



Modélisation et simulation de colonies de bactéries et de biofilms.

14 mai 2023

Code PSC : "MEC15"

Coordinateur : Laurence Bodelot

Tuteur : Jules Dichamp

Membres : DABERT Darius, DUBREUIL Ulysse, BARLOY Renaud, IFRAH Ruben, SOUDA Matthieu



SOMMAIRE

1	Introduction	3
2	Motivations	3
2.1	Revue de l'état de l'art	3
2.1.1	Choix du modèle	3
2.1.2	Mécanique de déplacement de la cellule	4
2.1.3	Division de la cellule et accroissement de l'agrégat	4
2.2	Spécificité de la bactérie	5
2.2.1	Implémentation numérique	6
2.3	Objectifs	6
3	Cadre du projet	6
3.1	Organisation générale	6
3.2	Structure du travail de modélisation : Gitlab	7
3.3	Structure du travail d'implémentation : CompuTiX	7
4	Modélisation des phénomènes et interactions	8
4.1	idée générale du modèle	8
4.2	Modèle effectivement codé	8
4.2.1	Croissance	9
4.2.2	Division	9
4.2.3	Contact adhésif et répulsif entre les cellules	12
4.3	Parties du modèle prévues mais non implémentées	13
4.3.1	Mort des cellules	13
4.3.2	Contact avec le substrat	14
4.3.3	Frottements	16
5	Implémentation de la modélisation en C++	17
5.1	Implémentation des modèles	17
5.2	Production d'une simulation	21
6	Résultats et exploitation	24
7	Conclusion	25
8	Références	26
9	Annexe	27

1

INTRODUCTION

L'étude de la modélisation et de la simulation de colonies de bactéries est un domaine de recherche scientifique qui suscite un intérêt croissant. En effet, ces organismes unicellulaires sont omniprésents dans notre environnement, et jouent un rôle crucial dans des processus biologiques tels que la décomposition des matières organiques, la fermentation, la digestion et la production d'antibiotiques.

La modélisation et la simulation de colonies de bactéries permettent de mieux comprendre les mécanismes qui régissent leur comportement individuel et collectif. Cela peut être particulièrement utile pour développer de nouvelles méthodes de lutte contre les infections bactériennes, ou pour optimiser des processus industriels tels que la production d'enzymes ou de médicaments.

De plus, les enjeux liés à la modélisation et à la simulation de colonies de bactéries sont nombreux et variés. D'un point de vue scientifique, cette approche permet de mieux comprendre les mécanismes de communication et de coordination entre les cellules au sein d'une colonie. Elle peut également aider à élucider les facteurs qui influencent la croissance et la survie des bactéries.

Du point de vue de la médecine, la modélisation et la simulation de colonies de bactéries peuvent être utilisées pour étudier les mécanismes de résistance aux antibiotiques, ou pour développer de nouvelles stratégies thérapeutiques. En outre, cette approche peut être appliquée dans le domaine de l'agriculture pour optimiser les processus de fermentation, ou pour améliorer la qualité des sols.

2

MOTIVATIONS

Dans le cadre de ces applications multiples, notre projet scientifique collectif a pour objectif de modéliser, puis implémenter des colonies de cellules sur un substrat, en prenant en compte tant le comportement intrinsèque des cellules (croissance et division), que leurs interactions entre elles et avec le substrat.

2.1 REVUE DE L'ÉTAT DE L'ART

2.1.1 • CHOIX DU MODÈLE

Notre étude portera sur la modélisation mécanique de bactéries, mais se fondera en grande partie sur les études déjà menées par l'INRIA sur le comportement mécanique des cellules. Plusieurs approches et modèles différents existent pour décrire le comportement des cellules. Les chercheurs de l'INRIA ont choisi de s'intéresser aux modèles dont la description des interactions entre cellules et avec la matrice extracellulaire repose sur l'action des forces (par opposition aux énergies et potentiels mis en jeu)[1]. C'est un moyen plus intuitif de modéliser ces interactions et cela donne l'avantage d'avoir une échelle de temps mieux définie, puisqu'on cherche alors à résoudre les équations du mouvement de chaque cellule (celles données par la deuxième loi de Newton). Il existe également un modèle appelé "déformable cells model" [5] qui représente chaque cellule par un ensemble de noeuds reliés par une membrane visco-élastique. Ce modèle permet toutefois de décrire de manière détaillée la forme des cellules et les déformations complexes qu'elles pourraient subir. Cependant, du fait de sa complexité, il limite le nombre de cellules représentables dans la simulation. C'est pourquoi nous retiendrons plutôt le "center-based model" [2], dans lequel les cellules sont représentées par de simples sphères, leur position décrite

par leur centre, dont on cherche à connaître la trajectoire en résolvant l'équation de mouvement. On considère que la cellule suit un mouvement brownien au cours duquel elle est en interaction avec les autres cellules et le milieu extracellulaire.

2.1.2 • MÉCANIQUE DE DÉPLACEMENT DE LA CELLULE

Dans le "center-based model", l'équation de base est donc une équation de Langevin (associé à un processus stochastique). Pour une cellule i on écrit :

$$\gamma V_i(t) + \sum_j \Gamma_{ij}(V_i - V_j) = \sum_j F_{ij} + \zeta_i(t)$$

- $\gamma V_i(t)$ est la force de frottement avec le milieu extracellulaire.

- $\sum_j \Gamma_{ij}(V_i - V_j)$ est l'ensemble des forces de frottements entre cellules.

$-\sum_j F_{ij}$ est l'ensemble des forces d'adhésion/répulsion entre les cellules et avec le milieu extracellulaire. Dans le modèle, elles sont prises comme des forces de contact de Hertz. [2]

- ζ représente le processus aléatoire de migration sous la forme d'une force de migration stochastique

Le mouvement de la cellule est donc régi par les forces qu'elle subit en interaction avec le milieu et les autres cellules ainsi que son comportement intrinsèquement aléatoire.

2.1.3 • DIVISION DE LA CELLULE ET ACCROISSEMENT DE L'AGRÉGAT

Le processus de division est fondamental dans la croissance d'une population de cellules. Ce processus de multiplication cellulaire peut être modélisé de nombreuses façons. Nous nous intéresserons ici à deux approches dans le cadre d'une représentation sphérique des cellules [4]. Ce processus doit notamment respecter la conservation du volume lors de l'étape de la mitose (processus de division cellulaire en deux cellules-filles identiques).

La division cellulaire constitue un cycle, qui se répète de manière périodique (de période moyenne τ) au cours du temps.

Le premier modèle étudié (références ??) considère deux entités cellulaires dès le début de l'interphase (étape de la mitose ou la cellule se scinde en 2). D'abord possédant le même centre, la distance séparant les deux centres augmente jusqu'à atteindre le diamètre d'une cellule. Cette méthode nécessite un pas d'incrémentation de τ .

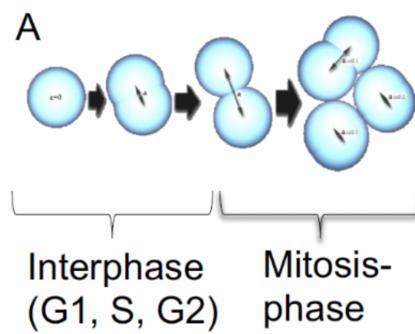


FIGURE 1 – Premier modèle

Le second modèle consiste à ne considérer qu'une cellule durant l'interphase. Le volume de cette sphère augmente jusqu'à atteindre deux fois celle de la cellule d'origine. Cette cellule est alors supprimée et remplacée

par deux de volume moitié, tangente l'une de l'autre. Nous choisirons d'utiliser ce second modèle dans notre étude car pour le "center-based model", la complexité algorithmique est optimale (moins de temps et de mémoire nécessaire pour un résultat similaire) et l'on peut par conséquent effectuer des simulations plus réalistes.

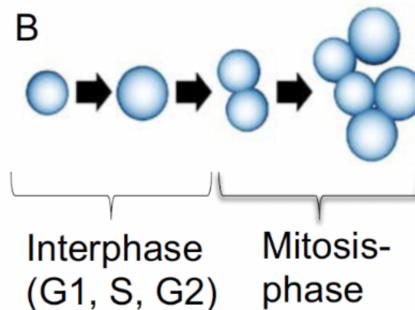


FIGURE 2 – Second modèle

2.2 SPÉCIFICITÉ DE LA BACTÉRIE

La modélisation des bactéries diffère de celle des cellules du fait de la forme caractéristique qu'a la bactérie, et des forces d'interaction légèrement différentes. Les modélisations effectuées sur les bactéries traitent des différents mécanismes homéostatiques agissant sur la croissance et la division des bactéries. En outre, elles expliquent comment ces mécanismes peuvent affecter la forme de la colonie de bactéries (ou biofilm).

Dans ces modélisations, les bactéries sont représentées par des capsules (un rectangle de longueur croissant linéairement avec le temps entouré de deux demi-cercles fixes) et évoluent dans un environnement à deux dimensions.

La division des bactéries est régie par une combinaison de deux mécanismes limites :

- La bactérie se divise en 2 quand elle a atteint une taille fixée (SIZER)
- La bactérie se divise en 2 quand sa taille a augmenté d'une longueur fixée (ADDER). [4]

La position et l'orientation des bactéries dépendent de forces et de couples, soient d'interactions entre les bactéries les plus proches, soient stochastiques. (principe fondamental de la dynamique)

Le système de bactéries croît donc exponentiellement, et est aussi assujetti à une certaine diffusion des bactéries dans le milieu, semblable à une diffusion particulaire. Deux temps caractéristiques sont alors introduits, t_{dif} et t_{gr} (pour diffusion et growth), ainsi que leur rapport : $\Gamma = t_{dif}/t_{gr}$ [4]

Différents jeux de variables et de mécanismes ont été utilisés pour cette modélisation, et les résultats observés portaient essentiellement sur la densité de biomasse (le nombre de bactéries par unité de surface du biofilm) et la cohérence d'orientation des bactéries (grandeur caractérisant la propension des bactéries dont les orientations sont selon le même axe).

Les résultats ont montré que les mécanismes de division SIZER et ADDER ont peu d'impact sur les propriétés des biofilms, contrairement au rapport de temps Γ , qui contrôle presque intégralement le comportement du biofilm ($\Gamma = 15$: colonie compacte, croissance rapide et orientation cohérente, $\Gamma = 10^{-2}$: colonie peu dense et non cohérente, croissance plus lente).[4]

$\Gamma = 15$	$\Gamma = 10^{-2}$
Colonie compacte	Colonie éparpillée dans l'espace
orientation cohérente	orientation aléatoire
croissance rapide	croissance lente

Cette expérience nous donne un premier modèle de multiplication de bactéries à exploiter, ainsi que des résultats préliminaires soulignant les paramètres les plus importants à mettre en place dans nos modélisations.

2.2.1 • IMPLÉMENTATION NUMÉRIQUE

Le logiciel développé par l'équipe de l'INRIA, codé en C++, est basé sur une approche dite IBM (« Individual Based Model »)[5], où chaque bactérie ou cellule sera représentée individuellement par un objet dans le code. Le temps est discrétisé, et le déplacement des cellules entre ces intervalles de temps discret se calcule par principe fondamental de la dynamique appliquée à chaque cellule, incluant plus ou moins de forces et d'interactions détaillées avec le milieu et les autres cellules. L'utilisation de la programmation orientée objet permet une grande modularité et flexibilité du code. En effet, chaque élément important du modèle (forces, individus, milieu) est représenté par une classe, et cela permet de changer à volonté les paramètres du modèle et constater leurs influences sur la simulation. Les parties du code concernant la résolution des équations du mouvement étant déjà codées, notre principal travail sera de coder les caractéristiques des forces d'interactions dans le cas de cellules.

2.3 OBJECTIFS

Ce PSC a pour premier objectif d'exploiter les modèles proposés dans l'état de l'art, et de les modifier afin d'obtenir une modélisation fiable d'une colonie de cellules.

Le deuxième objectif de ce PSC est d'implémenter ces modèles en C++ afin de remplir deux critères :

- La correction des algorithmes du modèle proposé. Le programme doit permettre de représenter les colonies de cellules conformément au modèle.
- Une structuration du code lui permettant de s'intégrer correctement aux modélisations pré-établies de l'INRIA, afin d'avoir une cohérence dans l'ensemble des projets.

3 CADRE DU PROJET

3.1 ORGANISATION GÉNÉRALE

Notre PSC s'est fait en partenariat avec l'INRIA. Nous étions en contact avec deux tuteurs y travaillant, Jules Dichamp et Jiri Pesek. L'entièreté du projet a été mené en anglais, afin de s'inscrire dans la continuité des projets similaires précédemment effectués, et dans un souci de compréhension, car l'un de nos tuteurs n'était pas francophone.

Dès le départ, nous avons décidé de nous diviser en deux équipes : l'une composée de Ruben Ifrah, Renaud Barloy et Matthieu Souda chargée de modéliser l'ensemble du modèle, et l'autre, Darius Dabert et Ulysse Dubreuil, chargée de coder ce modèle en C++.

Chaque équipe travaille en autonomie sur Gitlab, et les équipes se réunissent avec le tuteur lors des créneaux dédiés au PSC.

Cette structure de travail très compartimentée nous a permis une répartition claire et instinctive des tâches tout au long de notre projet. Elle nécessite cependant une communication exemplaire à la fois au sein des équipes, entre les équipes et avec les tuteurs. De plus, le modèle n'était pas établi au début du projet, ce qui, couplé à des problèmes d'accès à la plateforme de code, a créé des hétérogénéités de travail entre équipes durant l'année.

En effet, l'équipe de modélisation fut plus sollicitée en début de projet, et celle d'implémentation en fin de projet.

Les premiers mois ont permis de modéliser les comportements de cellules (croissance et division), ainsi que d'apprendre à programmer de manière professionnelle en C++ malgré les contraintes liées à l'accord de confidentialité avec l'INRIA. Par la suite, nous avons affiné le modèle, modélisé les interactions entre cellules et commencé l'implémentation.

3.2 STRUCTURE DU TRAVAIL DE MODÉLISATION : GITLAB

La majorité de notre activité s'est faite sur Gitlab, un espace collaboratif nous permettant d'actualiser notre travail en temps réel, et d'avoir une structure claire du travail réalisé ainsi que du travail qu'il reste à réaliser.

Le projet était divisé sur Gitlab en "Milestones", représentant les propriétés principales de la colonie : l'une concernant le cycle de vie de la cellule, deux autres sur les interactions entre cellules et entre les cellules et le substrat. Ces mêmes "Milestones" sont séparées en "Issues", chaque Issue traitant une tâche spécifique à modéliser, puis à coder.

Pour l'équipe chargée de la modélisation, le travail consistait à étudier la bibliographie des modèles préexistants, et d'en tirer des modèles viables pour chaque détail de la modélisation.

Après avoir complété une issue, il a également fallu, pour chacune, écrire différents tests à coder, ainsi que les résultats théoriquement attendus, afin de vérifier la correction des algorithmes et la stabilité de variables conservées (voir si le volume se conserve lors des divisions par exemple). Ces tests sont effectués à chaque fois pour des situations spécifiques, d'où leur nom dans la modélisation de "test cases". L'objectif de ces tests est de couvrir un maximum de situations susceptibles d'arriver dans la simulation, afin d'avoir le programme le plus fonctionnel possible en sortie de test.

(L'objectif de ces tests est de s'assurer

L'équipe de programmation s'occupe ensuite d'implémenter ces modèles, afin d'obtenir au final une simulation informatique de la prolifération d'une colonie de cellules.

Cette méthode rigoureuse et séquencée du travail permet encore une fois une vision claire des objectifs à remplir tout au long du projet. D'une idée globale de modèle, nous avons défini 3 grands axes, puis, pour chacun, des tâches mineures représentant une ou deux pages de description et d'équations régissant le modèle. Cela a permis de séparer aisément les tâches, chaque membre de l'équipe travaillant sur une Issue différente.

Cette façon de travailler nous a cependant surpris au début de notre projet, car une telle séparation des tâches nécessite une grande rigueur, et une connaissance précise des différentes Issues. L'une des contraintes majeures était notamment l'uniformité des notations entre chaque Issue. En effet, pour chaque nouvelle étape du modèle, il était nécessaire de prendre en compte les notations cohérentes déjà établies, de vérifier leurs expressions, voire parfois de les adapter afin de préserver la cohérence du modèle. Cette nécessité de vérification nous a contraints à reprendre régulièrement les différentes Issues de base.

Il nous a donc fallu nous adapter à ce mode de travail inédit pour nous, afin de conserver l'efficacité visée par cette structuration des tâches.

3.3 STRUCTURE DU TRAVAIL D'IMPLÉMENTATION : COMPUTIX

Pour l'équipe de programmation, le développement a commencé avec un certain retard, en partie causé par les difficultés d'accès au code de l'INRIA liées à la signature de NDA, et à l'installation sur nos propres ordinateurs de l'environnement de développement nécessaire, consistant en une image docker contenant les bibliothèques nécessaires à la compilation du code en C++. L'ensemble de ces bibliothèques est regroupé sous le

nom « CompuTiX ». Notre principale tâche a alors été de coder une première simulation basique d'un ensemble de particules sphériques uniquement soumises à la gravité et aux frottements fluides, c'est à dire une version très simplifiée du processus de simulation décrit précédemment. Le produit de cette simulation sera une série de fichiers VTK (format d'image 3D) représentant le système à chaque instant.

La deuxième tâche de l'équipe de programmation a été d'implémenter les issues rédigées par l'équipe chargée de la modélisation. Chacune d'elle correspondait en général à une force ou un mécanisme de la dynamique des cellules. L'implémenter correspondait à la création d'une "Action", c'est à dire une classe C++ individuelle qui transcrira les mécanismes voulus. La rédaction du code de ces actions était basé sur la méthode du "test-driven development", méthode qui consiste d'abord à écrire les tests qui vérifient le bon fonctionnement du code avant de réellement écrire celui-ci. Cela nous a forcé à concevoir entièrement et en avance l'ensemble des paramètres nécessaires à une action et son fonctionnement précis avant de réellement en écrire le code, ce qui apporte beaucoup de fiabilité au code produit, comme nous l'expliquerons dans la partie 5 (Implémentation de la modélisation en C++). Ainsi, chaque action implémentée pouvait être ajoutée individuellement à la simulation, de manière à en augmenter graduellement la complexité, et d'avoir une traçabilité sur quelle action en particulier pourrait poser problème ou générer des bugs.

4

MODÉLISATION DES PHÉNOMÈNES ET INTERACTIONS

4.1 IDÉE GÉNÉRALE DU MODÈLE

Notre modèle a été pensé de la manière suivante : pour chaque bactérie, individuellement, on s'intéresse à sa potentielle mort, sa croissance, une potentielle division, puis ses interactions avec les autres bactéries et avec le milieu extérieur, le substrat, pour ensuite calculer les vitesses des différentes bactéries et enfin leurs positions, par intégration de la vitesse sur un pas de temps. C'est l'ordre qui a en premier lieu été choisi pour chaque étape de la simulation. (voir annexe)

La modélisation de la mort, la croissance et la division de la bactérie dépend de l'historique de la bactérie (du temps écoulé dans la simulation) et de sa taille, tandis que la modélisation des contacts de cette bactérie avec son milieu extérieur dépend surtout de sa forme. Justement, cette forme rajoute une difficulté supplémentaire au niveau de la modélisation. Une bactérie est une particule possédant une "rod-like shape", c'est-à-dire ayant une forme de cylindre au bout duquel se trouve 2 demi-sphères. Les forces au niveau des contacts entre surfaces ayant des dépendances fortes aux formes de ces surfaces, l'idée première a été de modéliser les 3 types de contacts, sphère/sphère, cylindre/sphère, cylindre/cylindre, et de choisir en fonction de la situation le type de contact adapté.

Mais étant pris par le temps, et par souci de simplicité, nous avons dû nous rabattre sur un modèle n'incluant que des sphères, et ainsi seul le premier type de contact a été traité. Aussi, bien que nous ayons réfléchi à toutes les étapes de la modélisation, seuls les "issues" de croissance, division et contact entre les cellules ont été effectivement implémentées. Également, la partie stochastique de l'équation de Langevin n'a pas pu être traitée.

4.2 MODÈLE EFFECTIVEMENT CODÉ

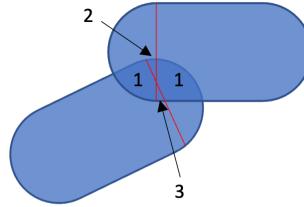


FIGURE 3 – Illustration d'une interaction entre deux cellules. Celle-ci est ici séparable en 3 contacts : sphère-cylindre (1), sphère-sphère (2), cylindre-cylindre (3)

4.2.1 • CROISSANCE

Pour la croissance d'une cellule, nous avons adopté le modèle de croissance adapté au "SIZER mechanism" [4], qui décide que la cellule croît jusqu'à atteindre une taille cible à laquelle elle se divise, contrairement au TIMER par exemple, qui décide de la division à un temps donné. Ce modèle donne une dépendance exponentielle au temps du rayon (et par conséquent du volume) de la cellule. Pour fixer les paramètres de ce modèle, on se base sur une durée de croissance qui correspond au doublement du volume de la cellule. En notant V_{cible} le volume final et V_0 le volume initial, on a $V_0 = \frac{V_{cible}}{2}$, ce qui se traduit en termes de rayon par : $R_0 = \frac{R_{cible}}{2^{\frac{1}{3}}}$.

La croissance du rayon pour un instant t où $R_0 = \frac{R_{cible}}{2^{\frac{1}{3}}} < R(t) < R_{cible}$ est donnée par :

$$R(t) = \frac{R_{cible}}{2^{\frac{1}{3}}} e^{\beta(t-t_0)}$$

où t_0 est telle que $R(t_0) = R_0$ et β est un coefficient calibré pour que $V_0 = \frac{V_{cible}}{2}$. Le calcul donne donc $\beta = \frac{\ln(2)}{3\tau_{doublement}}$ avec $\tau_{doublement}$ le temps de doublement de volume de la cellule.

L'avantage de ce modèle est que de l'instant t à l'instant $t + \Delta t$, le rayon est donné par :

$$R(t + \Delta t) = R(t)e^{\beta\Delta t}$$

et qu'il en va de même pour le volume :

$$V(t + \Delta t) = V(t)e^{3\beta\Delta t}$$

Ainsi, rayon et volume peuvent être facilement recalculé à chaque étape de la modélisation. On peut même, avec un pas de temps fixé, pré-calculer le facteur exponentiel $e^{\beta\Delta t}$.

À noter qu'une idée courante lorsque l'on utilise le modèle SIZER est de faire une approximation linéaire du modèle exponentiel, où le rayon de la cellule deviendrait :

$$R(t) = \frac{R_{cible}}{2^{\frac{1}{3}}} + v_{gr}(t - t_0)$$

où $v_{gr} = \frac{\beta R_{cible}}{2^{\frac{1}{3}}}$ est la vitesse de croissance linéaire de la cellule.

Ça n'a pas été notre cas, nos tuteurs nous conseillant de conserver le modèle original.

4.2.2 • DIVISION

La division d'une cellule a été décomposée entre 3 étapes de modélisation : la transition de la phase de croissance à la phase de division, la modélisation de la mécanique de division et la fin de la phase de division. L'idée générale est de passer d'une cellule de rayon R_{cible} et de volume V_{cible} à deux cellules de rayon adapté pour que le volume total soit conservé au cours de la division. Elles doivent donc avoir un volume $\frac{V_{cible}}{2}$ et

le rayon $\frac{R_{cible}}{2^{\frac{1}{3}}} = \left(\frac{3}{8}V_{cible}\right)^{\frac{1}{3}}$ correspondant à la fin de la division. On modélise également l'éloignement des 2 cellules au cours de la division par une force adaptée (voir explication ci-dessous).

Concernant la transition entre la phase de croissance et celle de division, elle a été modélisée en accord avec le modèle SIZER, c'est-à-dire que l'on a fixé un rayon limite R_{lim} de l'ordre de $1 \mu m$ [7] à partir duquel on arrête la croissance et la cellule passe en phase de division.

Concrètement, l'initialisation de la division se déroule selon le processus suivant :

- choix d'un axe de l'espace suivant lequel la cellule se divise
- calcul de la distance ϵ (overlap) à laquelle la deuxième cellule sera placée par rapport à la première
- création d'une deuxième cellule, et ajustement des rayons des cellules
- placement de la 2ème cellule à ϵ de la première

Initialement, on a une cellule de rayon R_{cible} qui doit donc commencer sa division.

À la fin de l'initialisation, on doit avoir 2 cellules de rayon légèrement inférieur à $cible$, dont les centres sont distants d' ϵ , prêtes à diminuer de volume et à s'éloigner. Elles vont former au cours de la division ce que l'on nomme dans la littérature "haltère" (dumbbell), pour désigner deux sphères dont l'intersection est non nulle. On doit s'assurer que le volume $V_{haltère}$ de cette haltère est toujours proche mais inférieur à $cible$, et on ajustera le taux de décroissance des cellules et la force qui les éloigne pour que cette différence de volume soit le moteur de la division.

On choisit donc d'abord l'axe spatial de la division en choisissant uniformément $\cos(\theta)$ et ϕ dans $[0, 1] \times [0, \pi]$, où θ et ϕ sont 2 angles qui définissent un axe en coordonnées sphériques.

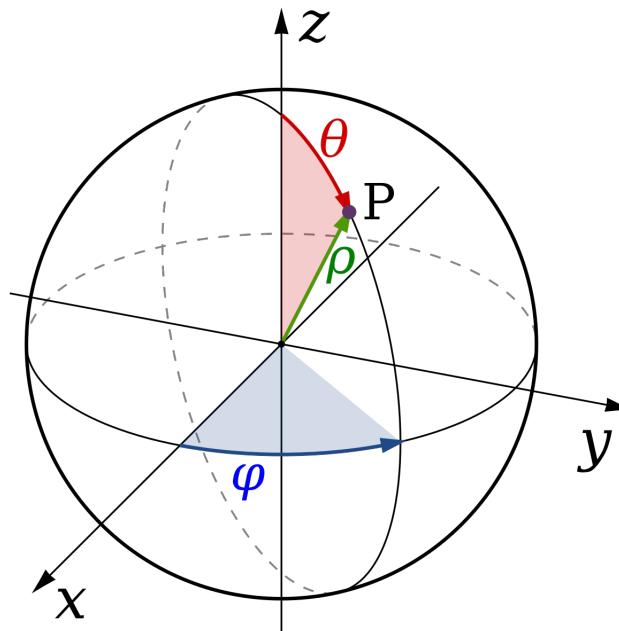


FIGURE 4 – Choix de l'axe de division

Nous avons fait le choix $\epsilon = R_{cible} \frac{dt}{T}$ où dt est un pas de temps de simulation, T l'ordre de grandeur de la durée d'une division, pour qu'en ordre de grandeur cela corresponde à un espacement standard des cellules sur un pas de temps de simulation.

Ensuite, une fois la deuxième cellule créée, on initialise les rayons des cellules à ϵ pour que celui-ci soit suffisamment proche de R_{cible} tout en ayant $V_{cible} > V_{haltère}$. Le calcul de ϵ découle de la formule du volume de l'haltère : $V_{haltère} = \frac{8}{3}\pi R(t)^3 - \frac{1}{12}\pi(6R - \epsilon)\epsilon^2$. Pour valider les conditions de volume et de rayon, le rayon suivant convient : $R_\epsilon = \sqrt[3]{R_{cible}^3 - \frac{\epsilon^3}{16}}$.

Une fois ce rayon calculé, il ne manque plus qu'à placer la deuxième cellule à ϵ suivant l'axe idoine.

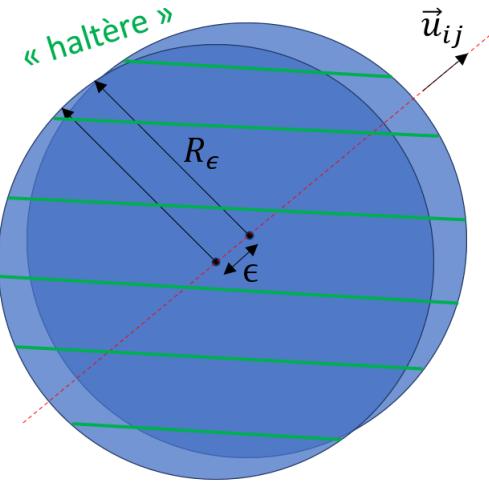


FIGURE 5 – Initialisation de la division

La mécanique de la division est modélisée en deux phénomènes décrit successivement à chaque pas de temps de la simulation :

- la décroissance du rayon des cellules
- une force répulsive qui dépend de la différence des volumes de l'haltère et de la cellule initiale

Le choix des paramètres de ces phénomènes est ajusté par la durée souhaitée de division.

La décroissance des rayons est prise linéaire : $R(t) = R_\epsilon - \alpha(t - t_0)$ en prenant t_0 l'instant de début de division. Le taux de décroissance est donné par $\alpha = \frac{R_{cible}}{T}$, où T est la durée de division souhaitée. En ordre de grandeur, cela permet d'ajuster la durée de division, puisque c'est atteindre le rayon $\frac{R_{cible}}{2^{\frac{3}{2}}}$ qui est le critère d'arrêt de la division.

La force répulsive entre les cellules i et j subie par i est choisie comme il suit :

$$\vec{F}_{ij} = F_0 \ln\left(\frac{V_{haltère}}{V_{cible}}\right) \vec{u}_{ij}$$

où F_0 est une constante choisie pour modéliser la compétition entre la division et la force de friction entre les 2 cellules, soit $F_0 \approx \frac{\gamma\alpha}{\xi}$, où γ est un coefficient de friction entre les 2 cellules [7] et ξ un ordre de grandeur de $\ln\left(\frac{V_{haltère}}{V_{cible}}\right) \approx \frac{V_{haltère} - V_{cible}}{V_{cible}}$. \vec{u}_{ij} est le vecteur unitaire dirigé du centre de la cellule i vers celui de la cellule j.

Le choix des paramètres α et ξ permettent donc d'ajuster la vitesse des cellules au cours de la division ainsi que la durée approximative de celle-ci. Notons que, pour δ_{ij} distance entre les centres des 2 cellules, on a toujours $V_{haltère} = \frac{8}{3}\pi R(t)^3 - \frac{1}{12}\pi(6R - \delta_{ij})\delta_{ij}^2$ inférieur à V_{cible} , donc une force toujours répulsive.

La fin de la division est décidée lorsque les rayons atteignent $\frac{R_{cible}}{2^{\frac{3}{2}}}$. Alors, on arrête de considérer la force répulsive décrite ci-dessus et on considère à nouveau la croissance des deux cellules alors créées. Notons que le volume final agrégé des 2 cellules est bien V_{cible} .

4.2.3 • CONTACT ADHÉSIF ET RÉPULSIF ENTRE LES CELLULES

La modélisation des contacts entre les cellules en termes de force a été décomposé entre les forces adhésives et les forces répulsives. À noter que les forces répulsives et adhésives sont, assez logiquement, proportionnelle à la surface de contact entre elles.

La répulsion entre les cellules a été modélisé simplement par un contact de Hertz [2], qui ne prend donc pas en compte l'adhésion entre les cellules, mais simplement leur élasticité. La force répulsive subie par la cellule i lors du contact entre les cellules i et j est donnée par :

$$\vec{F}_{ij}^{rep} = -\frac{4}{3}\hat{E}_{ij}\sqrt{\hat{R}_{ij}}\delta_{ij}^{\frac{3}{2}}\vec{u}_{ij}$$

où

- \hat{E} est l'opérateur module d'Young $\hat{E} = \left(\frac{1-\nu_i^2}{E_i} + \frac{1-\nu_j^2}{E_j}\right)^{-1}$, ' E_i ' et ' E_j ' étant les module d'Young des membranes et ν_i and ν_j leurs nombres de Poisson, d'une valeur d'environ 0,4 [7]. Habituellement, la valeur de ' \hat{E} ' est de l'ordre de grandeur de $100 MPa$ (REFERENCE DIERCX - 2019).
- \hat{R} est l'opérateur rayon $\hat{R} = \left(\frac{1}{R_i} + \frac{1}{R_j}\right)^{-1}$, R_i , R_j étant les rayons des cellules
- δ_{ij} est "l'overlap" (espacement géométrique) entre les 2 cellules dû à leur déformation, est doit être positif. Si les cellules ne sont pas en contact, $\delta_{ij} = 0$, et la force est nulle. d_{ij} étant la distance entre les centres des cellules, $\delta_{ij} = R_i + R_j - d_{ij}$ si $R_i + R_j > d_{ij}$, $\delta_{ij} = 0$ sinon.
- \vec{u}_{ij} est le vecteur unitaire dirigée du centre de la cellule ' i ' vers le centre de la cellule ' j '.

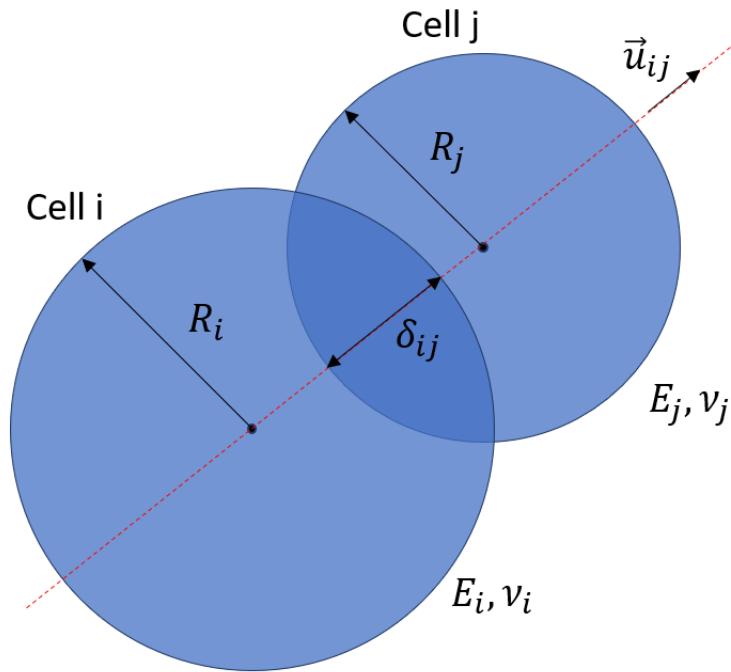


FIGURE 6 – Contact de Hertz

Pour l'adhésion, nous avons choisi (comme souvent dans la littérature) de modéliser le contact adhésif par le modèle de Johnson-Kendall-Roberts (JKR) [2]. Il tient donc compte que la surface des membranes des cellules est adhésive, contrairement au model de Hertz.

La force adhésive subie par une cellule i lors du contact avec une cellule j est donnée par :

$$\vec{F}_{i,j}^{adh} = \frac{4\hat{E}}{3\hat{R}}a^3 - \sqrt{8\pi\sigma\hat{E}a^3}\vec{u}_{ij}$$

où

- σ est une constante assimilable à une constante de raideur, qui caractérise l'importance de l'adhésivité entre les cellules. Habituellement, son ordre de grandeur est de $2.10^{-6} Nm^{-1}$ [7].
- \hat{E} est toujours l'opérateur module d'Young (identique au modèle de Hertz)
- \hat{R} est toujours l'opérateur rayon (identique au modèle de Hertz)
- a est le rayon de la surface de contact entre les cellules

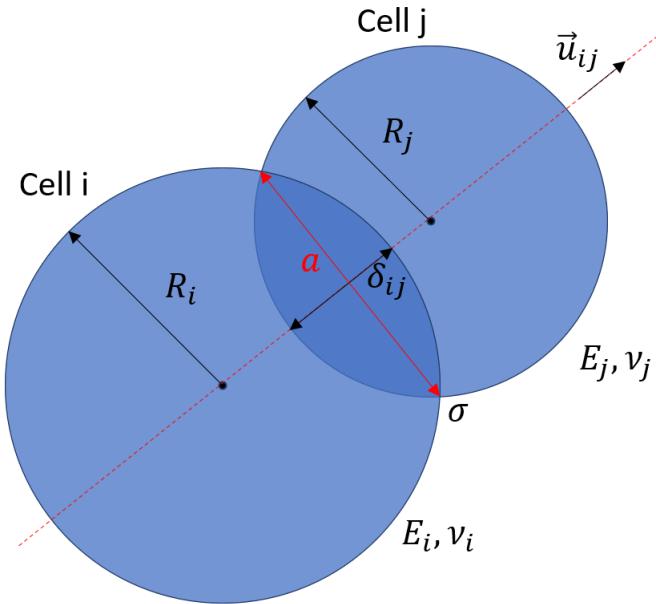


FIGURE 7 – Modèle adhésif JKR

Il est à noter que le paramètre a est une fonction implicite de l'overlap δ_{ij} entre les cellules, donnée par :

$$\delta_{ij} = \frac{a^2}{\hat{R}} - \sqrt{\frac{2\pi\sigma}{\hat{E}}}a$$

et donc a doit être recalculé à chaque étape par une méthode de résolution numérique itérative à partir de δ_{ij} .

4.3 PARTIES DU MODÈLE PRÉVUES MAIS NON IMPLÉMENTÉES

Nous avons établi des modèles pour d'autres parties de la modélisation de la vie d'une cellule mais qui n'ont pas été implémentées, principalement pour des raisons de temps.

4.3.1 • MORT DES CELLULES

La mort des cellules n'a pas pu être implémentée. Cependant, nous avions prévu de modéliser celle-ci par une loi exponentielle en temps pour chaque cellule, dont la densité est donc :

$$f(t) = \lambda e^{-\lambda t}$$

À chaque pas de temps de la modélisation, on tire uniformément dans $[0, 1]$ une variable aléatoire X pour chaque cellule, et si $X < F(t) = 1 - e^{-\lambda t}$, la cellule meurt. Le paramètre λ permet de choisir la durée de vie moyenne d'une cellule, l'espérance de la loi exponentielle étant $\frac{1}{\lambda}$.

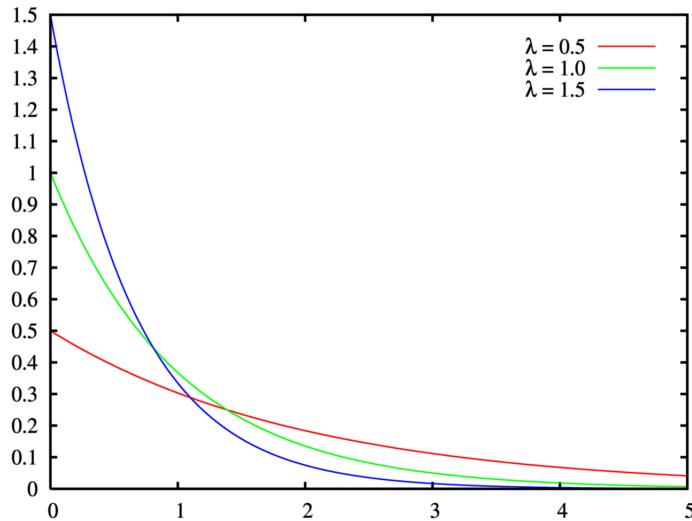


FIGURE 8 – Densité d'une loi exponentielle

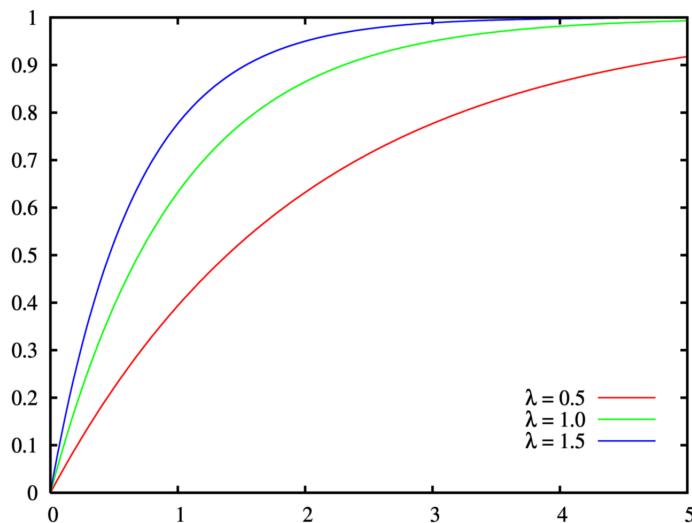


FIGURE 9 – Fonction de répartition d'une loi exponentielle

4.3.2 • CONTACT AVEC LE SUBSTRAT

Les mêmes enjeux de répulsion et d'adhésion qu'entre cellules étaient prévus entre les cellules et le substrat. On a modélisé les contacts correspondant par des contacts sphère/plan infini (sphère de rayon infini), mais dans l'idée le substrat aurait dû être modélisé par une méthode des éléments finis, par une juxtaposition de triangles identiques. Étant donné la dépendance des forces adhésives et répulsives à la surface de contact, on aurait ensuite considéré les contacts avec chacun des éléments triangulaires du substrat pour obtenir une résultante.

Pour la répulsion subie par une cellule i lors de son contact avec le substrat, on modélise par une force de contact de Hertz répulsive, s'appliquant à la cellule i , donnée dans ce cas par :

$$\vec{F}_{i,sub}^{rep} = -\frac{4}{3\pi} \hat{E}_{i,sub} \sqrt{\frac{\delta_{i,sub}}{R_i}} S_{contact} \vec{u}_{i,sub}$$

où :

- \hat{E} est l'opérateur de module d'Young $\hat{E} = \left(\frac{1-\nu_i^2}{E_i} + \frac{1-\nu_{sub}^2}{E_{sub}} \right)^{-1}$, E_i et E_{sub} sont les modules d'Young de la cellule et du substrat et ν_i et ν_{sub} leurs nombres de Poisson.
- R_i est le rayon de la cellule i
- $\delta_{i,sub}$ est le chevauchement géométrique dû à la déformation du capot et du substrat, qui doit être positif. S'ils ne sont pas en contact, $\delta_{i,sub} = 0$, et $\vec{F}_{i,sub}^{rep} = \vec{0}$.
- $\vec{u}_{i,sub}$ est le vecteur unitaire dirigé depuis le centre de la cellule i perpendiculairement à la zone de contact.

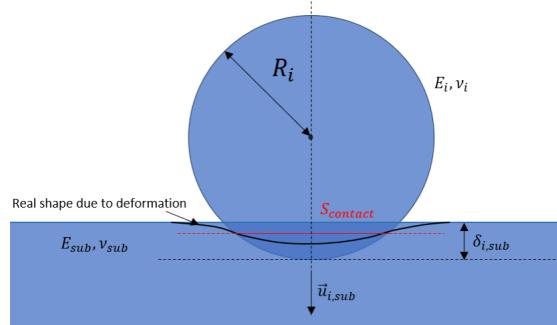


FIGURE 10 – Schéma du contact et placement des paramètres de la force répulsive du contact de Hertz entre la cellule et le substrat.

Pour ce qui est de l'adhésion, on modélise par le modèle JKR avec une des sphères de rayon infini, ce qui donne pour la force subie par la cellule i : La force adhésive subie par une cellule i lors du contact avec une cellule j est donnée par :

$$\vec{F}_{i,sub}^{adh} = \frac{4\hat{E}}{3R_i} a^3 - \sqrt{8\pi\sigma\hat{E}a^3} \vec{u}_{i,sub}$$

où

- σ est une constante assimilable à une constante de raideur, qui caractérise l'importance de l'adhésivité entre la cellules et le substrat. [7].
- \hat{E} est toujours l'opérateur module d'Young (identique au modèle de Hertz pour le substrat)
- R_i est le rayon de la cellule i
- a est le rayon de la surface de contact entre la cellule et le substrat

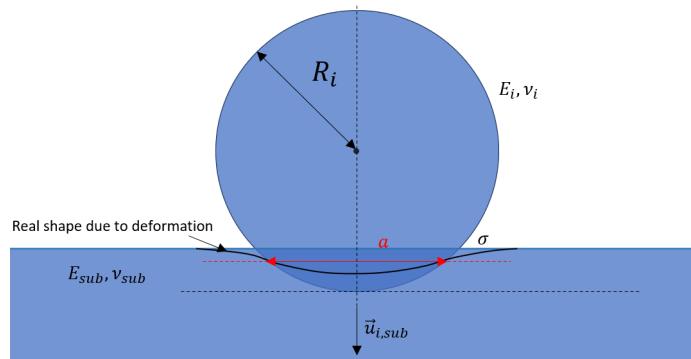


FIGURE 11 – Modèle d'adhésion JKR entre cellule et substrat

4.3.3 • FROTTEMENTS

Nous avions également prévus initialement de modéliser les frottements entre cellules, ainsi qu'entre les cellules et le substrat.

Pour ce qui est du frottement inter-cellules, étant donné que l'on manipule des objets sphériques dans l'espace, on sépare la force en une composante tangentielle et une composante normale au contact, ce qui nous permet d'obtenir un tenseur de frottement que l'on multiplie à droite par le vecteur vitesse relative entre les cellules. [2]. La force subie par la cellule i est donnée par :

$$\vec{F}_{ij}^{fric} = \Gamma_{ij} \cdot (\vec{v}_i - \vec{v}_j)$$

où :

- Γ_{ij} est le tenseur de friction
- \vec{v}_i et \vec{v}_j sont les vitesses des cellules

Plus précisément, le tenseur de friction est obtenu par : $\Gamma_{ij} = \gamma_{\perp}(S_{contact}) (\vec{u}_{ij} \otimes \vec{u}_{ij}) + \gamma_{||}(S_{contact}) (\hat{I} - \vec{u}_{ij} \otimes \vec{u}_{ij})$

où :

- γ_{\perp} est le coefficient de frottement selon la normale au contact, et dépend de la surface de contact
- $\gamma_{||}$ est le coefficient de frottement selon la tangente au contact, et dépend de la surface de contact
- \vec{u}_{ij} est le vecteur unitaire déjà précédemment utilisé

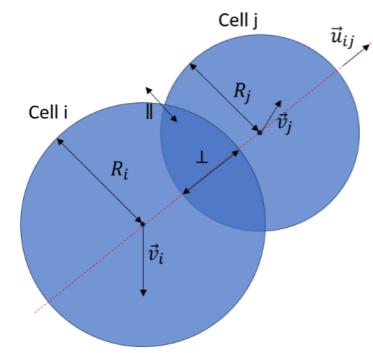


FIGURE 12 – Schéma du contact et placement des paramètres de la force de friction entre cellules

On modélise les frottements cellule-substrat de manière analogue, en considérant le substrat comme une sphère de rayon infini. Nous séparons toujours la force en une composante tangentielle et une composante normale au contact. La force subie par la cellule i est donnée par :

$$\vec{F}_{friction} = \Gamma_i \cdot \vec{v}_i$$

où :

- Γ_i est le tenseur de frottement entre la bactérie et le substrat.
- \vec{v}_i est la vitesse de la cellule.

La formule pour le tenseur de friction est similaire à celle utilisée pour la force de frottement entre deux cellules, considérant que le substrat est une sphère de rayon infini. On peut donc calculer Γ_i :

$$\Gamma_i = \gamma_{\perp} \cdot (\vec{u}_i \otimes \vec{u}_i) + \gamma_{||} \cdot (\hat{I} - \vec{u}_i \otimes \vec{u}_i)$$

- \vec{u}_i est le vecteur unitaire dirigé du centre de la cellule i vers le substrat.
- γ_{\perp} est le coefficient de frottement dans la direction formée par le vecteur normal du substrat.
- $\gamma_{||}$ est le coefficient de frottement dans la direction perpendiculaire à la précédente, tangente à l'aire de contact

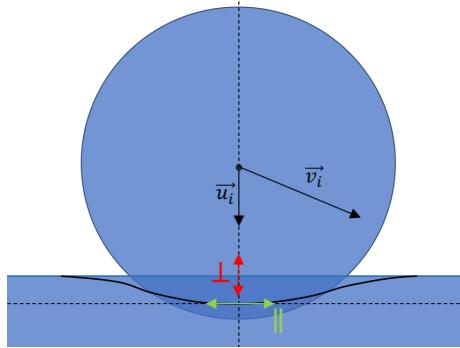


FIGURE 13 – Schéma du contact et placement des paramètres de la force de friction entre cellule et substrat.

5

IMPLÉMENTATION DE LA MODÉLISATION EN C++

5.1 IMPLÉMENTATION DES MODÈLES

CompuTiX est un logiciel informatique contenant un grand nombre de classes agissant sur des particules, et donc ici des cellules. Ce logiciel est en cours de développement depuis plusieurs années à l'INRIA. La partie informatique de notre projet consistait donc au développement de ce logiciel par l'ajout d'une extension, nommée CompuTiX-Bactéria visant à simuler l'évolution d'une colonie de bactéries. Rentrons à présent dans le détail du fonctionnement de CompuTiX :

Dans CompuTiX, il y a trois principales entités à manipuler :

- Les « Collections », qui permettent l'organisation des données, en regroupant les objets de même nature et qui contiennent les particules à manipuler.
- Les « DegreesOfFreedom » (DoF), qui stockent toutes les données et paramètres à tous les niveaux de la simulation.
- Les « Actions », qui représentent les principales opérations, telles que le calcul de forces, l'addition et la multiplication de paramètres scalaires, vectoriels, ou matriciels, la détermination de particules en contact à partir des positions, ou encore l'implémentation des méthodes numériques d'intégration. Ce sont ces dernières qui servent à animer les particules pour ensuite simuler leur développement.

L'objectif était donc de produire des Actions qui implémentaient dans des classes les phénomènes mécaniques modélisés au préalable.

L'écriture de ces classes doit donc suivre le processus de programmation adopté par les chercheurs de l'INRIA : le "test driven development". Cela consiste à d'abord écrire, pour une action donnée, les tests de fonctionnement et d'utilisation de la classe. Il faut donc produire dans un premier fichier de code des fonctions héritant du module google test qui visent à vérifier deux critères : - Tout les paramètres de la classe sont correctement utilisés : ils sont bien donnés en paramètres, possèdent la bonne unité et le bon type. - Le résultat produit par l'implémentation est bien celui attendu.

Pour s'assurer du deuxième point, l'équipe de modélisation prévoyait en même temps que la modélisation d'un phénomène précis une batterie de tests pratiques avec des valeurs numériques et un résultat attendu.

Ce n'est qu'une fois les tests codés et leur implémentation validée par un autre développeur que le travail d'implémentation pouvait commencer. L'implémentation d'une Action suit des codes très standardisés permettant de maximiser la réutilisation du code (modularité), la reproductibilité et le passage à l'échelle (appliquer les Actions à des particules de tout type et en quantité arbitraire).

La programmation se fait en C++ en utilisant presque exclusivement des fichiers faisant partie de la librairie de l'Inria. Certaines fonctions proviennent néanmoins de la librairie standard de C++. Les connaissances techniques en C++ requises pour implémenter les Actions étaient d'un niveau très avancés et un temps d'adaptation et de progression nous fut donc nécessaire.

Par ailleurs, le code doit être minutieusement documenté, pour cela, nous avons du apprendre à utiliser Doxygen, un language conçu spécialement pour la documentation. Là encore, il y a des standards à respecter dans le but que la documentation du logiciel soit la plus universelle et la plus uniforme possible. Ci-dessous nous présentons un extrait de la documentation que nous avons rédigée pour une action implémentant la croissance des cellules.

Public Types

```
using base_type = ComponentFactory
Inherit constructors.

▶ Public Types inherited from CompuTiX::Core::ComponentFactory< T, Base >
```

Public Member Functions

```
void after_create (Token token)
Initialize all relevant parameters. More...

▶ Public Member Functions inherited from CompuTiX::Core::ComponentFactory< T, Base >
```

Additional Inherited Members

```
▶ Static Public Member Functions inherited from CompuTiX::Core::ComponentFactory< T, Base >
▶ Protected Member Functions inherited from CompuTiX::Core::ComponentFactory< T, Base >
```

Detailed Description

Action computing exponential growth of a sphere.

Applies exponential growth to a **DegreeOfFreedom** *r* as

$$r_{\text{after}} = r_{\text{before}} e^{\alpha \Delta t},$$

where *r* is the radius of the particle [m] (before and after **Growth** is applied), $\alpha = \frac{\ln 2}{\tau}$ is the growth parameter [s^{-1}], τ is the doubling time [s] and Δt is time step [s].

Parameters

- collection points to a particle collection from which radius *r*, doubling time *tau* and time step *dt* can be referred to. By default points to parent of *r*.
- *r* points to a **DegreeOfFreedom** of **Scalar** type [m]. By default, it is automatically searched within collection as '*r*'.
- *tau* points to a **DegreeOfFreedom** of **Scalar** type [s]. By default, it is automatically searched within collection as '*tau*'.
- *dt* points to a **DegreeOfFreedom** of **Scalar** type [s]. By default points to '*dt*' upward in the hierarchy.

FIGURE 14 – Extrait de la Documentation de Growth

Enfin, le code est partagé via git et doit passer de nombreux tests de fonctionnement et de robustesse réunis dans une pipeline pour être ajouté au reste du code. La démarche de validation est caractéristique de la recherche en science. En effet, on implémente d'abord une version simple et naturelle de l'Action, cette implémentation est relue par des collègues émettant des remarques visant à améliorer l'implémentation. Puis lorsque la version de l'implémentation est assez robuste et passe tous les tests, on ajoute de nouveaux paramètres puis on optimise sa complexité, aussi bien temporelle que numérique et spatiale.

Nous avons eu l'occasions d'implémenter plusieurs actions au cours de notre projet que l'on peut regrouper en deux principales catégories :

- Les Actions liées à la croissance des cellules : SphereExponential.

- Les Actions liées aux mécanismes de division des cellules. GrowthToDivision, MecanicsOfCellDivision et End-Division.

Nous avons également utilisé deux actions liées aux interactions de contacts entre les cellules faisant déjà parti du code CompuTiX : Hertz et JKR.

Ces deux catégories, ajoutées aux actions élémentaires déjà existantes dans la librairie CompuTiX, permettent déjà de simuler le développement d'une colonie de cellules dans une approximation réaliste. Comme dit précédemment, chacune de ces classes effectue une action élémentaire précise. Ainsi, on présentera ci-dessous l'effet des actions que nous avons implémentées sur un petit nombre de cellules. Une fois mises ensembles, on dispose de la puissance de la modularité pour obtenir une simulation complexe.

Croissance des cellules Growth est une Action qui va augmenter les rayons de chaque particule placée dans une Collection de manière exponentielle.

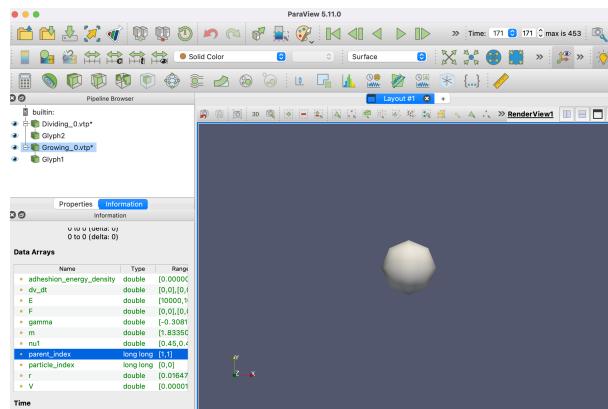


FIGURE 15 – Growth 1

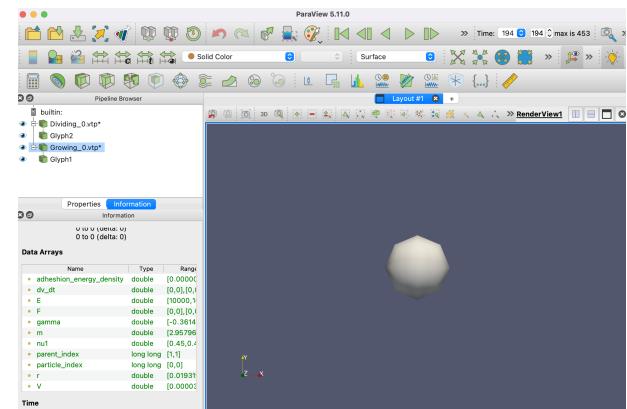


FIGURE 16 – Growth 2

Interactions de contacts entre les cellules Les deux Actions de contacts que nous avons utilisé, le JKR et la force de Hertz, agissent sur deux Collections : Une collection de particule regroupant les cellules et une collection de pair recensant les interactions. Elles suivent l'algorithme suivant : 1. Resolution des contacts, c'est à dire recensement des particules en contact physiquement.
 2. Détermination des surfaces en contact qui paramètrent la force. (comme présenté dans la modélisation)
 3. Calcul numérique des forces de contact
 4. Distribution des contacts aux deux membres de la paire en interaction

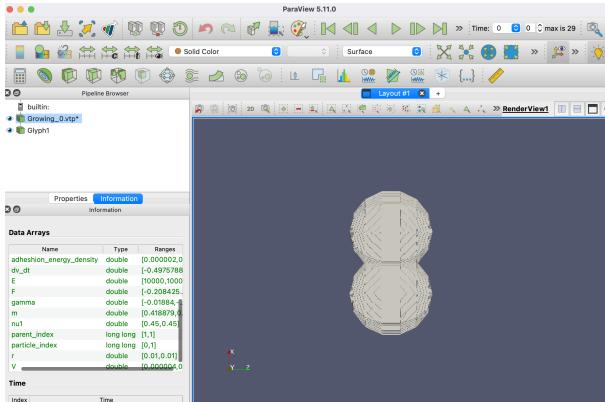


FIGURE 17 – Contact 1

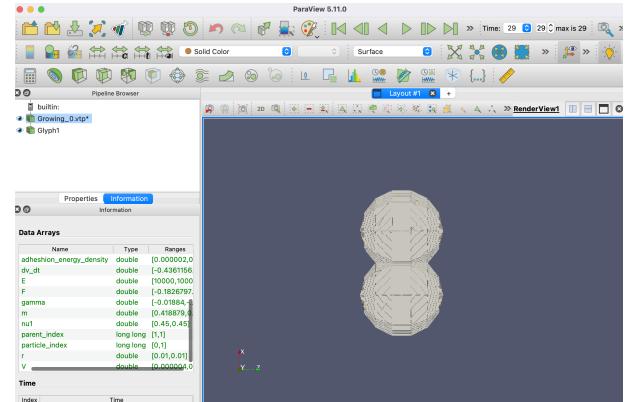


FIGURE 18 – Contact 2

Division La division se déroule en trois étapes implémentées dans une Action chacune, on peut résumer l'effect de la division par le schéma ci-dessous :

Algorithm Division:

Input Particle Collection**Set Degrees of Freedom: r, m, v, F**

- 1: **IF** ($r \geq max_r$) **do:**
 - 2: Move Particle to Dividing collection
 - 3: Duplicate Particle inside Dividing collection
 - 4: Set overlap between the two cells to epsilon
 - 5: Set r for the two cells to r_{epsilon}
 - 6: **WHILE** ($r \geq r_{\text{fin}}$) **do:**
 - 7: Compute Hertz forces
 - 8: Reduce radius linearly // $r = r - \alpha dt$
 - 9: Compute speed and position with Euler Method
 - 10: **END WHILE**
 - 11: Move the two cells to dividing collection
-

FIGURE 19 – Algorithme de Division

Il nous a fallu un certain temps pour s'adapter à ces méthodes de programmation que l'on retrouve dans les grandes entreprises. Cependant, une fois familiarisés avec ces pratiques, nous avons pu commencer à apprécier

la puissance d'une telle manière de programmer. En effet, la sémentation du code en actions élémentaires testées séparément facilite grandement la correction des bugs informatiques. En particulier, une fois le code ajouté à la librairie, on est en pratique presque sûr qu'il fonctionne correctement et produit le résultat attendu. Ainsi, lorsque nous avons composé la simulation, nous n'avons pas eu à nous soucier du fonctionnement interne des actions lors de la correction des problèmes de la simulation.

5.2 PRODUCTION D'UNE SIMULATION

En parallèle de l'implémentation des actions, nous avons mis au point une simulation qui nous permettrait d'appliquer ces actions nouvellement codées.

Tout d'abord, nous simulions donc l'évolution de particules sphériques de rayon R , caractérisées par leur masse m , leur vitesse \vec{v} , leur volume R , et la force qu'elles subissent F . Cette force est la somme de la force de gravitation $P = mg$ et de la force de frottement fluide du modèle de Stokes : $\vec{F}_f = -6\pi\mu R\vec{v}$. Nous suivons le processus de simulation déjà décrit, avec un temps discréteisé et une intégration pas à pas avec la méthode d'Euler. Un fichier vtk (rendu 3D) est produit à chaque itération de la simulation.

Sans rentrer démesurément dans les détails techniques de l'implémentation, une première collection universe contient, d'un côté, toutes les données de la simulation, à savoir le temps courant, l'intervalle de temps entre deux instants de la simulation, le coefficient de viscosité du milieu ambiant, et des valeurs numériques utiles pour le problème. Elle contient elle-même la collection « Particle materials », qui contient les différents types de particules utilisées, ici simplement déterminées par leur masse volumique. Cette organisation est explicitée graphiquement sur la figure suivante :

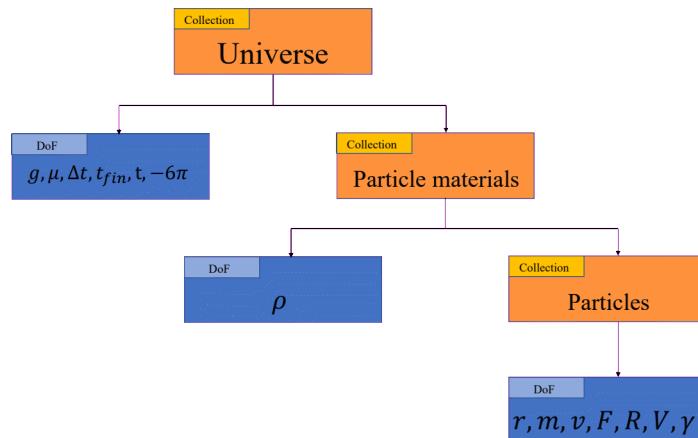


FIGURE 20 – Architecture de la première simulation

Pour donner une idée des délais et des difficultés rencontrées, cette simulation assez simple en apparence ne fut complètement fonction que courant février-mars 2023. De longues séances de déboggage nous ont même parfois fait découvrir des erreurs dans la documentation du fonctionnement de la bibliothèque CompuTiX, et nous devions ainsi attendre un dialogue avec nos tuteurs pour pouvoir nous en sortir.

Une fois cette simulation réalisée et fonctionnelle, il fut temps de rajouter les différentes actions que nous avions codées. D'abord la croissance, puis les interactions répulsives et adhésives, et enfin la division. Pour ce faire, nous avons opté pour l'architecture suivante : Désormais nous disposons de deux collections de particules : la collection growing, qui conteint les particules durant leur croissance, et la collection dividing, qui contient celles en cours de division. L'ajout d'interactions entre les cellules nous a conduit à manipuler une nouvelle structure de données : les Pair Collection, qui correspondent essentiellement à l'ensemble des paires formées à partir des particules de deux collections, distinctes ou non. Les forces d'interaction entre particules s'appliquent alors via des Actions qui agissent sur ces pair collections. Deux particules en cours de croissance interagissent selon le modèle Hertz-JKR décrit précédemment, tandis que deux cellules se divisant sont soumises à une force spéciale de séparation, moins violent que la répulsion de Hertz.

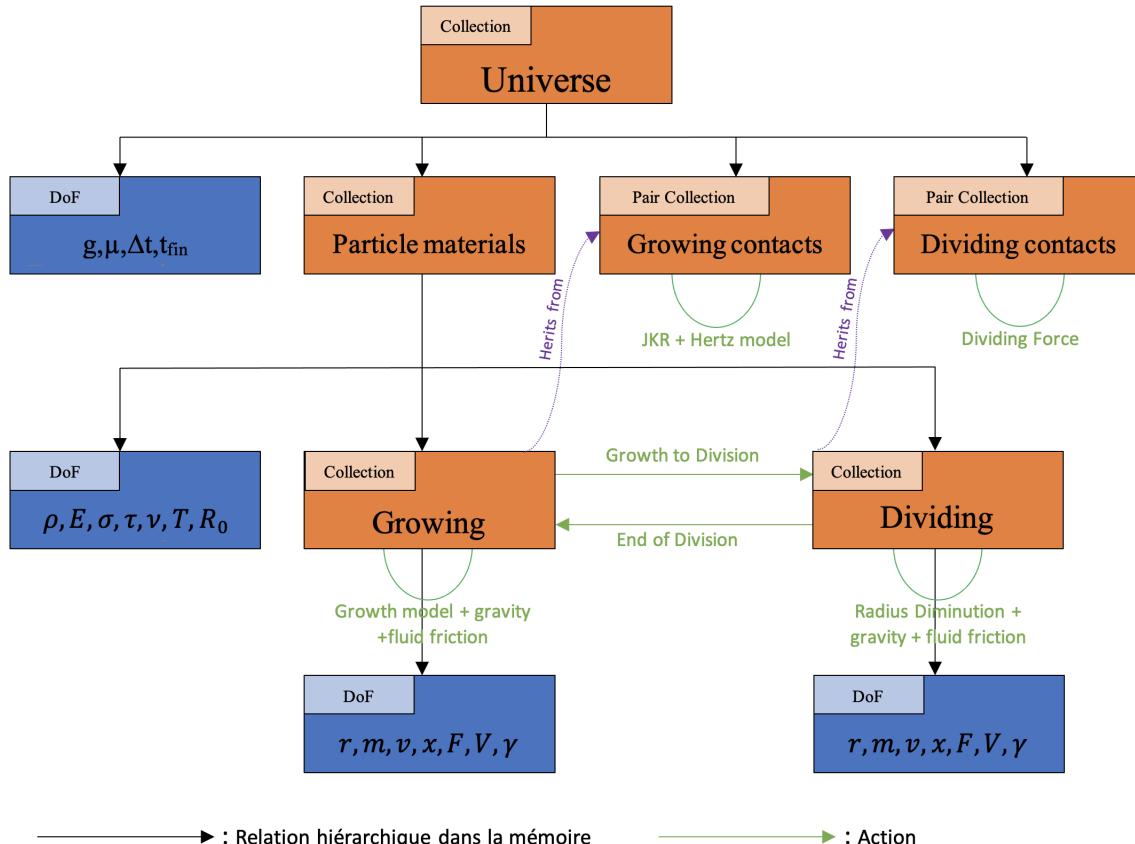


FIGURE 21 – Architecture de la simulation finale

Le fonctionnement de l'algorithme de simulation est itératif, et est résumé sur la Figure 24. Un certain nombre de failles peut être rencontré dans cet algorithme : déjà nous avons négligé les interactions entre les particules en cours de division et celles en cours de croissance, entre lesquelles nous aurions pu introduire un nouveau modèle Hertz-JKR. De même, si deux couples de particules en cours de division venaient à rentrer en contact, leurs interactions seraient gouvernées par le modèle qui donne la force qui fait se diviser une cellule, au lieu d'une interaction Hertz-JKR qui serait préférable. Ces deux problèmes peuvent néanmoins être nuancés : les particules se divisant étant assez rare dans la modélisation, il ya peu de chances que deux couples se rencontrent.

```

Input Universe
Input Materials
Input Growing Collection
Input Dividing Collection
Input Interactions Collection
Input Contacts Collection

Set Degrees of Freedom: r, m, v, F, E, nu, gamma

1: Create Loop: while  $t < t_{fin}$ 
2: Compute Volume of Particles //  $V = \frac{4}{3} * \pi * r^3$ 
3: Compute Mass //  $m = V * \rho$ 
4: Compute Fluid Friction
5: Execute Growth
6: Execute Division
7: Compute Contact Forces
8: Integrate velocity
9: Integrate Position
10: create vtk file of the current collection
11: END WHILE

```

FIGURE 22 – Fonctionnement de l'algorithme de la simulation finale

6

RÉSULTATS ET EXPLOITATION

Dans cette partie, nous exposerons quelques résultats de la simulation que nous avons implémenté. Nous avons lancé la simulation avec initialement 10 cellules, avec les valeurs suivantes pour les paramètres :

- $r = 10^{-6} \text{ m}$ (Le rayon)
- $\rho = 10^3 \text{ kg/m}^3$ (La masse volumique)
- $E = 100 \text{ MPa}$ (Le module d'Young)
- $\nu = 0.4$ (Le nombre de Poisson)
- $\gamma = 2 * 10^{-6} \text{ N/m}$ (L'énergie d'adhésion)
- $\eta = 1^{-2}$ (Le coefficient de frottement fluide)

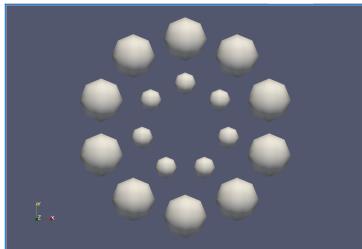


FIGURE 23 – Simulation 1

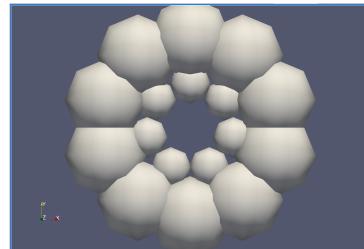


FIGURE 24 – Simulation 2

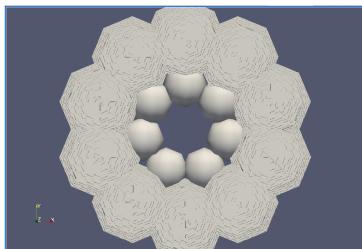


FIGURE 25 – Simulation 3

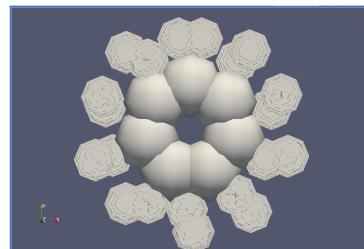


FIGURE 26 – Simulation 4

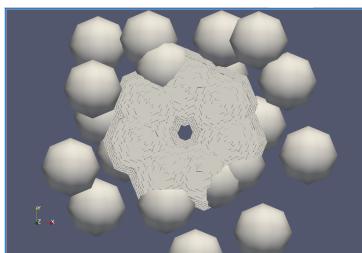


FIGURE 27 – Simulation 5

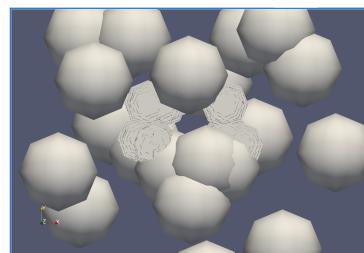


FIGURE 28 – Simulation 6

On observe sur les images ci-dessus qu'en raison des phénomènes de division et de croissance qui prennent le dessus sur les forces de contact, il se crée un amas de cellule. On retrouve ce genre d'amas dans les cultures de cellules et de bactéries et dans la littérature notamment.[2] En relançant la simulation avec des paramètres

différents, on peut observer que certains paramètres comme le taux de grossissement ou le module d'Young jouent un rôle majeure dans la vitesse de développement de l'amas.

Néanmoins, nous n'avons pu mettre au point cette simulation que tardivement et n'avons pas eu le temps de l'exécuter dans beaucoup de configurations différentes pour dégager des tendances et reléver des valeurs références pour les paramètres du modèle. Ceci constituerait un bon prolongement de notre projet.

On s'aperçoit cependant que la simulation n'est pas encore tout à fait fonctionnelle. Nous avons pu identifier les problèmes suivants dont certains sont résolus et dont les autres pourront être résolus à l'avenir :

- La simulation comporte des erreurs numériques. En effet, dans la méthode d'Euler, si les grandeurs varient trop brusquement et que dt n'est pas assez petit pour capter ces variations brusques, alors la simulation commet des erreurs d'approximations très importantes qui ont pour conséquence la divergence des paramètres du modèle tels que la Force, la vitesse ou la position.

Certaines cellules ne semblent pas bouger dans la bonne direction. En effet, nous nous sommes rendus compte en exécutant la simulation que certaines cellules n'étaient pas à la bonne place. Nous avons pu identifier un oubli dans la modélisation du problème qui est la prise en compte des interactions entre les cellules se divisant et les cellules en croissance. Nous pourrons à l'avenir ajouter ces modifications à notre simulation.

L'intérêt de la modélisation basé sur l'individu est donc de permettre de constituer peu à peu un modèle de plus en plus réaliste en ajoutant de nouveaux paramètres en fonction des résultats observés sur la simulation.

7 CONCLUSION

Ce projet scientifique a été une occasion unique pour nous de développer nos compétences en matière de travail en équipe professionnelle. Nous avons dû apprendre à collaborer efficacement avec des collègues ayant des compétences et des appétences différentes des nôtres, à communiquer clairement et à résoudre les problèmes et qui pouvaient survenir, ce qui nous a en somme permis de développer nos compétences en matière de gestion de projet.

Nous avons également été confrontés à des codes et des contraintes qui n'existent pas dans un cadre scolaire. Nous avons dû nous adapter régulièrement pour produire une simulation satisfaisant nos objectifs, tout en respectant les exigences de qualité et de délai fixées par notre équipe de projet.

De plus, nous avons dû faire face à des problèmes juridiques liés au partage des données de l'INRIA. Nous avons dû trouver des solutions pour contourner ces obstacles et continuer à avancer vers nos objectifs.

Dans l'ensemble, ce projet scientifique nous a permis de développer nos compétences en matière de travail d'équipe, de communication, de résolution de problèmes et de gestion de projet. Nous avons appris à travailler efficacement ensemble pour produire un travail de qualité dans un délai imparti. Ces compétences sont précieuses et nous seront utiles tout au long de notre vie professionnelle.

8

RÉFÉRENCES

- [1] Andreas Buttensch on1, Margriet Palm, Paul van Liedekerke, Philippe Chavrier, and Dirk Drasdo, Does single cell migration behaviour permit prediction of multi cellular migration patterns Lessons from a physics based model, 2022
- [2] Van Liedekerke, Simulating tissue mechanics with agent-based models : concepts, perspectives and some novel results, 2015
- [3] Sherli Koshy-Chenthittayil, Linda Archambault, Dhananjai Senthilkumar , Reinhard Laubenbacher , Pedro Mendes 1,5 and Anna Dongari-Bagtzoglou, Agent Based Models of Polymicrobial Biofilms and the Microbiome—A Review,
- [4] Andres Delgado-Campos and Alejandro Cueto, Influence of homeostatic mechanisms of bacterial growth and division on structural, 2022 properties of microcolonies. A computer simulation study, 2022
- [5] Jan-Ulrich Kreft, Individual-based modelling of biofilms, 2001
- [6] Dmitri Volfson, Scott Cookson, Jeff Hasty, and Lev S. Tsimring, Biomechanical ordering of dense cell populations, 2008
- [7] Caroline Dierckx, Investigation of the influence of the EPS production on the biofilm development. A numerical essay, 2019

9 ANNEXE

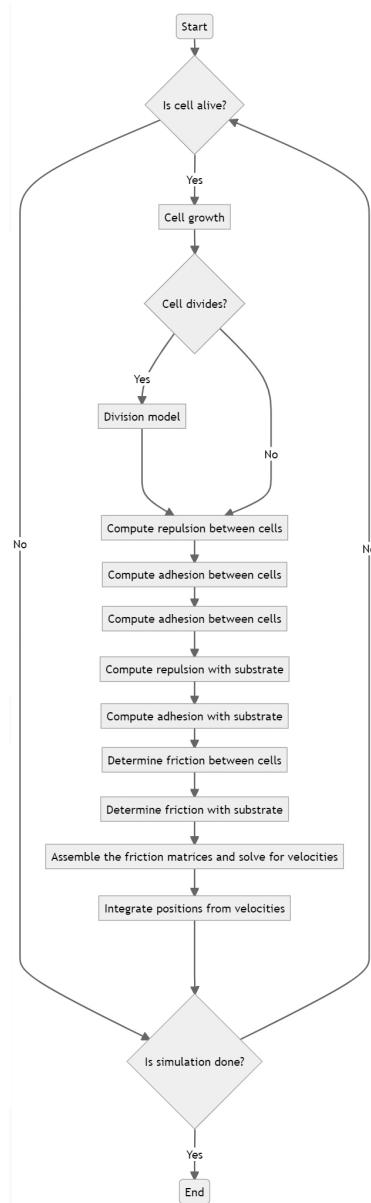


FIGURE 29 – Cycle de la simulation de la vie d'une cellule