modeling of systems biology. In *5th IEEE Conference on Nanotechnology, 2005.*, pages 160–163. IEEE.

[Yim et al., 2018] Yim, A., Chung, C., and Yu, A. (2018). *Matplotlib for Python Developers : Effective techniques for data visualization with Python.* Packt Publishing Ltd.

[Yogya and Kosala, 2014] Yogya, R. and Kosala, R. (2014). Comparison of physics frameworks for webgl-based game engine. In *EPJ Web of Conferences*, volume 68, page 00035. EDP Sciences.

[Young, 1802] Young, T. (1802). Ii. the bakerian lecture. on the theory of light and colours. *Philosophical transactions of the Royal Society of London*, (92) :12–48.

[Younossi-Hartenstein et al., 1996] Younossi-Hartenstein, A., Nassif, C., Green, P., and Hartenstein, V. (1996). Early neurogenesis of the drosophila brain. *Journal of Comparative Neurology*, 370(3) :313–329.

[Yu et al., 2019] Yu, Z., Bai, C., Seinturier, L., and Monperrus, M. (2019). Characterizing the usage, evolution and impact of java annotations in practice. *IEEE Transactions on Software Engineering*, 47(5) :969–986.

[Yusoff et al., 2011] Yusoff, Y., Ngadiman, M. S., and Zain, A. M. (2011). Overview of nsga-ii for optimizing machining process parameters. *Procedia Engineering*, 15 :3978–3983.

[Zawidzki, 2011] Zawidzki, M. (2011). Application of semitotalistic 2d cellular automata on a triangulated 3d surface. *International Journal of Design & Nature and Ecodynamics*, 6(1) :34–51.

[Zeilinger et al., 2016] Zeilinger, K., Freyer, N., Damm, G., Seehofer, D., and Knöspel, F. (2016). Cell sources for in vitro human liver cell culture models. *Experimental Biology and Medicine*, 241(15) :1684–1698.

[Zhang and DeAngelis, 2020] Zhang, B. and DeAngelis, D. L. (2020). An overview of agent-based models in plant biology and ecology. *Annals of Botany*, 126(4) :539–557.

[Zhang et al., 2019] Zhang, W., Jang, S., Jonsson, C. B., and Allen, L. J. (2019). Models of cytokine dynamics in the inflammatory response of viral zoonotic infectious diseases. *Mathematical medicine and biology : a journal of the IMA*, 36(3) :269–295.

[Zhou et al., 2011] Zhou, Y., Liepe, J., Sheng, X., Stumpf, M. P., and Barnes, C. (2011). Gpu accelerated biochemical network simulation. *Bioinformatics*, 27(6) :874–876.

[Zhu et al., 2009] Zhu, J., Zhang, Y.-T., Newman, S. A., and Alber, M. (2009). Application of discontinuous galerkin methods for reaction-diffusion systems in developmental biology. *Journal of Scientific Computing*, 40 :391–418.

[Ziebarth et al., 2013] Ziebarth, J. D., Bhattacharya, A., and Cui, Y. (2013). Bayesian network webserver : a comprehensive tool for biological network modeling. *Bioinformatics*, 29(21) :2801–2803.

[Zou and Conzen, 2005] Zou, M. and Conzen, S. D. (2005). A new dynamic bayesian network (dbn) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21(1) :71–79.

[Zubler and Douglas, 2009] Zubler, F. and Douglas, R. (2009). A framework for modeling the growth and development of neurons and networks. *Frontiers in computational neuroscience*, 3 :757.

[Zupanc, 2006] Zupanc, G. (2006). Neurogenesis and neuronal regeneration in the adult fish brain. *Journal of Comparative Physiology A*, 192 :649–670.

[Zupanc, 2001] Zupanc, G. K. (2001). Adult neurogenesis and neuronal regeneration in the central nervous system of teleost fish. *Brain behavior and evolution*, 58(5) :250–275.

[Zupanc, 2008] Zupanc, G. K. (2008). Adult neurogenesis and neuronal regeneration in the brain of teleost fish. *Journal of physiology-paris*, 102(4-6) :357–373.