Multi-Scale Modelling of the Intra-Cellular Actin Dynamics

Anne van Gorp

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAX034

Thèse de doctorat

Multi-Scale Modelling of the Intra-Cellular Actin Dynamics

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École polytechnique

École doctorale n°574 École doctorale de mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques appliquées

Thèse présentée et soutenue à Palaiseau, le 6 juillet 2022, par

ANNE VAN GORP LANVIN

Composition du Jury :

Vincent Bansaye Professeur, École polytechnique (CMAP)	Président
Jean-François Delmas Professeur, École des Ponts ParisTech (CERMICS)	Rapporteur
Benoîte de Saporta Professeure, Université de Montpellier (IMAG)	Rapporteure
Nicolas Meunier Professeur, Université d'Évry Val d'Essonne (LaMME)	Examinateur
Anne-Cécile Reymann Chargée de recherche CNRS, Université de Strasbourg (IGBMC)	Examinatrice
Amandine Véber Directrice de recherche CNRS, Université Paris Cité (MAP5)	Directrice de thèse
François Robin Chargé de recherche INSERM, Sorbonne Université (IBPS - Biologie du Développement)	Co-directeur de thèse

INSTITUT POLYTECHNIQUE DE PARIS

ÉCOLE POLYTECHNIQUE

IP PARIS

Multi-Scale Modelling of the Intra-Cellular Actin Dynamics

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École polytechnique

École doctorale n°574 École doctorale de mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques appliquées

Thèse présentée et soutenue à Palaiseau, le 6 juillet 2022, par

ANNE VAN GORP LANVIN

Composition du Jury :

Vincent Bansaye Professeur, École polytechnique (CMAP)	Président
Jean-François Delmas Professeur, École des Ponts ParisTech (CERMICS)	Rapporteur
Benoîte de Saporta Professeure, Université de Montpellier (IMAG)	Rapporteure
Nicolas Meunier Professeur, Université d'Évry Val d'Essonne (LaMME)	Examinateur
Anne-Cécile Reymann Chargée de recherche CNRS, Université de Strasbourg (IGBMC)	Examinatrice
Amandine Véber Directrice de recherche CNRS, Université Paris Cité (MAP5)	Directrice de thèse
François Robin Chargé de recherche INSERM, Sorbonne Université (IBPS - Biologie du Développement)	Co-directeur de thèse

Thèse de doctorat

Abstract

Title: Multi-Scale Modelling of the Intra-Cellular Actin Dynamics

Keywords: Mathematical modelling of actin, Measure-valued Markov processes, Tree-valued processes, Convergence theorems, Homogenisation theorem, Metric measure spaces

This thesis focuses on understanding the dynamics of the actin polymer network constituting the mechanical envelope of a cell. We have developed different mathematical models, considering different scales, in order to understand the different mechanisms.

The mechanical properties of embryonic cells, which are involved in many cellular behaviours, such as cell division or cell shape changes, are largely defined by the dynamics of a network of biological polymers, the actomyosin cortex. Each of these polymers, consisting of actin monomers, has a dynamic of elongation, shortening and potentially branching. These dynamics are subject to a high degree of stochasticity, which is not yet fully considered in existing models. In order to understand how the mechanical properties of cells stem from actin biochemistry, it is necessary to understand how the assembly dynamics of these polymers operate in the cell, first at the scale of a single polymer, then at the level of a population of actin polymers.

This thesis is divided into three parts. The first part is focused on the study of the elongation-shortening dynamics of actin in the presence of two proteins, formin and profilin, which accelerate the elongation of the actin polymer. Initially, we consider a single actin polymer, formed by a number of monomers that fluctuates over time. The main idea of our model is to see an actin polymer as a queue; this allows us to introduce a stochastic component, inherent to the randomness of the encounters and fixations between the different entities involved. The elongation-shortening dynamics of the polymers is then entirely described by a jump Markov process. In order to obtain an average behaviour, we studied the limit when the total number of monomers tends to infinity (average field limit). The study of the deterministic limit process obtained allows us to obtain an average polymer length but also to observe the influence of the change in elongation rate on the use of the resource, namely free monomers. Secondly, we consider a population of polymers. Each polymer evolves with the dynamics described in the previous model. The dynamics of the population is this time described by a measure-valued Markov process. The study of the large population limit allows us to obtain a distribution of polymer lengths and specially to observe that there is competition between the polymers for free monomers. The size distribution of the polymers is particularly important since the mechanical properties of the cells depend directly on the length of the actin polymers.

The second part deals with the study of the branching dynamics of an actin polymer. In order to model a branched polymer, we use the formalism of metric measure spaces. The aim of this part is to better understand the structure of branched polymers, which could, for example, help to better understand the formation of protrusions. In this model, we consider two proteins that can interact with actin: the Arp2/3 protein complex, which creates a new branching, and cofilin, which causes the fragmentation of the polymer and thus the loss of branches. The dynamics of the branched polymer is then fully

described by a tree-valued process. We proved that the process describing the number of extremities (renormalized) converges in distribution to a Feller diffusion.

The third part is focused on numerical exploration of the different models. The simulations for the models studying the elongation-shortening dynamics allowed us to understand the role of each protein and to obtain a qualitative study of the average behaviours. The simulations for the model studying a single branched polymer allowed us to compare structures obtained by testing different hypotheses. Finally, we extended the model for a population of polymers (and the corresponding code) to consider more proteins interacting with actin.

Résumé

Titre : Modélisation multi-échelle de la dynamique intra-cellulaire de l'actine

Mots-clefs : Modélisation mathématique de l'actine, Processus de Markov à valeurs mesure, Processus à valeurs arbre, Théorèmes de convergence, Théorème d'homogénéisation, Espaces métriques mesurés

Cette thèse porte sur la compréhension de la dynamique du réseau de polymères d'actine constituant l'enveloppe mécanique d'une cellule. Nous avons élaboré différents modèles mathématiques, prenant en compte différentes échelles, afin de comprendre les différents mécanismes.

Les propriétés mécaniques des cellules embryonnaires, responsables d'un grand nombre de comportements cellulaires, comme la division cellulaire ou les changements de forme des cellules, sont largement définies par la dynamique d'un réseau de polymères biologiques, le cortex d'actomyosine. Chacun de ces polymères, constitué de monomères d'actine, a une dynamique d'élongation, raccourcissement et potentiellement de branchement. Cette dynamique est soumise à une forte stochasticité, encore très peu prise en compte dans les modèles existants. Afin de comprendre comment les propriétés mécaniques des cellules émergent de la biochimie de l'actine, il faut comprendre comment la dynamique d'assemblage de ces polymères fonctionne dans la cellule, d'abord à l'échelle d'un seul polymère, puis au niveau d'une population de polymères d'actine.

Cette thèse s'articule en trois parties. La première est consacrée à l'étude de la dynamique d'élongation-raccourcissement de l'actine en présence de deux protéines, la formine et la profiline, qui permettent d'accélérer l'élongation du polymère d'actine. Dans un premier temps, nous considérons un seul polymère d'actine, formé d'un nombre de monomères qui fluctue au cours du temps. La dynamique d'élongation-raccourcissement des polymères est entièrement décrite par un processus markovien de sauts. Afin d'obtenir un comportement moyen, nous avons étudié la limite quand le nombre de monomères total tend vers l'infini (limite de champ moyen). L'étude du processus limite déterministe obtenu nous permet d'obtenir une longueur moyenne de polymère mais aussi d'observer l'influence du changement de vitesse d'élongation sur l'utilisation des monomères libres. Dans un second temps, nous nous plaçons à l'échelle d'une population de polymères. La dynamique de la population est cette fois décrite par un processus de Markov à valeurs mesure. L'étude de la limite en grande population nous permet d'obtenir une distribution des tailles des polymères mais surtout de constater qu'il existe une compétition entre les polymères pour les monomères libres. La distribution des tailles des polymères est particulièrement importante puisque les propriétés mécaniques des cellules dépendent directement de la taille des polymères d'actine.

La deuxième partie se concentre sur l'étude de la dynamique de branchement d'un polymère d'actine. Afin de modéliser un polymère ramifié, nous utilisons le formalisme des espaces métriques mesurés. Cette partie a pour but de mieux comprendre la structure des polymères ramifiés, ce qui pourra par exemple permettre de mieux comprendre la formation des protrusions. Nous considérons dans ce modèle deux protéines pouvant interagir avec l'actine : le complexe protéique Arp2/3, permettant de créer un nouveau branchement et la cofiline, entrainant la fragmentation du polymère et donc la perte de

branches. La dynamique du polymère ramifié est alors entièrement décrite par un processus à valeurs arbre. Nous avons prouvé que le processus décrivant le nombre d'extrémités (renormalisé) converge en distribution vers une diffusion de Feller.

La troisième partie est dédiée à l'exploration numérique des différents modèles. Les simulations pour les modèles étudiant la dynamique d'élongation-raccourcissement nous ont permis de comprendre le rôle de chaque protéine et d'obtenir une étude qualitative des comportements moyens. Les simulations pour le modèle étudiant un unique polymère branché nous ont permis de comparer les structures obtenues en testant différentes hypothèses. Finalement, nous avons élargi le modèle pour une population de polymères (et le code correspondant) afin de prendre en compte plus de protéines interagissant avec l'actine.

Remerciements

Je tiens tout d'abord à remercier mes rapporteur·e·s Jean-François Delmas et Benoîte de Saporta pour avoir accepté de relire mon manuscrit ainsi que pour leur riche retour. Je remercie également chaleureusement mes examinateurs et examinatrice Vincent Bansaye, Nicolas Meunier et Anne-Cécile Reymann pour avoir accepté de faire partie de mon jury.

Je tiens ensuite à remercier ma directrice et mon directeur, Amandine et François. J'ai appris énormément à vos côtés, tant sur le point mathématique ou biologique que sur le plan humain. J'étais loin d'imaginer en me lançant dans mon projet de M2 que ce sujet m'occuperait pendant six ans. Merci pour votre patience et votre soutien. Merci d'avoir cru en moi, même quand moi je n'y croyais plus.

Je remercie la chaire Modélisation Mathématique et Biodiversité pour le financement de cette thèse. Merci pour les rencontres et écoles de printemps qui mêlent à merveille l'utile à l'agréable. Merci à l'ensemble de ses membres pour les nombreux exposés toujours très intéressants et toujours présentés dans une ambiance agréable et détendue. Merci François et Thibaut pour les superbes sorties dans la montagne à Aussois.

Merci aux laboratoires qui m'ont accueillie. Merci aux doctorant·e·s du CMAP pour leur accueil constant malgré mes absences de plus en plus régulières. Merci Apolline pour tes conseils pour les procédures de fin de thèse. Merci Céline pour les discussions et pour ton aide dans mes premiers moments de doute. Merci également à l'équipe administrative pour votre efficacité.

Merci à toute l'équipe CADMO de l'IBPS. Merci Camille, Jean-François, Shashi, Simon et Vlad pour votre accueil lors de mon stage en biologie puis lors de mes visites. Merci de m'avoir guidé et expliqué le fonctionnement d'un laboratoire de biologie. Merci Séréna pour ton soutien sans faille. Merci d'avoir toujours été mon alliée et de me comprendre si bien.

Merci aux doctorant·e·s de l'IRIF pour m'avoir permis de m'incruster dans vos sorties et dans vos concours de gâteaux. Merci aux doctorants du bureau 3035 pour avoir accepté ma présence alors que je n'avais, a priori, rien à faire là. Merci Adrien pour les soirées jeux. Merci Niols, parce que bien sûr que ça compte, le 3035 est sans nul doute le bureau dans lequel j'ai passé le plus de temps. Merci de m'avoir aidé à me mettre au travail certains jours et à arrêter de penser au travail d'autres jours.

Merci à toute l'équipe enseignante du département GEII2 de l'IUT de Cachan. Merci de m'avoir guidé alors que je faisais mes premiers pas en tant que professeure. Merci de m'avoir aidé à dédramatiser ma situation ; faire une thèse en cinq ans ce n'est pas si grave finalement. Merci Eric, Fabien, Franck et Pascal pour ces quelques questions régulières sur ma thèse qui m'ont finalement motivées à la finir (parce que oui, moi aussi à un moment j'ai cru que je ne la terminerais pas). Merci Eric de partager ta pédagogie. Tu es un modèle pour moi, j'ai appris beaucoup grâce à toi et je suis sûre que j'apprendrais encore plein de choses dans les prochaines années. Je suis heureuse et fière de continuer à travailler avec vous.

Merci à tou·te·s mes étudiant·e·s, échanger avec vous me permet (je l'espère) de devenir une meilleure prof à chaque cours. J'espère vous avoir aidé d'une manière ou d'une autre.

L'aboutissement de cette thèse, et de mes études en général, n'aurait pas été possible sans mes (trop) nombreuses activités, des personnes que j'y ai rencontré, de mes ami·e·s et de ma famille.

Merci à tous les membres de la RSCDS Paris Branch. Vous retrouver chaque lundi (et certains samedis ou dimanches) pour s'hydrater, danser et se réhydrater m'a permis de m'évader et d'oublier tous mes soucis (d'où le nom de ma danse "Stop Worrying and have Fun" [Par21]). Merci Niols de m'avoir fait découvrir le doctorat en spécialité mineure danse écossaise. Merci Antoine et Judith pour votre attention et vos conseils pour la fin de thèse. Merci aux membres du Paris Band, j'adore jouer de la musique avec vous (surtout en pleine nuit quand on échange nos instruments). Merci Caroline pour ton oreille attentive et d'avoir toujours une histoire incroyable à raconter. Merci Pascaline pour ta bienveillance. Merci pour ces discussions qui m'ont fait réfléchir à toutes sortes de sujets. Merci de m'aider à avoir confiance en moi et à m'affirmer telle que je suis. Merci Rémi pour ton extrême gentillesse.

Merci à mes professeur·e·s de musique de m'aider à continuer à pratiquer mes instruments malgré mes études, mon emploi du temps chargé et parfois ma fatigue intense. Jouer de la musique m'a toujours permis de me vider la tête et m'a infiniment aidé pendant la rédaction, particulièrement pendant les confinements. Merci Carole et Isabelle de m'avoir accompagné depuis l'enfance et de m'avoir transmis votre passion. Merci Florent pour ta positivité en toute circonstance. Merci Kyo et Lucie de m'avoir permis de reprendre le violoncelle malgré le peu de temps dont je dispose. Merci Michel d'accepter de m'apprendre le piano alors que pour la majorité des gens, apprendre un troisième instrument c'est juste de la folie. Merci à mes camarades d'orchestres et de musique de chambre pour ces moments de partage.

Merci Laura d'avoir était ma binôme en M2 et d'avoir développé avec moi ce qui est devenu le préambule de cette thèse. Merci d'avoir partagé toutes tes connaissances sur l'actine. Merci Carole et Florence pour nos années passées à la fac d'Orsay. Merci Florence de m'avoir motivée à réviser mon agreg (même quand je disais que c'était perdu d'avance) et pour ton aide précieuse. Désolée pour mon pessimisme de l'époque et pour mes paroles maladroites ; quand j'y repense je me dis que j'ai pu être méchante sans le vouloir... Pardon aussi, Florence et Laura, d'avoir pris trop de place dans nos projets de M1 et de M2. J'espère avoir amélioré ma capacité à travailler en groupe depuis.

Merci Caroline de m'accompagner dans toutes les étapes de ma vie depuis le CE1. Merci d'être toujours là pour m'écouter. J'ai une chance incroyable d'avoir une amie comme toi.

Merci à ma famille. Maman, Papa, Cécile, Laurent, Romain, Adrien, Jérémy, Nina, Clément, merci d'être toujours présent·e·s, je vous aime. Merci à mes parents de m'avoir rendue aussi curieuse et de m'avoir appris tant de choses dans tant de domaine. Merci à Adrien et Jérémy d'être passé par là avant moi et d'avoir toujours de bon conseils. Merci à ma belle-famille parce qu'on ne peut pas rêver en avoir une meilleure.

Finalement, merci infiniment à Victor. Tu m'as sauvé tant de fois que j'ai perdu le compte. Nous avons partagé nos douze années d'études mais c'est tellement plus que ça. Sans toi j'aurais abandonné dès la première année. Sans toi rien de tout cela n'aurait été possible. Merci d'avoir grandi avec moi, en regardant toujours dans la même direction que moi. Merci de me supporter même quand je suis insupportable. Merci de tout partager avec moi, les rires comme les larmes. Merci de croire en moi pour nous deux. Merci pour ton amour et ton soutien inconditionnel. Merci d'être mon mari, je t'aime.

Table of Contents

List of Figures

Introduction

Sommaire

Les propriétés mécaniques des cellules embryonnaires sont responsables d'un grand nombre de comportements cellulaires, comme la division cellulaire ou les changements de forme des cellules. Ces comportements cellulaires jouent un rôle critique au cours du développement embryonnaire puisqu'ils permettent aux cellules, et aux organes, de se positionner dans l'organisme. Ces propriétés mécaniques, et *a fortiori* les comportements cellulaires, sont largement définies par la dynamique d'un réseau de polymères biologiques, le cortex d'actomyosine [LL07]. Chacun de ces polymères, constitué de monomères d'actine, a une dynamique d'élongation, raccourcissement et potentiellement de branchement. Cette dynamique est soumise à une forte stochasticité, encore très peu prise en compte dans les modèles existants. L'objectif de cette thèse est de comprendre la dynamique du réseau de polymères d'actine constituant l'enveloppe mécanique d'une cellule, à l'échelle d'un filament puis à l'échelle du réseau entier, afin de comprendre comment les propriétés mécaniques des cellules émergent de la biochimie de l'actine.

1 - Contexte biologique

Le cytosquelette est un système de filaments protéiques dans le cytoplasme d'une cellule eucaryote, qui donne à la cellule la capacité, entre autres, de changer de forme et de se mouvoir dans l'organisme. Ses composants les plus abondants sont les filaments d'actine, les microtubules et les filaments intermédiaires. L'actine joue donc un rôle essentiel dans la cellule. Les filaments d'actine ont des propriétés mécaniques et une dynamique particulière ; il est donc intéressant d'étudier cette dynamique pour mieux comprendre le fonctionnement de la cellule. Sauf mention contraire, les informations de cette section sont issues du livre *Molecular Biolology of the Cell* [AJL+15].

1.1 - Description de l'actine

La protéine d'actine est un monomère possédant un site de liaison pour un nucléotide de type ATP (Adénosine Tri-Phosphate) ou ADP (Adénosine Di-Phosphate) selon l'état de phosphorylation du nucléotide (Figure 1). Sous cette forme, on parle d'actine globulaire ou Actine-G et on parlera de monomères ADP (respectivement ATP) lorsque l'actine globulaire est associée à un nucléotide ADP (respectivement ATP). Ces sous-unités d'actine s'assemblent pour produire des polymères d'actine, on parle alors de filaments d'actine ou Actine-F. Les filaments d'actine sont des polymères hélicoïdaux à deux brins. Ils apparaissent comme des structures souples, d'un diamètre de 5 à 9 nm. Dans les cellules non-musculaires, ces filaments sont principalement présents dans le cortex cellulaire, juste en dessous de la membrane plasmique.

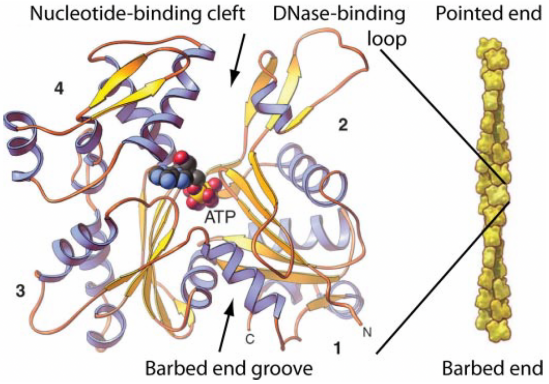


FIGURE 1 *tirée de [PB09], reproduite avec permission.* Structure d'un monomère d'actine et d'un filament d'actine. Les monomères d'actine ont un nucléotide (ATP ou ADP) attaché au coeur de la molécule. Les monomères d'actine s'assemblent suivant une double hélice.

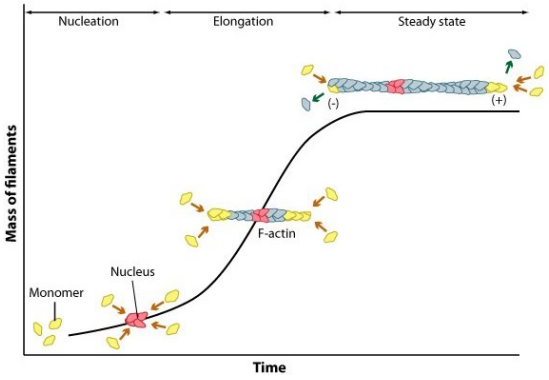


FIGURE 2 *tirée de [mec], reproduite avec permission.* Évolution en fonction du temps de la polymérisation de l'actine.

In vitro, la création d'un polymère d'actine s'effectue en trois phases (Figure 2). Le processus donnant naissance à un filament est appelé nucléation : pour qu'un nouveau filament se forme, les sous-unités doivent s'assembler en un agrégat initial ou noyau. Cette phase peut être assez longue car les dimères d'actine sont instables. L'addition d'un troisième monomère d'actine pour former un trimère stabilise l'ensemble du groupe. Une fois le noyau créé et stabilisé par de nombreux contacts entre les sous-unités, on observe la phase d'élongation, où les filaments s'allongent rapidement. On atteint ensuite une phase d'équilibre : les filaments ne s'allongent plus, chaque polymérisation est suivie d'une dépolymérisation. Les filaments gardent donc une taille constante alors que les monomères du filament sont sans cesse renouvelés.

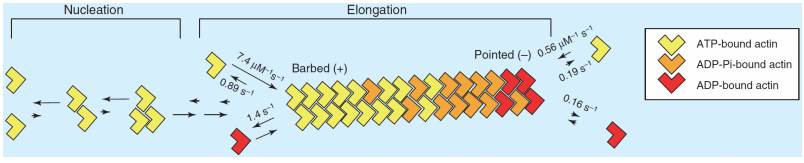


FIGURE 3 *tirée de [NDHH05], reproduite avec permission.* Nucléation et élongation spontanée d'un filament d'actine.

L'orientation régulière des monomères donne aux filaments d'actine une polarité structurelle. Cette orientation différencie les deux extrémités de chaque polymère, ce qui a des conséquences importantes sur la vitesse d'élongation. L'extrémité où la croissance

est la plus rapide est appelée extrémité plus ou barbée. L'autre est appelée extrémité moins ou pointue. Les vitesses d'élongation et de dépolymérisation dépendent également de la nature du monomère : un monomère associé à un nucléotide ATP s'accroche plus rapidement à l'extrémité plus du filament qu'un monomère ADP. À l'inverse, un monomère ATP se détache plus lentement de l'extrémité moins du filament qu'un monomère ADP (Figure 3). Cette dynamique admet un état d'équilibre appelé *treadmilling*. Cet équilibre est atteint lorsque la concentration de monomères libres atteint une valeur critique appelée concentration critique. À chaque extrémité, le polymère va cesser de grandir lorsque la concentration de monomères libres atteint la valeur $C = \frac{k_{Doff}}{k_{Ton}}$ où k_{Doff} est le taux de dépolymérisation d'un ADP et k_{Ton} est le taux de polymérisation d'un ATP. La valeur de C étant différente à l'extrémité plus (C^+) et à l'extrémité moins (C^-), il existe une concentration critique comprise entre C^- et C^+ où les sous-unités subissent à la même vitesse, un assemblage sous forme ATP à l'extrémité plus, et un désassemblage sous forme ADP à l'extrémité moins. Le filament garde alors une longueur constante alors qu'il y a un flux de monomères à travers le polymère : on parle de « vissage par vis sans fin ». La durée pendant laquelle un monomère reste dans un filament est appelée temps de *turnover*. La dynamique d'assemblage et de désassemblage de l'actine permet une modification rapide du cytosquelette.

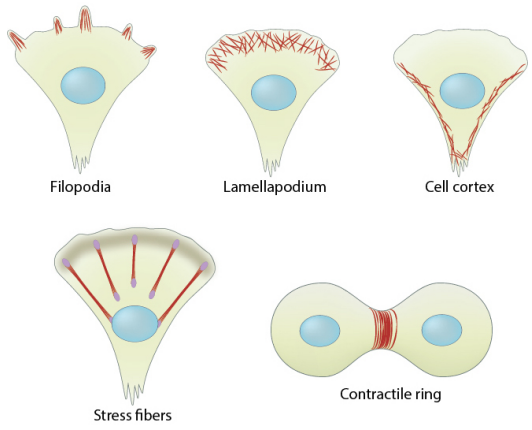


Figure 4 *tirée de [mec], reproduite avec permission.* Différentes dispositions de l'actine dans la cellule. Les filaments d'actine sont représentés en rouge.

Les filaments d'actine des cellules animales sont organisés en deux types de motifs : des faisceaux et des réseaux de type gel. Il est possible de passer dynamiquement d'un type à l'autre. Ces différentes structures sont initiées par l'action de protéines de nucléation distinctes et apportent des propriétés mécaniques différentes à la cellule. La Figure 4 montre ces différentes dispositions. Les fibres de stress sont contractiles et exercent une tension. Le cortex d'actine se trouve sous la membrane plasmique et se compose de réseaux de type gel ou de réseaux dendritiques d'actine qui permettent la protrusion de la membrane au niveau des lamellipodes (large extension membranaire). Les filopodes sont des projections en pointe de la membrane plasmique qui permettent à la cellule d'explorer son environnement.

1.2 - Rôle de l'actine

Les filaments d'actine déterminent la forme de la surface cellulaire et sont nécessaires pour la locomotion des cellules entières. À l'échelle de la cellule, les filaments d'actine sont des structures presque droites mais ils peuvent se déformer [BBPSP14]. La force à exercer pour courber le filament dépend de la longueur totale du polymère (Figure 5). Les propriétés mécaniques et structurelles des cellules sont donc directement influencées par la longueur des filaments d'actine.

Associée à la myosine, l'actine fournit la force de contraction aussi bien aux cellules musculaires (cellules dont la fonction est de se contracter) qu'aux cellules non-musculaires (Figure 6). Le mécanisme de contraction-extension induit par l'actomyosine est à la base des déplacements cellulaires, de la division cellulaire et de la morphogénèse [MOLG15]. Dans les cellules des muscles striés, la myosine-II s'associe aux filaments d'actine pour former des structures fibreuses appelées sarcomères. Le cortex d'actomyosine des cellules non-musculaires et les cellules des muscles lisses sont organisées de dif-

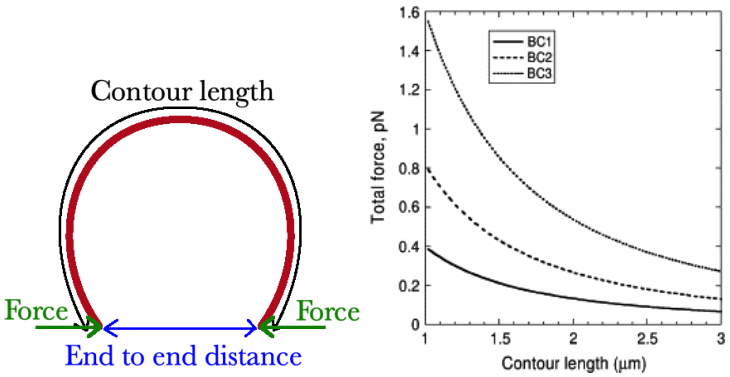


FIGURE 5 *tirée de [BMB+07], reproduite avec permission.* La force à exercer pour courber un filament d'actine varie en fonction de sa longueur. (BC1, BC2 et BC3 correspondent à 3 conditions initiales sur les forces exercées sur le polymère.)

férentes manières qui ne sont pas observées dans les sarcomères. Ces structures contractiles appelées fibres de stress, sont plus fines et moins organisées que les sarcomères.

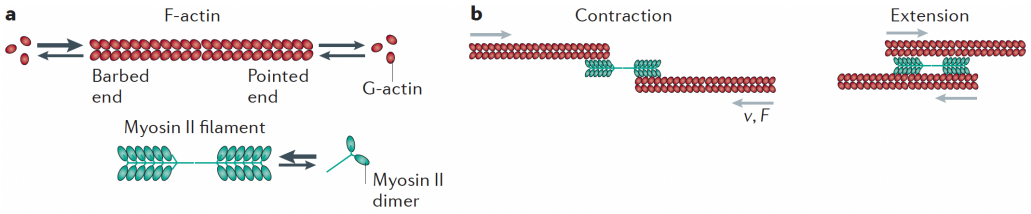


FIGURE 6 *tirée de [MOLG15], reproduite avec permission.* Contraction induite par l'actomyosine. **a.** L'actine-F (en rouge) possède une extrémité barbée et une extrémité pointue indiquées respectivement par le sens ouvert et fermé des chevrons d'actine qui constituent le polymère. Les taux d'association plus élevés de l'actine-G à l'actine-F sont indiqués par une flèche plus grande. Les filaments de myosine (en vert) sont bipolaires avec une zone centrale sans tête moteur et sont obtenus par assemblage de dimères de myosine-II. **b.** Les filaments de myosine entraînent le déplacement des filaments d'actine vers leur extrémité barbée. Il peut en résulter une contraction ou une extension de deux filaments d'actine liés, selon la position de la myosine.

Les filaments d'actine forment plusieurs types de projections cellulaires de surface. Certaines d'entre elles sont des structures dynamiques, comme les lamellipodes et les filopodes que certaines cellules utilisent pour explorer leur environnement et y progresser. La Figure 7 explique comment les filaments d'actine et la création d'un lamellipode par la polymérisation permettent à la cellule de se déplacer : la protrusion et la fixation ferme du lamellipode font avancer le bord de la cellule et étirent le cortex d'actine, la contraction à l'arrière de la cellule propulse alors le corps de la cellule vers l'avant pour relâcher une partie de la tension.

Comme le montre la Figure 8, l'actine (représentée en rouge), joue un rôle essentiel dans la division cellulaire puisque l'anneau contractile, qui se resserre jusqu'à couper la cellule mère en deux cellules filles, est constitué de filaments d'actine.

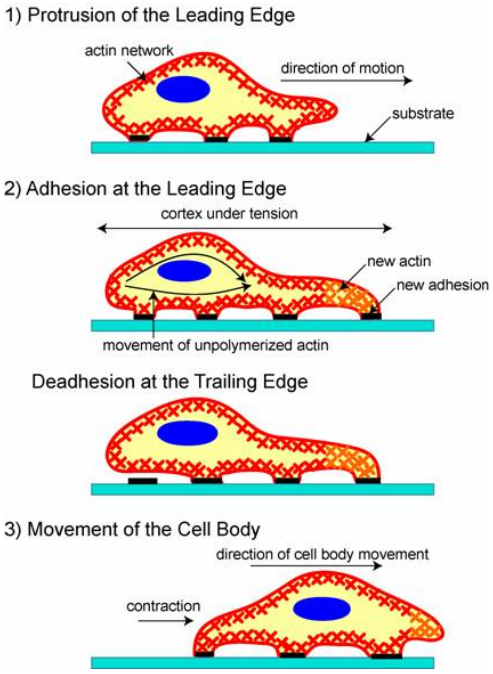


Figure 7 *tirée de [AE07], reproduite avec permission.* Modèle montrant comment les forces engendrées dans le cortex riche en actine peuvent déplacer une cellule. La protrusion et la fixation ferme du lamellipode, qui dépendent de la polymérisation de l'actine, font avancer le bord de la cellule et étirent le cortex d'actine. La contraction à l'arrière de la cellule propulse le corps de la cellule vers l'avant pour relâcher une partie de la tension (traction). À mesure que la cellule avance, de nouveaux contacts ponctuels sont établis à l'avant et les anciens sont défaits à l'arrière. L'actine corticale nouvellement polymérisée est représentée en orange.

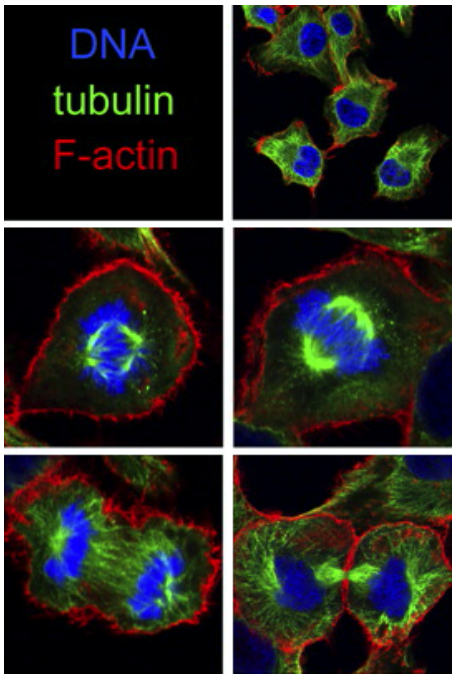


Figure 8 *tirée de [MBS+07], reproduite avec permission.* Changements de l'organisation du cytosquelette lors de la division cellulaire. Lorsque la cellule se divise, le réseau de microtubules polarisés (en vert) se réorganise pour former un fuseau mitotique bipolaire, responsable de l'alignement puis de la ségrégation des chromosomes dupliqués (en bleu). Les filaments d'actine (en rouge) forment un anneau contractile au centre de la cellule qui pince la cellule en deux. Une fois la division cellulaire achevée, les deux cellules filles réorganisent à la fois les microtubules et le réseau d'actine pour reformer la structure du cytosquelette de la cellule mère (en version réduite), ce qui leur permet de poursuivre leur chemin séparément.

1.3 - Protéines accessoires

Plusieurs protéines interagissent avec l'actine et modifient sa dynamique. On ne présente dans cette partie que les protéines que nous prendrons en compte dans notre étude.

Le complexe Arp2/3 (formé notamment des protéines Arp2 et Arp3) facilite la nucléation, se fixe sur les filaments d'actine et permet de créer un branchement (Figure 9). Il en résulte une ramification qui croît à un angle de 70° relativement au filament originel [PC09]. Des cycles répétés de nucléation par ramification forment un réseau arborescent de filaments d'actine. L'activité régulée

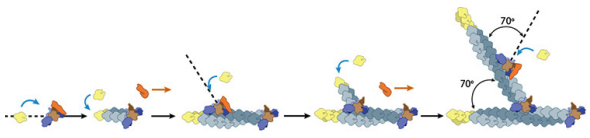


Figure 9 *tirée de [mec], reproduite avec permission.* Nucléation par le complexe Arp2/3.

du complexe Arp2/3 dans les cellules animales tend à conduire à l'assemblage de réseaux d'actine ramifiés en forme de gels. Dans la cellule, on trouve aussi de grandes structures d'actine constituées de faisceaux parallèles de filaments non branchés. Ces faisceaux d'actine sont créés par d'autres ensembles de protéines de nucléation : les formines.

Les formines sont une grande famille de protéines dimériques qui interviennent dans la polymérisation de l'actine et qui s'associent avec l'extrémité plus du filament. Chaque sous-unité de formine a un site de liaison pour l'actine monomérique. La formine permet d'accélérer l'élongation en capturant les monomères. Tant que le filament s'allonge, le dimère de formine reste associé à l'extrémité barbée qui grandit rapidement, tout en permettant la liaison de nouvelles sous-unités d'actine pour allonger le filament (Figure 10). Certains membres de la famille des formines possèdent des domaines non structurés qui contiennent plusieurs sites de liaison pour la profiline. Si une telle formine est liée à un filament, l'élongation n'est possible qu'avec un complexe actine-G/profiline. La profiline est une protéine qui se fixe aux monomères libres d'actine. Une fois le complexe fixé au filament, la profiline se détache du monomère.

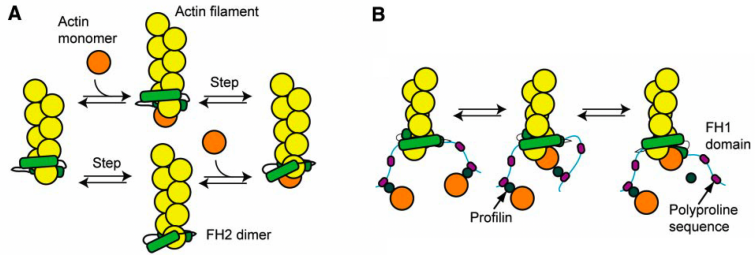


FIGURE 10 *tirée de [PB09], reproduite avec permission.* Elongation de l'actine par l'intermédiaire des formines. **A.** Deux voies d'addition des monomères d'actine. Dans la voie supérieure, une sous-unité d'actine s'associe à l'extrémité plus avant que le domaine FH2 de la formine ne se fixe sur la nouvelle sous-unité. Dans la voie inférieure, la formine s'écarte de l'extrémité avant que la nouvelle sous-unité ne se lie. **B.** Transfert de l'actine ancrée par la profiline sur le domaine FH1 de la formine sur l'extrémité plus du filament, suivi par la dissociation de la profiline.

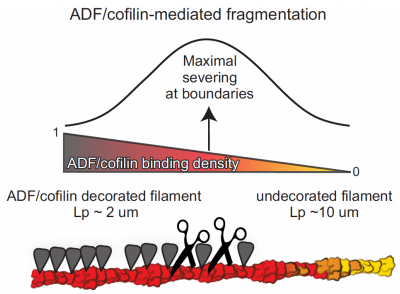


FIGURE 11 *tirée de [BBPSP14], reproduite avec permission.* Fragmentation par la cofiline. La fragmentation du filament se produit à la limite entre les parties nues et les parties décorées par la cofiline.

La cofiline est une protéine qui peut se fixer sur chaque monomère d'un filament d'actine. Cette protéine recouvre le filament et change ses propriétés mécaniques. Les zones recouvertes de cofiline étant moins souples, le filament casse et les deux parties étant instables, se dépolymérisent totalement. La cofiline accélère donc la dépolymérisation. On peut noter que les cassures ont lieu aux transitions entre les parties recouvertes de cofiline et les parties sans [BBPSP14]. Il en résulte donc qu'à partir d'une certaine quantité de cofiline fixée sur le filament, celui-ci redevient moins cassant (Figure 11).

Les protéines de coiffe se fixent à l'extrémité plus des filaments et empêchent la polymérisation. Elles ralentissent donc considérablement la vitesse d'élongation. Ces protéines ne se détachent qu'après dépolymérisation complète du filament.

1.4 - De l'actine à la morphogénèse

Comme nous l'avons déjà mentionné, l'actine intervient à différentes échelles. Les monomères d'actine s'assemblent pour former les filaments d'actine, principaux constituant du cortex cellulaire. Le cortex définit les propriétés mécaniques des cellules constituant les embryons. Afin de mieux com-

prendre la morphogénèse, nous devons donc étudier les mécanismes de la cellule et pour cela, nous cherchons à comprendre comment la biochimie de l'actine influence la structure et la dynamique du cortex cellulaire.

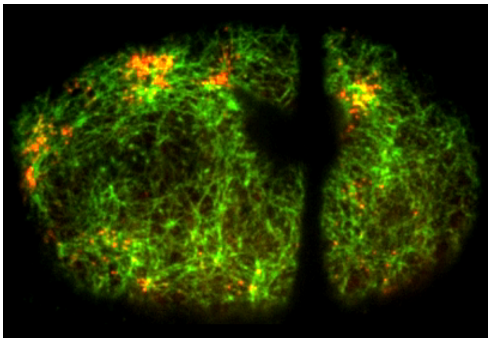


Figure 12 Molécule d'actine marquée avec une GFP formant un réseau de filaments visible par fluorescence dans un embryon de *C. elegans* au stade deux cellules (une cellule mesure environ $50\mu m$). On observe la myosine (en rouge) et le réseau de filaments d'actine (en vert) formant le cortex cellulaire.

L'équipe CADMO (Cortical Actomyosin Dynamics in Development and Morphogenesis) du laboratoire de biologie de développement de l'IBPS (Institut de Biologie Paris-Seine) avec qui j'ai collaboré tout au long de mon doctorat se concentre sur l'étude d'un système biologique spécifique et assez simple : l'embryon de ver nématode *C. elegans* au stade d'une cellule. Les concentrations des protéines intervenant dans la dynamique de l'actine sont mesurées par microscopie de fluorescence dans des embryons exprimant la protéine endogène marquée avec une GFP (Green Fluorescent Protein) par la technologie CRISPR/Cas9 (Courtes répétitions palindromiques groupées et régulièrement espacées) (Figure 12).

Au cours de ma première année de doctorat, j'ai réalisé un stage de quatre mois au sein de cette équipe. J'ai ainsi pu apprendre à utiliser le microscope de fluorescence par réflexion totale interne (TIRF microscopy, Figure 13) qui permet d'observer les protéines marquées d'une GFP se situant dans le cortex cellulaire. J'ai également découvert comment ces images (obtenues grâce à la microscopie de fluorescence) étaient traitées afin de récupérer des données telles que des concentrations ou des vitesses de réaction. Ceci m'a permis de savoir quelles sont les données disponibles et celles qui sont potentiellement mesurables et ainsi d'adapter ma modélisation.

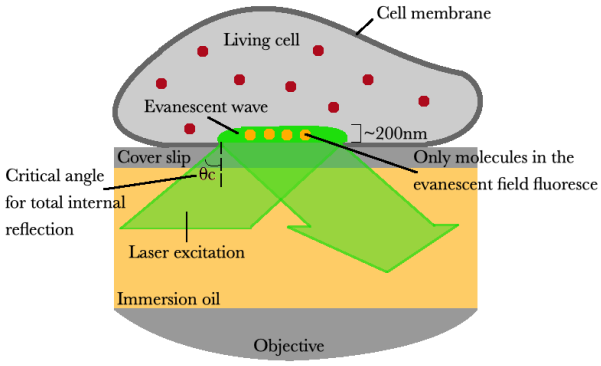


Figure 13 La microscopie TIRF permet la détection de molécules individuelles fluorescentes. La microscopie TIRF utilise la lumière laser excitatrice pour éclairer la surface de la lamelle à l'angle critique auquel toute la lumière est réfléchie par l'interface verre-eau. Une partie de l'énergie électromagnétique traverse l'interface sur une courte distance sous la forme d'une onde évanescente qui n'excite que les molécules fixées à la lamelle ou très proches de sa surface.

2 - Modèles existants

Plusieurs modélisations de la dynamique d'assemblage de l'actine ont déjà été développées [BODM04, EKE98, EEK98, YB09, YDB08]. La majorité de ces modèles sont soit déterministes et se concentrent sur un état d'équilibre, soit stochastiques et sont étudiés par des simulations de Monte-Carlo [Bin10, PB09]. La plupart des modèles prenant en compte une population de polymères d'actine reposent sur des

hypothèses très spécifiques et ne sont pas développés dans un contexte général. En effet, beaucoup d'entre eux se concentrent sur la motilité cellulaire et donc sur la dynamique de l'actine dans les protrusions [BODM04, MEK02a] ou sur l'organisation géométrique du réseau d'actine [CEK94]. Les modèles plus généraux, comme ceux de [EKE98, EEK98], ne prennent pas en compte l'élongation rapide des polymères liés à une formine ni l'effet du changement de vitesse, entre l'élongation spontanée et l'élongation avec une formine. Les travaux [HO11] et [MLH$^+$17] utilisent la même approche que la nôtre pour étudier la longueur moyenne d'un polymère d'actine et la distribution des longueurs de polymères. Cependant, [HO11] ne considère pas la limite en grande population de leur modèle et [MLH$^+$17] se concentre davantage sur l'importance d'avoir deux types de monomères (ATP et ADP) que sur les effets des protéines accessoires. En dehors des approches de population énumérées ci-dessus, la plupart des modèles existants dans la littérature tendent à se concentrer sur un seul polymère, ou sur deux polymères concurrents, avec l'idée que de très grandes populations de polymères seront alors bien décrites par la loi d'un polymère « moyen ».

Certains modèles s'intéressent particulièrement à la corrélation entre les propriétés mécaniques des cellules et la longueur des polymères d'actine. Dans [LGD12], ils s'intéressent à la contractilité des faisceaux d'actomyosine et utilisent pour cela la longueur de persistance des polymères (propriété mécanique permettant de caractériser la rigidité d'un polymère linéaire).

Les modélisations de la dynamique des polymères d'actine ramifiés sont rarement étudiées de manière analytique. En général, il s'agit de modèles étudiés par simulations comme [Car01] qui propose un modèle stochastique prenant en compte l'élongation spontanée, la dépolymerisation, le branchement, le détachement des branches et les protéines de coiffe. Cette étude permet d'observer la structure en 3D du réseau d'actine et d'obtenir la vélocité de croissance, l'espacement des branches et la densité du réseau de polymères d'actine. Les modèles analytiques existants ne permettent pas d'obtenir des informations sur la topologie d'un polymère d'actine ramifié (e.g. nombre de branchements, tailles des branches) et se concentrent souvent sur la force exercée par le réseau d'actine sur la membrane cellulaire. Par exemple, dans [MB01] un polymère d'actine ramifié est modélisé à l'aide d'un processus d'évolution mais cela ne permet d'obtenir que l'orientation de l'actine au bord de la cellule et la force exercée par le réseau.

3 - Modèles préliminaires

3.1 - Modélisation d'un unique polymère d'actine : première approche

J'ai découvert l'actine et ses problématiques dans le cadre de mon projet de premier semestre de master 2 (en binôme avec Laura Hedon). Au cours de ce projet, nous avons cherché comment modéliser la dynamique d'un unique polymère d'actine. Nous nous sommes alors concentrées sur la dynamique d'élongation-raccourcissement spontanée sans différencier les monomères ATP et ADP. Pour introduire une composante stochastique, inhérente à l'aléa des rencontres et fixations entre les différentes entités en jeu, nous avons choisi d'utiliser la théorie des files d'attente. L'idée principale de notre modèle consiste à voir un polymère d'actine comme une file d'attente (Figure 14).

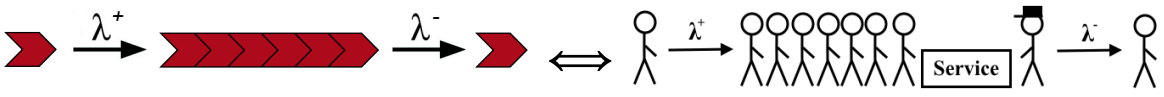


FIGURE 14 Modèle préliminaire pour la dynamique d'élongation-raccourcissement spontanée d'un unique polymère d'actine.

Le taux d'élongation λ^+ correspond alors au taux d'arrivée des clients alors que le taux de dépolymérisation λ^- correspond au taux de service. La longueur du polymère en nombre de monomère correspond au nombre de clients dans la file d'attente et le temps de turnover correspond à la somme du temps d'attente et du temps de service d'un client. Ce premier modèle nous a permis de déterminer l'espérance et la variance de la longueur du polymère et du temps de turnover dans les trois cas : sous-critique ($\lambda^+ < \lambda^-$), critique ($\lambda^+ = \lambda^-$) et sur-critique ($\lambda^+ > \lambda^-$). Bien que le cas sur-critique soit acceptable pour modéliser la dynamique d'élongation rapide de l'actine en réponse à un signal extérieur, ce modèle reste très limité. Afin de l'améliorer, il faudrait tout d'abord prendre en compte que l'élongation semble être limitée par la quantité de monomères d'actine. Pour cela, il faudrait choisir un taux d'élongation proportionnel à la quantité de monomères disponibles mais aussi considérer plusieurs polymères d'actine ; la dynamique sera modifiée s'il existe une compétition entre les polymères pour les monomères. Ces améliorations ont fait l'objet de mon stage de second semestre de master 2.

3.2 - Modélisation d'une population de polymères d'actine : première approche

L'objectif de mon stage de master 2 était de modéliser la dynamique d'une population de polymères d'actine en prenant en compte uniquement l'élongation, la dépolymérisation et la nucléation (Figure 15).

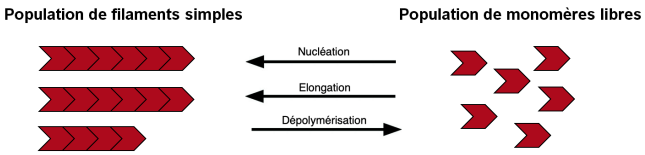


Figure 15 Modèle préliminaire pour une population de polymères d'actine prenant en compte l'élongation, la dépolymérisation et la nucléation.

Pour modéliser une population de polymères, j'ai utilisé un processus à valeurs mesure. Afin que ce modèle soit davantage cohérent avec la biologie, j'ai choisi cette fois des taux d'élongation et de nucléation proportionnels à la concentration en monomères d'actine disponibles. J'ai alors étudié la tension de ce processus et obtenu une limite en grande population. J'ai de plus remarqué que cette limite déterministe admet un état d'équilibre qui coïncide avec le résultat de [EKE98]. Ceci nous a permis d'obtenir la distribution en taille de polymères à l'équilibre et de conclure que les polymères courts, présents en grande quantité, jouent un rôle significatif dans la dynamique d'élongation-raccourcissement de l'actine.

4 - Dynamique d'élongation-raccourcissement

Le premier chapitre de cette thèse est consacré à l'étude de la dynamique d'élongation-raccourcissement de l'actine en présence de deux protéines, la formine et la profiline, qui permettent d'accélérer l'élonga-tion du polymère d'actine. Cette étude s'intéresse à deux échelles : dans un premier temps on se place à l'échelle d'un unique polymère et dans un second à l'échelle d'une population de polymères. L'ensemble de ce travail fait l'objet de la publication [RVGV21].

4.1 - Dynamique d'un polymère d'actine

Le premier modèle du chapitre I prend en compte un unique polymère d'actine pouvant s'allonger ou se raccourcir spontanément ou à l'aide de la formine et de la profiline. Il y a donc deux vitesses d'élongation et deux types de ressources : les monomères d'actine disponibles permettant l'élongation spontanée et les complexes actine-G/profiline permettant l'élongation à l'aide d'une formine (Figure 1.1). Ce problème peut être vu comme une file d'attente à deux vitesses de service [BR19] mais nous avons préféré utilisé le formalisme des problèmes d'homogénéisation.

L'objectif de cette partie est de trouver un comportement moyen afin de déterminer s'il y a une compétition pour les monomères d'actine, de déterminer pour quels jeux de paramètres la longueur du polymère atteint un état d'équilibre et dans ces cas, déterminer quelle est la longueur moyenne du polymère à l'équilibre. Afin de simplifier la dynamique, nous avons émis plusieurs hypothèses :

- On ne prend pas en compte la nucléation. On démarre avec un polymère de longueur supérieure ou égale à 3 et on regarde uniquement la phase de croissance et la phase d'équilibre. Si le polymère disparaît, il ne peut pas être recréé ; 0 est donc un état d'équilibre pour la longueur du polymère.
- Afin de simplifier la dynamique, on suppose que la polymérisation a lieu uniquement au bout barbu et la dépolymérisation au bout pointu. Cela nous permet de voir un polymère d'actine comme une file d'attente (comme nous l'avons fait dans notre modèle préliminaire). Nous ne différencions pas les monomères ATP et ADP.
- Les protéines accessoires (formines et profilines) ne sont pas considérées comme un facteur limitant.
- Tant qu'une formine est fixée à l'extrémité barbée du polymère, on suppose que l'élongation ne peut pas avoir lieu avec un monomère libre, il faut nécessairement un complexe actine-G/profiline. On suppose également que le taux de dépolymérisation est le même avec ou sans formine.
- Un polymère de taille 1 n'est pas un monomère libre et peut donc s'allonger.

Nous nous intéressons dans cette partie à un seul polymère d'actine, formé d'un nombre de monomères qui fluctue au cours du temps. Les fluctuations de la longueur du polymère dépendent de la concentration en monomères libres et de la présence éventuelle de deux protéines — la formine et la profiline — qui agissent conjointement pour accélérer l'élongation des polymères et ce uniquement durant la période où une formine est fixée au polymère.

Dans ce modèle, nous considérons qu'il y a un nombre fixe et fini N de monomères disponibles et deux modes, correspondant aux deux vitesses d'élongation (avec ou sans formine). La dynamique d'élongation-raccourcissement des polymères est alors entièrement décrite par le processus markovien de sauts $(L^N(t), M(t))_{t\in\mathbb{R}_+} = \left(L_0^N(t), L_1^N(t), L_2^N(t), M(t)\right)_{t\in\mathbb{R}_+}$ où $L_0^N(t)$, $L_1^N(t)$ et $L_2^N(t)$ sont respectivement le nombre de monomères libres, la longueur du polymère et le nombre de complexes protéiques actine-G/profiline au temps t et où $(M(t))_{t\in\mathbb{R}_+}$ est un processus Markovien sur $\{0,1\}$ déterminant le mode d'élongation .

Afin d'obtenir un comportement moyen, nous avons étudié la limite quand le nombre de monomères N tend vers l'infini, ce qui correspond alors à une limite de champs moyen. Après avoir prouvé que notre processus (renormalisé) est tendu (Proposition 1.3.2) et admet une unique limite (Chapitre I Section 3.2), nous avons prouvé que notre processus, dépendant de N, converge en loi pour la topologie de Skorokhod vers un processus déterministe quand N tend vers l'infini (Théorème 1.2.1).

Théorème 1.2.1
Soit $\Delta = \left\{(x_1, x_2, x_3) \in [0,1]^3 \mid x_1 + x_2 + x_3 = 1\right\}$.

On suppose que la condition initiale $\overline{L^N}(0) \in \Delta$ converge en distribution vers un vecteur déterministe $(l_0(0), l_1(0), l_2(0)) \in \Delta$.

Alors, lorsque $N \longrightarrow \infty$, le processus $(\overline{L^N}(t))_{t\in\mathbb{R}_+}$ converge en distribution dans $D_\Delta[0,\infty[$ vers le processus déterministe $(l(t))_{t\in\mathbb{R}_+} = (l_0(t), l_1(t), l_2(t))_{t\in\mathbb{R}_+}$ vérifiant : pour tout $t \leq t_0$,

$$\begin{cases} l_0(t) = \left(l_0(0) - \frac{\lambda^-}{C_0}\right)e^{-C_0 t} + \frac{\lambda^-}{C_0} \\ l_1(t) = 1 - l_0(t) - l_2(t) \\ l_2(t) = l_2(0)e^{-C_2 t} + \frac{\Phi_P}{C_2 - C_0}\left(l_0(0) - \frac{\lambda^-}{C_0}\right)(e^{-C_0 t} - e^{-C_2 t}) + \frac{\Phi_P \lambda^-}{C_0 C_2}(1 - e^{-C_2 t}), \end{cases}$$

et pour tout $t > t_0$,

$$\begin{cases} l_0(t) = l_0(t_0)e^{-\Phi_P(t-t_0)} \\ l_1(t) = 0 \\ l_2(t) = 1 - l_0(t) \end{cases}$$

où $t_0 = \min\{t \in \mathbb{R}_+ \mid l_1(t) = 0\}$ est le temps auquel la longueur du filament atteint 0 (par convention, $t_0 = +\infty$ si l_1 n'atteint pas 0).

Tout d'abord, nous avons remarqué qu'il s'agissait d'un problème d'homogénéisation puisque la limite obtenue correspond à la somme des limites de chaque mode pondérée par la probabilité d'être dans ce mode. Finalement, l'étude du processus déterministe obtenu nous permet d'obtenir une longueur moyenne de polymère mais aussi de déterminer le rôle du changement de mode dans l'utilisation de la ressource, à savoir ici les monomères libres.

4.2 - Dynamique d'une population de polymères

Pour le second modèle du chapitre I, nous nous plaçons désormais à l'échelle d'une population de polymères. Chaque polymère évolue avec la dynamique décrite dans le modèle précédent (Figure 1.2). Les fluctuations de la longueur des polymères dépendent alors toujours de la concentration en monomères libres et en différentes protéines mais aussi des interactions entre les différents polymères.

L'objectif de cette partie est de trouver un comportement moyen afin de savoir s'il existe une compétition entre les polymères pour les monomères libres, de déterminer la quantité d'actine polymérisée et de déterminer la distribution en longueur des polymères d'actine à l'équilibre. Nous avons émis les mêmes hypothèses que pour le modèle précédent.

Dans ce modèle, nous considérons comme dans le cas précédent que le nombre de monomères est fixe et fini égal à N et nous ne prenons en compte que les protéines formine et profiline. La dynamique de la population est cette fois décrite par le processus de Markov à valeurs mesure :

$$\left(\mathcal{Z}_t^N\right)_{t\in\mathbb{R}_+} = \left(K_t^N(M,1)\delta_{(M,1)} + K_t^N(P,1)\delta_{(P,1)} + \sum_{l=1}^{N} K_t^N(S,l)\delta_{(S,l)} + \sum_{l=1}^{N} K_t^N(F,l)\delta_{(F,l)}\right)_{t\in\mathbb{R}_+}$$

où $K_t^N(T,\ell)$ est le nombre de polymères de type T et de longueur ℓ au temps t ; les différents types étant : les monomères libres (M), les polymères simple (S), les polymères associés à une formine (F) et les complexes actine-G/profiline (P).

Cette fois encore, nous avons cherché un comportement moyen. En considérant qu'il y a un nombre de polymères de l'ordre de N dans la population et en étudiant la limite quand N tend vers l'infini, nous avons obtenu une limite en grande population. Après avoir prouvé que notre processus (renormalisé) est tendu (Proposition 1.4.1), nous prouvons en utilisant le théorème 4.8.10 de [EK86] que nous avons de nouveau une convergence en loi pour la topologie de Skorokhod quand N tend vers l'infini, cette fois vers un processus déterministe à valeurs mesure (Théorème 1.2.2).

THÉORÈME 1.2.2
Soit $\mathcal{I} = \{(M,1),(P,1)\} \cup \big(\{S,F\} \times \mathbb{N}\big)$ et soit $\mathcal{M}_{\mathcal{I}}^{(1)} = \big\{ Z \in \mathcal{M}_{\mathcal{I}} \mid \langle Z, 1\rangle \leq 1 \big\}$.

On suppose que la condition initiale $\overline{\mathcal{Z}_0^N}$ converge en distribution vers une limite déterministe $\mathcal{Z}_0 \in \mathcal{M}_{\mathcal{I}}^{(1)}$ lorsque $N \longrightarrow +\infty$, et qu'il existe $K_0 \in \mathbb{N}$ tel que pour tout N suffisamment grand,

$$\langle \overline{\mathcal{Z}_0^N}, \mathbb{1}_{\{l \geq K_0\}} \rangle = 0.$$

Alors le processus $\big(\overline{\mathcal{Z}_t^N}\big)_{t\in\mathbb{R}_+}$ converge en distribution dans $D_{\mathcal{M}_{\mathcal{I}}^{(1)}}[0,\infty[$ lorsque $N \longrightarrow +\infty$, vers une limite déterministe $(\mathcal{Z}_t)_{t\in\mathbb{R}_+}$ qui peut être caractérisée de la façon suivante : si pour tout $(T,l) \in \mathcal{I}$ on écrit $k_t(T,l)$, la masse $\mathcal{Z}_t(\{(T,l)\})$ donnée par la mesure $\mathcal{Z}_t$ au point (T,l), alors

$$k_0(T,l) = \mathcal{Z}_0(\{(T,l)\}),$$

et

$$\begin{cases}
\frac{d}{dt}k_t(M,1) &= \lambda^- \sum_{l=1}^{+\infty}(k_t(S,l)+k_t(F,l)) - \lambda^+ k_t(M,1)\sum_{l=1}^{+\infty} k_t(S,l) - \Phi_P k_t(M,1)\\
\frac{d}{dt}k_t(P,1) &= \Phi_P k_t(M,1) - \lambda_F^+ k_t(P,1)\sum_{l=1}^{+\infty} k_t(F,l),\\
\frac{d}{dt}k_t(S,1) &= \Phi_F^- k_t(F,1) + \lambda^- k_t(S,2) - (\lambda^+ k_t(M,1) + \lambda^- + \Phi_F^+)k_t(S,1)\\
\frac{d}{dt}k_t(F,1) &= \Phi_F^+ k_t(S,1) + \lambda^- k_t(F,2) - (\lambda_F^+ k_t(P,1) + \lambda^- + \Phi_F^-)k_t(F,1)\\
\text{et } \forall l \geq 2,&\\
\frac{d}{dt}k_t(S,l) &= \lambda^+ k_t(M,1)k_t(S,l-1) + \Phi_F^- k_t(F,l) + \lambda^- k_t(S,l+1) - (\lambda^+ k_t(M,1) + \lambda^- + \Phi_F^+)k_t(S,l)\\
\frac{d}{dt}k_t(F,l) &= \lambda_F^+ k_t(P,1)k_t(F,l-1) + \Phi_F^+ k_t(S,l) + \lambda^- k_t(F,l+1) - (\lambda_F^+ k_t(P,1) + \lambda^- + \Phi_F^-)k_t(F,l).
\end{cases}$$

L'étude de ce comportement moyen nous permet d'obtenir la distribution en longueur des polymères mais surtout de constater qu'il existe une compétition entre les polymères d'actine pour les monomères libres. La distribution en longueur des polymères est particulièrement importante puisque les propriétés mécaniques des cellules dépendent directement de la taille des polymères d'actine.

5 - Polymères ramifiés

En plus de la dynamique d'élongation-raccourcissement, les polymères ont également la capacité de se ramifier. Le modèle du chapitre II prend en compte un unique polymère d'actine pouvant s'allonger spontanément, se ramifier à l'aide du complexe Arp2/3 ou se fragmenter à l'aide de la cofiline (Figure 2.1). Le polymère d'actine n'est donc plus rectiligne mais ramifié et pour le modéliser nous avons besoin d'un processus à valeurs arbre.

Trouver le bon cadre de travail pour modéliser la dynamique des polymères ramifiés s'est révélé être une tâche difficile. Notre première idée fut de modéliser les branchements et les fragmentations en

s'inspirant des modèles utilisant les sous-arbres "prune and re-graft". Nous avons trouvé assez rapidement comment étudier la convergence de tels processus [EW06, HW14] mais la question de comment modéliser un arbre est longtemps rester en suspend. Dans la majorité des cas, comme dans [HMPW08], la croissance des branches n'est pas prise en compte. Afin de prendre en compte l'élongation, chaque branches menant à une feuille doit pouvoir croître de manière aléatoire. Le modèle présenté dans [EPW06] prend en compte une croissance déterministe des branches. Ils montrent par ailleurs que le processus répétant : choix uniforme d'un point de coupe puis croissance déterministe d'une nouvelle branche au point de coupe, est équivalent au processus répétant : choix uniforme d'un point de coupe puis "prune-off" de la partie au dessus du point de coupe puis "re-graft" de ce sous-arbre à la racine puis croissance déterministe de la racine. Ceci facilite donc le processus d'élongation puisque seule la racine peut croître. Cependant, s'il est simple de rendre le processus d'élongation aléatoire, nous n'avons pas trouver comment conserver la taille de chaque branche au cours du temps. C'est finalement la thèse de P.K Glöde [Glö13] (utilisant le formalisme des espaces métriques mesurés introduit dans [GPW13]) et l'idée de prendre en compte uniquement les feuilles et la distance généalogique entre elles qui nous a permis de développer notre modèle.

Le modèle développé dans le chapitre II de cette thèse a pour but de mieux comprendre la structure des polymères ramifiés, ce qui pourra par exemple permettre de mieux comprendre la formation des protrusions : extensions membranaires qui permettent aux cellules de se déplacer. Dans cette optique, nous avons décidé de limiter notre cadre d'étude à la dynamique des polymères ramifiées dans les protrusions.

Nous considérons alors que deux protéines peuvent interagir avec l'actine : le complexe protéique Arp2/3 et la cofiline. Alors que Arp2/3 permet de créer un nouveau branchement, la cofiline entraine la fragmentation du polymère et donc la perte de branches. Par ailleurs, l'élongation peut avoir lieu à chaque extrémité du polymère.

La concentration de monomères libres dans les protrusions étant élevée, les monomères libres ne semblent pas être une quantité limitante dans ce cas. Notre paramètre d'échelle N correspond dans ce modèle à l'ordre de grandeur du nombre d'extrémités du polymère (feuilles de l'arbre). Nous supposons de plus que notre polymère est touffu, c'est à dire qu'il a de nombreuses branches très courtes, et dans un premier temps, que les fixations de protéines (et *a fortiori* les branchements et les fragmentations) n'ont lieu qu'aux extrémités du polymère.

La dynamique du polymère ramifié est alors entièrement décrite par le processus à valeurs arbre (Définition 2.4.6):

$$\left(\mathbb{T}_t^N\right)_{t\in\mathbb{R}_+} = \left(\left[\mathfrak{I}_t^N, \mathfrak{r}_t^N, \mathfrak{u}_t^N\right]\right)_{t\in\mathbb{R}_+}$$

où :

- $\mathfrak{I}_t^N$ correspond à l'ensemble des indices des feuilles au temps t,
- $\left(\mathfrak{r}_t^N\right)_{t\in\mathbb{R}_+}$ est un processus à valeurs distance correspondant à la distance généalogique (i.e. le long des branches de l'arbre) entre les feuilles,
- $\left(\mathfrak{u}_t^N\right)_{t\in\mathbb{R}_+}$ est un processus à valeurs mesure donnant une masse $\frac{1}{N}$ à chaque feuille de l'arbre.

Nous notons également $\left(\tilde{X}_t^N\right)_{t\in\mathbb{R}_+} = \left(\mathfrak{u}_t^N\left(\mathfrak{I}_t^N\right)\right)_{t\in\mathbb{R}_+}$ la masse totale de $\left(\mathfrak{u}_t^N\right)_{t\in\mathbb{R}_+}$, qui correspond aussi au nombre de feuilles renormalisé. Ce processus $\left(\tilde{X}_t^N\right)_{t\in\mathbb{R}_+}$ correspond à un processus de naissance et de mort. Chaque feuille s'associe à un complexe Arp2/3 ou à une cofiline à taux $N\beta$ (avec $\beta > 0$), indépendamment des autres feuilles. Lorsqu'une telle fixation de protéine se produit, il s'agit d'un branchement (fixation d'un complexe Arp2/3) à probabilité $p_2^N = \frac{1}{2} + \frac{a}{N}$ (avec $a \in \mathbb{R}$), ou d'une fragmentation (fixation d'une cofiline) à probabilité $p_0^N = \frac{1}{2} - \frac{a}{N}$. Un branchement entraine la création d'une nouvelle feuille alors que la fragmentation entraine la disparition d'une feuille.

Nous avons prouvé que le processus arrêté associé à $\left(\mathfrak{T}_t^N\right)_{t\in\mathbb{R}_+}$ est tendu (Proposition 2.7.1). Afin de prouver la convergence en loi de notre processus à valeurs arbre, nous avons prouvé que $\left(\mathfrak{T}_\cdot^N\right)_{N\in\mathbb{N}}$ admet une sous-suite convergente et que cette sous-suite vérifie un certain problème de martingale (Théorème 2.8.26), puis que ce problème de martingale est bien posé (Théorème 2.8.27).

Théorème 2.8.26
Soit $R \in \mathbb{N}^*$.

On suppose $\limsup_{N\to+\infty} \mathbb{E}\left[\tilde{X}_0^N\right] < \infty$ et $\mathfrak{T}_{0,R}^N \xrightarrow[N\to+\infty]{\mathcal{L}} \nu \in \mathcal{M}_1(\mathbb{T})$ où $\nu(\mathbb{M}_{>0}) = 1$.

Alors, $\left(\mathfrak{T}_{\cdot,R}^N\right)_{N\in\mathbb{N}}$ est relativement compact sur $D_{\mathbb{T}}[0,+\infty[$ et la limite de chaque sous-suite convergente est solution du problème de martingale :

$$\left(\mathcal{G}^\infty,\ {}^{S}H_R,\ \nu\right),$$

où :

- $\mathcal{G}^\infty$ est la limite quand $N \longrightarrow +\infty$ du générateur infinitésimal du processus $\mathfrak{T}^N$,
- ${}^{S}H_R$ est l'espace vectoriel engendré par :

$$H_R = \left\{\Psi \in \Pi_\Psi\left(C_b^{1bc}, C_K^\infty\right) \mid \mathcal{G}^\infty\Psi \equiv 0 \text{ on } \mathbb{T}\left([0,1/R]\right)\right\}.$$

Théorème 2.8.27
Soit $R \in \mathbb{N}^*$.

On suppose $\limsup_{N\to+\infty} \mathbb{E}\left[\tilde{X}_0^N\right] < \infty$ et $\mathfrak{T}_{0,R}^N \xrightarrow[N\to+\infty]{\mathcal{L}} \nu \in \mathcal{M}_1(\mathbb{T})$ où $\nu(\mathbb{M}_{>0}) = 1$.

Alors, le processus arrêté $\mathfrak{T}_R^N$ converge en distribution dans $D_{\mathbb{T}}[0,+\infty[$ quand $N \longrightarrow +\infty$ vers $\mathfrak{T}^\infty = (\mathfrak{T}_t^\infty)_{t\in\mathbb{R}_+}$. De plus, la limite vérifie la propriété de Feller (ce qui signifie que pour toute fonction continue et bornée Φ sur $\mathbb{T}$, la fonction $\nu \mapsto \mathbb{E}_\nu\left[\Phi\left(\mathfrak{T}_t^\infty\right)\right]$ est continue pour la topologie faible sur $\mathcal{M}_1(\mathbb{T})$), la propriété de Markov forte et il existe presque sûrement une version à trajectoires continues.

Par ailleurs, nous avons prouvé que le processus $\left(\tilde{X}_t^N\right)_{t\in\mathbb{R}_+}$ converge en distribution vers une diffusion de Feller (Proposition 2.8.28).

Proposition 2.8.28
On suppose que $\limsup_{N\to+\infty} \mathbb{E}\left[\tilde{X}_0^N\right] < M < +\infty$.

On suppose de plus que $\nu^N = \mathcal{L}\text{oi}\left[\tilde{X}_0^N\right] \xrightarrow[N\to+\infty]{\mathcal{L}} \nu \in \mathcal{M}_1(\mathbb{R}_+)$.

Alors

$$\tilde{X}^N \xrightarrow[N\to+\infty]{\mathcal{L}} \tilde{X}^\infty$$

sur $D_{\mathbb{R}_+}[0,+\infty[$, où $\tilde{X}^\infty$ est une diffusion de Feller solution de :

$$dZ_t = 2a\beta Z_t dt + \sqrt{\beta Z_t} dB_t,$$

et telle que $\mathcal{L}\text{oi}\left[\tilde{X}_0^N\right] = \nu$.

Finalement, nous détaillons les améliorations qu'il serait intéressant de prendre en compte — ainsi que quelques pistes de mise en œuvre — afin de rendre notre modèle plus pertinent biologiquement et plus innovant mathématiquement.

6 - Exploration numérique des modèles

Chaque modèle que nous avons développé a été implémenté en C++ et/ou R afin de pouvoir réaliser des simulations. La Section 5 du Chapitre I se concentre sur un exemple intéressant de comportement obtenu pour un polymère simple puis pour une population de polymères. Cet exemple permet entre autres d'observer l'effet du changement de vitesse d'élongation. En effet, lorsqu'on considère un unique polymère d'actine, on observe que le changement de vitesse d'élongation permet une meilleure utilisation des monomères libres. Nous avons ensuite déterminé le critère (1.56) qui assure que ce phénomène se produit dès lors que Φ_P est suffisamment petit. Lorsqu'on considère une population de polymères d'actine, nous avons constaté que le changement de vitesse ne permet pas d'améliorer l'utilisation des monomères libres mais permet d'obtenir des polymères plus long. Les Sections 2 et 3 du Chapitre III présentent l'exploration numérique qui nous a conduit a cet exemple particulièrement intéressant. Cette étude qualitative des différents comportements pouvant être rencontrés nous permet également d'identifier le rôle des paramètres et donc de chaque protéine. Nous étudions dans la Section 3 les mêmes cas que dans la Section 2 mais dans une population de polymères. Cela nous permet d'exhiber l'effet de la compétition entre les polymères. Le code utilisé pour réaliser l'ensemble de ces simulations est disponible en Annexe A.

La Section 10 du Chapitre II présente succinctement l'effet des différents paramètres sur la topologie du polymère branché. Dans cette partie, une seule réalisation est présentée alors qu'il existe une variabilité plus ou moins importante selon les cas ; l'objectif est donc de donner un caractère concret au modèle et non de proposer une étude qualitative. Dans un second temps, nous observons l'effet des différentes améliorations proposées dans la Section 9 dédiée aux perspectives. Cela nous permet de tester nos hypothèses et de déterminer quelles améliorations influencent significativement le comportement. Le code utilisé pour réaliser l'ensemble de ces simulations est disponible en Annexe B.

Finalement, dans la Section 4 du Chapitre III nous présentons un dernier modèle prenant en compte plus de protéines accessoires : formine, profiline, Arp2/3, cofiline et protéines de coiffe. La dynamique est alors complexe puisque beaucoup de phénomènes sont considérés : élongation et dépolymérisation spontanée, élongation rapide par la formine et la profiline, création de complexe actine-G/profiline, attachement et détachement des formines, attachement et détachement des complexes Arp2/3, capping, nucléation et fragmentation. Ce dernier modèle a pour but d'explorer, par simulations uniquement, les interactions entre les polymères linéaires et les polymères branchés. Dans la volonté d'obtenir des résultats particulièrement proches de la réalité, nous avons fait le lien entre nos paramètres et les valeurs observées en biologie.

L'implémentation de ce modèle (disponible en Annexe C) ayant été plus délicate que prévu, nous n'avons pas eu le temps d'explorer ce modèle autant que nous l'aurions souhaité. Nous présentons toutefois les résultats obtenus pour une réalisation aléatoire en utilisant les paramètres déterminés à partir des données biologiques. Ceci permet entre autres de montrer l'étendue des possibilités offerte par cette implémentation. Nous terminons en présentant une application biologique concrète ainsi qu'une conjecture mathématique que nous avions l'ambition d'étudier mais que nous laissons en perspective.

Chapter I

The Role of Mode Switching in a Population of Actin Polymers with Constraints

Contents

This chapter corresponds to article [RVGV21] "The role of mode switching in a population of actin polymers with constraints" in collaboration with Amandine Véber and François Robin, published in *Journal of Mathematical Biology*.

Corollary 1.4.3 appears here with a minor correction. Indeed, in the published version there is an unwanted coefficient $\overline{F^\infty}(t)$ multiplying the elongation rates (spontaneous and with a formin). System (1.58) and condition (1.59) (resulting from this corollary) have been corrected too. This mistake has no impact on our discussion and conclusion.

1 - Abstract

In this paper, we introduce a stochastic model for the dynamics of *actin* polymers and their interactions with other proteins in the cellular envelop. Each polymer elongates and shortens, and can

switch between several *modes* depending on whether it is bound to accessory proteins that modulate its behaviour as, for example, elongation-promoting factors. Our main aim is to understand the dynamics of a large population of polymers, assuming that the only limiting quantity is the total amount of monomers, set to be constant to some large N.

We first focus on the evolution of a very long polymer, of size $\mathcal{O}(N)$, with a rapid switch between modes (compared to the time-scale over which the macroscopic fluctuations in the polymer size appear). Letting N tend to infinity, we obtain a *fluid limit* in which the effect of the switching appears only through the fraction of time spent in each mode at equilibrium. We show in particular that, in our situation where the number of monomers is limiting, a rapid binding-unbinding dynamics may lead to an increased elongation rate compared to the case where the polymer is trapped in any of the modes.

Next, we consider a large population of polymers and complexes, represented by a random measure on some appropriate type space. We show that as N tends to infinity, the stochastic system converges to a deterministic limit in which the switching appears as a flow between two categories of polymers. We exhibit some numerical examples in which the limiting behaviour of a single polymer differs from that of a population of competing (shorter) polymers for equivalent model parameters.

Taken together, our results demonstrate that under conditions where the total number of monomers is limiting, the study of a single polymer is not sufficient to understand the behaviour of an ensemble of competing polymers.

2 - Introduction

2.1 - Actin Cytoskeleton and Cell Mechanics

During the embryonic development, cell mechanics plays a central role in many cell behaviours, from cell shape changes to cell division, cell migration and cell rearrangements [LL07]. In animals, these mechanical properties are largely determined by the *cortical actomyosin cytoskeleton*, a 200-500nm thick network of actin and myosin polymers assembled beneath the cell membrane. Several parameters play a key role in determining the properties of this 2d active gel, including its turnover rate, density, crosslinking and its fine architecture. In particular, the distribution of polymer lengths in the cytoskeleton has a strong impact on the mechanical properties of the gel and the cell. In this paper, we focus on the dynamics of actin polymer assembly in the presence of two classes of proteins that modulate the assembly dynamics, formins and profilins. Our main assumption is that the total number of monomers in the system is fixed to some (large) N, reflecting the fact that this quantity is a limiting factor in the evolution of the population of polymers.

In cells, actin exists in two forms: monomers (or G-actin) and polymers (or F-actin). The G-actin monomer addition is coupled to an ATP hydrolysis. Along with the structural polar organisation of the F-actin filament, this causes both extremities of a polymer to display distinct kinetic properties: one with fast kinetics, or *plus* end, and one with slow kinetics, or *minus* end. *In vitro* actin assembly can reach a steady state where polymerisation, occurring at the plus end, is balanced by depolymerisation, occurring at the minus end. Note that in this work we are not interested in the spatial conformation of the polymers or their spatial organisation in the cellular cortex: provided there are available free monomers, a polymer can grow irrespective of its size.

A large number of accessory proteins interact with actin polymers and modify their dynamics, thus contributing to the shaping of the actin network in a cell- and tissue-specific manner. Here we consider only two such interactors: formins and profilins. Formins form a large family of dimeric proteins with

a wide range of biochemical properties associated with their function in the cell. These proteins bind to the plus end of actin filaments, drive a *processive* filament elongation and modulate the actin elongation rate [Kov06, KHM+06]. Profilin is an abundant protein which regulates the actin dynamics by promoting ADP-to-ATP exchanges on G-Actin, renewing the ATP-actin pool. Importantly, profilin has been shown to affect differently distinct actin sub-networks, increasing the elongation rate by formins up to 15-fold, while decreasing the branching of branched networks [BCB+14, RWH+15, SCB+15]. In order to disentangle the different mechanisms underlying the composition in distinct polymer structures of the whole population of actin polymers, in this work we do not consider the sub-network of branched filaments (i.e. tree-like structures induced by another protein called Arp2/3) and instead focus on the positive effect of formins and profilins on the elongation rate of the sub-network of simple polymers. Taking branching into account is significatively more difficult and will be the object of future work. Each formin can thus bind to or unbind from a polymer at a given rate. We shall say that a polymer is in "fast" mode if it is associated with a formin and therefore polymerises faster. When it is not bound with a formin, we say that it is in "normal" mode. To account for the role of profilin balancing the two network populations, we formally assembled filaments in "normal" mode from a G-actin complexes, while filaments in "fast" mode were assembled from G-actin/profilin (Figure 1.1 and [SK16]). Finally, note that our simple model, we also ignore the effect of the formins on actin nucleation.

Our main interest is twofold: understand the consequences of the fact that monomers are in limited numbers (being free, in a complex with a profilin or inserted in a polymer), and characterise the impact of the switching between two modes on the distribution of polymer lengths and on the fraction of monomers incorporated in a polymer. Several models for actin assembly have already been developed [BODM04, EKE98, EEK98, YB09, YDB08]. The majority of these models are either deterministic and focus on a steady-state equilibrium, or are stochastic and are studied by Monte-Carlo simulations [Bin10, PB09]. Most of the models taking into account a population of actin polymers rely on very specific assumptions and are not developed in a general context. Indeed, many of them concentrate on cell motility and therefore focus on the dynamics of actin in protrusions [BODM04, MEK02a] or in the geometrical organisation of the actin network [CEK94]. More general models like [EKE98, EEK98] do not take into account the faster elongation of polymers bound with a formin, and neither do they consider the effect of the switching between spontaneous elongation and elongation with a formin. The works [HO11] and [MLH+17] use the same approach as ours to study the average length of an actin polymer and the distribution of the polymer lengths. However, [HO11] does not consider the large population limit of their model and [MLH+17] focuses more on the importance of having two types of monomers (ATP and ADP) than on the effects of accessory proteins. Apart from the population approaches listed above, most of the existing literature tends to focus on a single polymer, or on two competing polymers, with the idea that very large populations of polymers will then be well-described by the law of an "average" polymer. Importantly, we wanted to explore the effect of the competition between different sub-populations of polymers, elongated by distinct factors, for a unique, limited monomer pool.

To this end, we first consider a single polymer and its interactions with formins and profilins in Section 2.2. Taking a particular limit as the total number of monomers in the system tends to infinity, we show that if we allow this polymer to "live" for a very long time (by taking its initial length to be of order $\mathcal{O}(N)$), then its interactions with formins average out and the resulting asymptotic polymerisation rate is a weighted average between the polymerisation rate in fast and in normal mode (i.e. when bound or not with a formin). In Section 2.3, we instead consider a large population of short competing polymers and show that, because short polymers never reach such a long lifetime, the intuitive homogenisation result of Section 2.2 does not apply and the competition for free monomers or G-actin/profilin complexes has a strong impact on the size distribution within the population of polymers. In Section 5, we show by simulation that the characteristics of a single polymer considered on its own can differ

significantly from the distribution in polymer lengths in a large population even for equivalent model parameters.

2.2 - Switching Dynamics of a Single Actin Polymer

2.2.1 - The model

We first focus on the effect of the switching between different modes on the evolution of a single polymer. We restrict our attention to two modes (fast and normal), but our result would easily carry over to the case of finitely many modes under some natural ergodicity conditions.

The model we propose below takes into account the spontaneous elongation of the polymer as well as the effects of formins and profilins. If there are no formins attached to the plus end of the polymer, elongation is spontaneous and consumes free monomers. In contrast, if there is a formin on the polymer, elongation is induced by the formin and G-actin/profilin complexes are integrated into the polymer instead of free monomers. Only one formin can be bound to the polymer at a time. Recall that in the first case we say that the polymer is in "normal" mode, while the second case is called the "fast" mode. For every $t \geq 0$, we write $M(t) = 0$ (resp. $M(t) = 1$) if the polymer is in normal (resp. fast) mode at time t.

As we want monomers to be a limiting factor, we assume that the total number of monomers in the system is fixed through time to some $N \in \mathbb{N}$. These monomers can be in three states: free, in the polymer or held in a G-actin/profilin complex (each involving only one monomer). At any given time $t \geq 0$, the current number of free monomers is denoted by $L_0^N(t)$, the length of the polymer by $L_1^N(t)$ and the number of G-actin/profilin complexes by $L_2^N(t)$. We also define:

$$L^N(t) \overset{def}{=} \left(L_0^N(t), L_1^N(t), L_2^N(t)\right). \tag{1.1}$$

Writing $[N]$ for the set $\{0, 1, 2, \dots, N\}$, we thus have for any $i \in \{0, 1, 2\}$ and any $t \in \mathbb{R}_+$: $L_i^N(t) \in [N]$ and

$$L_0^N(t) + L_1^N(t) + L_2^N(t) = N. \tag{1.2}$$

The model incorporates six types of events, illustrated in Figure 1.1:

1. **Spontaneous elongation.** If the polymer has positive length and is in normal mode, elongation (at the plus end) is spontaneous, and we assume that the rate at which it occurs is proportional to the current density of free monomers. That is, the instantaneous rate of spontaneous elongation a time t is given by $\mathbb{1}_{\{M(t-)=0\}}\mathbb{1}_{\{L_1^N(t-)>0\}}\lambda^+ L_0^N(t-)/N$, for some constant $\lambda^+ \geq 0$ and where we denote by $\mathbb{1}_S$ the indicator function of a set S. The new value of the vector describing the system is then:
$$L^N(t) = \left(L_0^N(t-) - 1, L_1^N(t-) + 1, L_2^N(t-)\right).$$

2. **Elongation with a formin.** If the system is in fast mode and the polymer has positive length, elongation (also at the plus end) is induced by a formin and consumes G-actin/profilin complexes. Therefore, in this case we suppose that the instantaneous rate of polymerisation is proportional to the density of available complexes, i.e. is equal to $\mathbb{1}_{\{M(t-)=1\}}\mathbb{1}_{\{L_1^N(t-)>0\}}\lambda_F^+ L_2^N(t-)/N$ for some constant $\lambda_F^+ \geq 0$. The vector L^N then jumps to:
$$L^N(t) = \left(L_0^N(t-), L_1^N(t-) + 1, L_2^N(t-) - 1\right).$$

3. **Depolymerisation.** In both modes, the polymer can release a monomer (from the minus end) if its length has not already reached 0. We write $\mathbb{1}_{\{L_1^N(t-)>0\}}\lambda^-$ for rate of depolymerisation, where λ^- is a non-negative constant, and the new value of L^N is then:

$$L^N(t) = \left(L_0^N(t-) + 1, L_1^N(t-) - 1, L_2^N(t-)\right).$$

4. **Production of complexes.** Independently of the mode of the polymer, a free monomer and a profilin can react and produce a G-actin/profilin complex. Assuming that profilin is not a limiting factor, the instantaneous rate of production of G-actin/profilin complexes at time t is given by $\Phi_P L_0^N(t-)/N$, for some constant Φ_P that takes into account the concentration in profilin in the cellular cortex. The new value of L^N after such an event is:

$$L^N(t) = \left(L_0^N(t-) - 1, L_1^N(t-), L_2^N(t-) + 1\right).$$

5. **Binding of a formin.** If the polymer is in normal mode, a formin can bind to its plus end and the polymer switches to the fast mode. We write Φ_F^+ for the constant rate of occurrence such a switch from normal to fast mode. This event does not modify the value of L^N, but corresponds to a jump from 0 to 1 for $M(t)$.

6. **Release of a formin.** If the polymer is in fast mode, the formin bound to it can be released and the polymer switches to the normal mode. We denote Φ_F^- the constant rate of such a switch. Again, this event only affects the process M, which jumps from 1 to 0.

Observe that monomers cannot spontaneously detach from a G-actin/profilin complex. However, when a complex is integrated into a polymer, its depolymerisation results in the separation of the complex into a free monomer and a (free) profilin. Once the length of the polymer has reached 0, for mathematical convenience we suppose that $(M(t))_{t\in\mathbb{R}_+}$ keeps on jumping from 0 to 1 at rate Φ_F^+ and from 1 to 0 at rate Φ_F^- for all times, even if this assumption has no biological meaning.

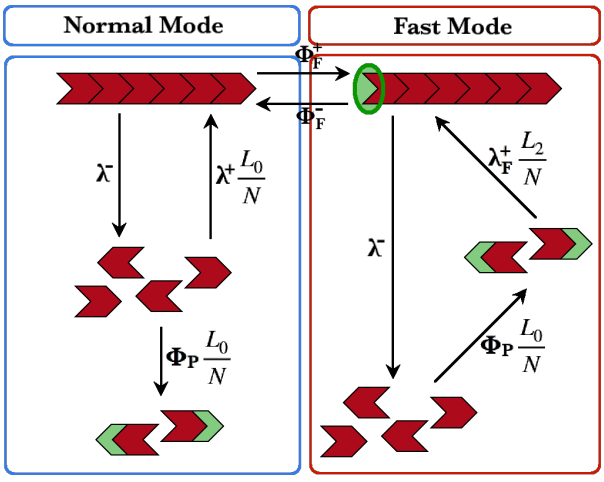


Figure 1.1 Schematic representation of the model for the dynamics of a single actin polymer and its interaction with formins and profilins. Red single units represent monomers, red-green units represent G-actin/profilin complexes and the strand of red units at the top of each box represents the polymer. The green oval depicts a formin bound to the polymer.

The system is fully described by the Markov process $(L^N(t), M(t))_{t\in\mathbb{R}_+}$. From the above, we see that $(M(t))_{t\in\mathbb{R}_+}$ is a Markov jump process with values in $\{0, 1\}$, switching from 0 to 1 at rate Φ_F^+ and from

1 to 0 at rate Φ_F^-. We assume that $\Phi_F^+ + \Phi_F^- > 0$ to ensure the presence of mode switching. It evolves independently of the process $(L^N(t))_{t\in\mathbb{R}_+}$, while the dynamics of $(L^N(t))_{t\in\mathbb{R}_+}$ depends on $(M(t))_{t\in\mathbb{R}_+}$. Since $(M(t))_{t\in\mathbb{R}_+}$ is an irreducible Markov process on a finite state space, it has a unique stationary distribution π_M given by:

$$(\pi_M(0)\,,\ \pi_M(1)) = \left(\frac{\Phi_F^-}{\Phi_F^+ + \Phi_F^-}\,,\ \frac{\Phi_F^+}{\Phi_F^+ + \Phi_F^-}\right). \tag{1.3}$$

This stationary distribution will appear in the homogenisation result presented in the next section.

2.2.2 - Large-N limit

In order to understand the first order fluctuations of the system, we let the total number of monomers tend to infinity and look for a *fluid limit* as N tends to infinity. That is, we consider the scaled (non-Markovian) process,

$$\overline{L^N}(t) = \left(\overline{L_0^N}(t), \overline{L_1^N}(t), \overline{L_2^N}(t)\right) \overset{def}{=} \left(\frac{L_0^N(Nt)}{N}, \frac{L_1^N(Nt)}{N}, \frac{L_2^N(Nt)}{N}\right), \tag{1.4}$$

and we want to prove that $\overline{\boldsymbol{L^N}} = \left(\overline{L^N}(t)\right)_{t\in\mathbb{R}_+}$ converges to a deterministic limit as $N \longrightarrow +\infty$.

Let us write:

$$\Delta \overset{def}{=} \left\{(x_1, x_2, x_3) \in [0,1]^3 \mid x_1 + x_2 + x_3 = 1\right\} \tag{1.5}$$

for the compact set in which $\overline{\boldsymbol{L^N}}$ takes its values for every N and $\mathcal{D}_\Delta[0, +\infty[$ for the space of all càdlàg Δ-valued paths, which we endow with the standard Skorokhod topology. Let

$$C_0 \overset{def}{=} \pi_M(0)\lambda^+ + \Phi_P \qquad \text{and} \qquad C_2 \overset{def}{=} \pi_M(1)\lambda_F^+, \tag{1.6}$$

where $\pi_M(0)$ and $\pi_M(1)$ are defined in (1.3). The constants C_0 and C_2 are respectively the rate of consumption of free monomers and of G-actin/profilin complexes when the process M is at stationarity. To avoid trivialities, we suppose that these two quantities are positive.

We can now formulate the main result of this section.

Theorem 1.2.1 (Fluid Limit)
Suppose that the initial condition $\overline{L^N}(0) \in \Delta$ converges in distribution to some deterministic vector $(l_0(0), l_1(0), l_2(0)) \in \Delta$. Then as $N \longrightarrow +\infty$, the process $\overline{\boldsymbol{L^N}}$ converges in distribution in $D_\Delta[0, +\infty[$ to the deterministic process $(l(t))_{t\in\mathbb{R}_+} = (l_0(t), l_1(t), l_2(t))_{t\in\mathbb{R}_+}$ satisfying: for every $t \le t_0$,

$$\begin{cases} l_0(t) &= \left(l_0(0) - \frac{\lambda^-}{C_0}\right) e^{-C_0 t} + \frac{\lambda^-}{C_0}, \\ l_1(t) &= 1 - l_0(t) - l_2(t), \\ l_2(t) &= l_2(0)e^{-C_2 t} + \frac{\Phi_P}{C_2 - C_0}\left(l_0(0) - \frac{\lambda^-}{C_0}\right)\left(e^{-C_0 t} - e^{-C_2 t}\right) + \frac{\Phi_P \lambda^-}{C_0 C_2}\left(1 - e^{-C_2 t}\right), \end{cases}$$

and for every $t > t_0$,

$$\begin{cases} l_0(t) &= l_0(t_0)e^{-\Phi_P(t-t_0)}, \\ l_1(t) &= 0, \\ l_2(t) &= 1 - l_0,(t) \end{cases}$$

where $t_0 = \min\{t \in \mathbb{R}_+ \ :\ l_1(t) = 0\}$ is the time at which the polymer "size" cancels (by convention, $t_0 = +\infty$ if l_1 does not reach 0).

We easily see from Theorem 1.2.1 that the system obtained in the limit has a unique equilibrium, which depends on whether $t_0 < +\infty$ or not. If $t_0 < +\infty$ then the system converges to $(0, 0, 1)$ and every monomer ends up associated with a profilin. In contrast, if $t_0 = +\infty$, the limiting distribution of monomers into the three different categories is given by:

$$\left(\frac{\lambda^-}{C_0}, 1 - \frac{\lambda^-}{C_0} - \frac{\lambda^-\Phi_P}{C_0 C_2}, \frac{\lambda^-\Phi_P}{C_0 C_2}\right). \tag{1.7}$$

Observe that the limiting amount of free monomers corresponds to the ratio of the asymptotic rate of increase of the free monomer pool, λ^-, by the asymptotic rate $C_0 = \pi_M(0)\lambda^+ + \Phi_P$ at which free monomers are consumed by the polymer elongation in normal mode and the creation of G-actin/profilin complexes. Likewise, since complexes can only be created by the reaction of a free monomer with a profilin, the limiting amount of complexes can be read as the ratio of their creation rate from the pool of free monomers at equilibrium, $(\lambda^-/C_0) \times \Phi_P$, by the rate of consumption of complexes through polymerisation in fast mode, C_2.

A natural question, on which we shall not dwell, is of course to ask when t_0 is finite. An obvious sufficient (but not necessary) condition for $t_0 < \infty$ is:

$$1 - \frac{\lambda^-}{C_0} - \frac{\lambda^-\Phi_P}{C_0 C_2} < 0. \tag{1.8}$$

Theorem 1.2.1 pertains to the family of stochastic homogenisation results such as [PP15]. Indeed, although most reactions occur on a similar time-scale, the time it takes to the polymer to see its length change by $\mathcal{O}(N)$ units (so that this change may be visible after our rescaling of L_1^N by N) is of the order of $\mathcal{O}(N)$ original time units, and is therefore much larger than the time of order $\mathcal{O}(1)$ that the binding and release of formins take. What our result shows is that, in the limit as N tends to infinity, the fast dynamics of mode switching equilibrates and only its stationary distribution π_M impacts the slow dynamics of the polymer elongation and shortening. Recalling that $\pi_M(0)$ (resp. $\pi_M(1)$) can be interpreted as the fraction of time spent in the normal mode (resp. fast mode) over the very long time-scale considered here (by ergodicity), we see that, as mentioned earlier, C_0 (resp. C_2) corresponds to a global rate of consumption of free monomers (resp. of G-actin/profilin complexes) when M is at stationarity, and the average behaviour described by the equations stated in Theorem 1.2.1 corresponds to the barycenter between the behaviour of the system when it is in normal mode and the behaviour of the system when it is in fast mode.

Notice that our approach is similar to that of [PP15, RS17], in which each individual switches state (or spatial location) independently, but it differs from many stochastic homogenisation results such as [BV17] or [BL16], in which it is the environment where all individuals evolve which switches fast, affecting the population as a whole. More precisely, in our study fluctuations are not on the scale of the population but on the scale of the individuals since each polymer can be in several modes independently of the others. This kind of random individual state switching seems less studied than the long-term averaging effect of environmental switches.

The limiting process $(l(t))_{t\in\mathbb{R}_+}$ allows us to understand the impact of the restriction on monomers and of the interaction with formins and profilins on the dynamics of a single polymer. However, our model does not take into account the competition between the numerous polymers which are present in the cellular cortex. As a consequence, we now build on our model for a single polymer to construct and analyse a more complex model for a *population* of actin polymers.

2.3 - Switching Dynamics for a Population of Actin Polymers

2.3.1 - The model

We now consider a population of actin polymers undergoing the same reactions as in the previous section. Recall that we assume that the total number of monomers in the system is fixed, equal to $N \in \mathbb{N}$. Again, these monomers can be free, in a G-actin/profilin complex, in a polymer bound with a formin or in a simple polymer (with no formin attached). As earlier, if an actin polymer is simple its elongation consumes free actin monomers, and if the polymer is associated with a formin, its elongation consumes only G-actin/profilin complexes. To ease the exposition, we shall consider that free monomers and G-actin/profilin complexes are two types of actin polymers.

Since we want to understand the distribution of polymer lengths within the population as well as the amounts of monomers contained in each category of polymers, we describe every element of the population by its type and its length. More precisely, free monomers are denoted by M, G-actin/profilin complexes by P, simple actin polymers by S and actin polymers bound with a formin by F. Polymers of type M and P can only be of size 1, while polymers of type S and F can be of any length in $\{1, 2, \dots, N\}$. By convention, we distinguish between polymers of length one and free monomers as polymers made of a single monomer can grow again. Other conventions are possible and similar results to the ones presented below can be obtained using the same arguments as in the proof of Theorem 1.2.2.

Later it will be more convenient to have a state space independent of N, we thus let

$$\mathcal{I} \overset{def}{=} \{(M,1),(P,1)\} \cup (\{S,F\} \times \mathbb{N}), \tag{1.9}$$

and for every pair $(T,l) \in \mathcal{I}$ and every $t \geq 0$, we define $K_t^N(T,l)$ as the number of polymers of type T and length l present in the system at time t. Here we use the convention $\mathbb{N} = \{1,2,\dots\}$. The system at every time t is described by its empirical distribution $\mathcal{Z}_t^N$:

$$\mathcal{Z}_t^N \overset{def}{=} K_t^N(M,1)\delta_{(M,1)} + K_t^N(P,1)\delta_{(P,1)} + \sum_{l=1}^{N} K_t^N(S,l)\delta_{(S,l)} + \sum_{l=1}^{N} K_t^N(F,l)\delta_{(F,l)}, \tag{1.10}$$

where δ_x denotes a Dirac mass at x. Each $\mathcal{Z}_t^N$ is thus a random measure on $\mathcal{I}$. We write $\mathcal{M}_{\mathcal{I}}$ for the space of all finite measures on $\mathcal{I}$, endowed with the topology of weak convergence.

Let us make a few remarks. First, let p_l denote the projector $(T,l) \mapsto l$ that returns the second coordinate of an element in $\mathcal{I}$. For every $Z \in \mathcal{M}_{\mathcal{I}}$ and every measurable function $f : \mathcal{I} \to \mathbb{R}$, we use the standard notation:

$$\langle Z, f \rangle = \int_{\mathcal{I}} f(i) Z(di),$$

whenever this integral makes sense. Since the total number of monomers is N, for every $t \geq 0$ we have $K_t^N(\cdot, l) = 0$ if $l > N$ and

$$\left\langle \mathcal{Z}_t^N, p_l \right\rangle = K_t^N(M,1) + K_t^N(P,1) + \sum_{l=1}^{N} l K_t^N(S,l) + \sum_{l=1}^{N} l K_t^N(F,l) = N. \tag{1.11}$$

The total number of "real" polymers (i.e. that are not free monomers or complexes) is denoted by $F^N(t)$:

$$F^N(t) \overset{def}{=} \sum_{l=1}^{N} \left(K_t^N(S,l) + K_t^N(F,l) \right) = \left\langle \mathcal{Z}_t^N, \mathbb{1}_{(S,\cdot)} + \mathbb{1}_{(F,\cdot)} \right\rangle. \tag{1.12}$$

Observe that the quantity $K^N(M,1)$ corresponds to what we called L_0^N in the model with one polymer and $K^N(P,1)$ corresponds to what we called L_2^N. The other quantities are not quite comparable to those introduced in the previous model, but an analogue of L_1 would be the total number of monomers inserted in a polymer:

$$M_{inF}^N(t) \overset{def}{=} \sum_{l=1}^{N} l\left(K_t^N(S,l) + K_t^N(F,l)\right) = \left\langle \mathcal{Z}_t^N, p_l\left(\mathbb{1}_{(S,\cdot)} + \mathbb{1}_{(F,\cdot)}\right)\right\rangle. \tag{1.13}$$

The process $(\mathcal{Z}_t^N)_{t\in\mathbb{R}_+}$ evolves according to the same six types of transitions as in the single polymer model.

1. **Spontaneous elongation.** Each simple polymer can elongate by catching a free monomer, independently of the other polymers. As in the first model, $\lambda^+ K_{t-}^N(M,1)/N$ is the constant rate at which a given polymer captures a free monomer. Consequently, the total rate at which a spontaneous elongation happens at time t is $\lambda^+(K_{t-}^N(M,1)/N)\sum_{l\geq 1} K_{t-}^N(S,l)$ and conditionally on this polymer being of size ℓ just before the event (which happens with probability $K_{t-}^N(S,\ell)/\sum_l K_{t-}^N(S,l)$), this corresponds to a transition to:

$$\mathcal{Z}_t^N = \mathcal{Z}_{t-}^N - \delta_{(S,\ell)} - \delta_{(M,1)} + \delta_{(S,\ell+1)},$$

or equivalently to:

$$\begin{cases} K_t^N(S,\ell) &= K_{t-}^N(S,\ell) - 1, \\ K_t^N(S,\ell+1) &= K_{t-}^N(S,\ell+1) + 1, \\ K_t^N(M,1) &= K_{t-}^N(M,1) - 1. \end{cases}$$

2. **Elongation with a formin.** Each polymer associated with a formin can elongate by absorbing a G-actin/profilin complex (again, independently of the other polymers). As in the previous type of transition, the total rate at which an elongation with a formin happens at time t is $\lambda_F^+(K_{t-}^N(P,1)/N)\sum_l K_{t-}^N(F,l)$ and conditionally on this polymer being of size ℓ just before the event (which happens with probability $K_{t-}^N(F,\ell)/\sum_l K_{t-}^N(F,l)$), this corresponds to a transition to:

$$\mathcal{Z}_t^N = \mathcal{Z}_{t-}^N - \delta_{(F,\ell)} - \delta_{(P,1)} + \delta_{(F,\ell+1)},$$

or equivalently to:

$$\begin{cases} K_t^N(F,\ell) &= K_{t-}^N(F,\ell) - 1, \\ K_t^N(F,\ell+1) &= K_{t-}^N(F,\ell+1) + 1, \\ K_t^N(P,1) &= K_{t-}^N(P,1) - 1. \end{cases}$$

3. **Depolymerisation.** Each polymer of length $l \geq 1$ (bound or not with a formin) can release a monomer at rate λ^- and this monomer becomes "free". The (total) instantaneous rate at which depolymerisation occurs to a polymer of length ℓ at time t is thus $\lambda^-(K_{t-}^N(S,\ell) + K_{t-}^N(F,\ell))$, and if $T \in \{S,F\}$ is the type of this polymer, the new value of the empirical measure at time t is:

$$\mathcal{Z}_t^N = \mathcal{Z}_{t-}^N - \delta_{(T,\ell)} + \delta_{(M,1)} + \delta_{(T,\ell-1)}\mathbb{1}_{\{\ell\geq 2\}}.$$

or equivalently:

$$\begin{cases} K_t^N(T,\ell) &= K_{t-}^N(T,\ell) - 1, \\ K_t^N(T,\ell-1) &= K_{t-}^N(T,\ell-1) + 1, \\ K_t^N(M,1) &= K_{t-}^N(M,1) + 1. \end{cases}$$

4. **Production of complexes.** Each free monomer can bind with a profilin and create a G-actin/profilin complex at rate Φ_P, independently of the others. Therefore, the total rate of creation of complexes at time t is $\Phi_P K^N_{t-}(M,1)$ and the new state of the system after such an event is:

$$\mathcal{Z}^N_t = \mathcal{Z}^N_{t-} - \delta_{(M,1)} + \delta_{(P,1)}.$$

or equivalently:

$$\begin{cases} K^N_t(P,1) &= K^N_{t-}(P,1) + 1, \\ K^N_t(M,1) &= K^N_{t-}(M,1) - 1. \end{cases}$$

5. **Binding of a formin.** Each simple polymer of length l can bind with a formin and therefore change category (or "mode"), independently of the other polymers. If we write again Φ^+_F for the rate at which a given simple polymer becomes bound with a formin, the total rate at which such an event occurs is $\Phi^+_F K^N_{t-}(S,l)$ and conditionally on the length of the polymer concerned being ℓ, the new state of the system is:

$$\mathcal{Z}^N_t = \mathcal{Z}^N_{t-} - \delta_{(S,\ell)} + \delta_{(F,\ell)}.$$

or equivalently:

$$\begin{cases} K^N_t(S,\ell) &= K^N_{t-}(S,\ell) - 1, \\ K^N_t(F,\ell) &= K^N_{t-}(F,\ell) + 1. \end{cases}$$

6. **Release of a formin.** Each polymer associated with a formin can have this association broken at rate Φ^-_F, independently of the other polymers. Hence, the total rate of release of a formin by a polymer of length ℓ at time t is $\Phi^-_F K^N_{t-}(F,\ell)$, and the new state of the system after such an event is:

$$\mathcal{Z}^N_t = \mathcal{Z}^N_{t-} - \delta_{(F,\ell)} + \delta_{(S,\ell)}.$$

or equivalently:

$$\begin{cases} K^N_t(F,\ell) &= K^N_{t-}(F,\ell) - 1, \\ K^N_t(S,\ell) &= K^N_{t-}(S,\ell) + 1. \end{cases}$$

Figure 1.2 summarises all the events described above.

2.3.2 - Large-N limit

We now look for a large population limit, describing the first order behaviour of our population of polymers. More precisely, for every $N \in \mathbb{N}$ and $t \in \mathbb{R}_+$, we set:

$$\overline{\mathcal{Z}^N_t} \overset{def}{=} \frac{1}{N}\mathcal{Z}^N_t. \tag{1.14}$$

By construction, each $\overline{\mathcal{Z}^N_t}$ takes its values in the set $\mathcal{M}^*_{\mathcal{I}}$ defined by:

$$\mathcal{M}^*_{\mathcal{I}} = \{Z \in \mathcal{M}_{\mathcal{I}} \mid \langle Z, p_l\rangle = 1\}, \tag{1.15}$$

where we recall that $\mathcal{M}_{\mathcal{I}}$ is the set of all finite measures on $\mathcal{I}$ (see (1.9)), and $\langle Z, p_l\rangle$ is defined as in (1.11). In particular, since the length l of every monomer/polymer is at least one, we have

$$\mathcal{M}^*_{\mathcal{I}} \subset \mathcal{M}^{(1)}_{\mathcal{I}} \overset{def}{=} \{Z \in \mathcal{M}_{\mathcal{I}} \mid \langle Z, 1\rangle \leq 1\}. \tag{1.16}$$

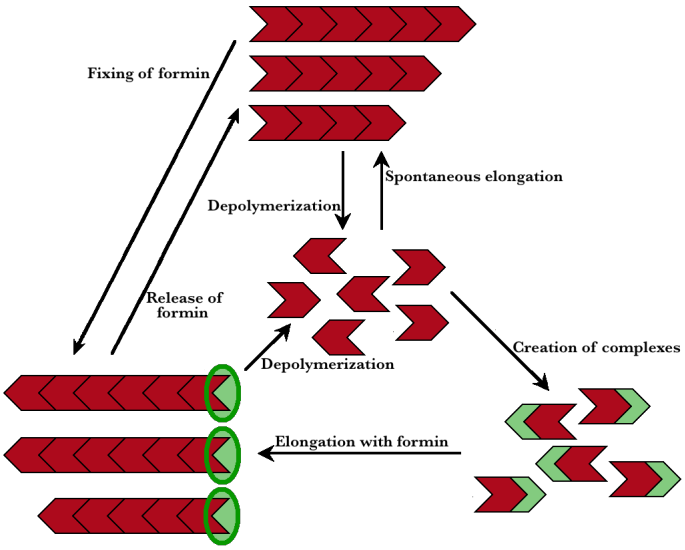


Figure 1.2 Transitions in the model for a population of polymers. Simple polymers can elongate by catching a free monomer, release a monomer that becomes free, or switch to the "fast" mode by binding with a formin. Polymers bound with a formin can elongate by catching a G-actin/profilin complex, release a monomer that becomes free, or switch to the "normal" mode by releasing the attached formin. Finally, free monomers and profilins can bind to form G-actin/profilin complexes.

We now let N tend to infinity, assuming that the sequence of initial values $\overline{\mathcal{Z}_0^N}$ converges to some measure in $\mathcal{M}_I^{(1)}$ and that the support of $\overline{\mathcal{Z}_0^N}$ is bounded uniformly in N. Note that we do not scale time by a factor N as we did to obtain the fluid limit for a single polymer in Section 2.2. Our main result is the following. We denote the space of all càdlàg trajectories with values in $\mathcal{M}_I^{(1)}$ by $D_{\mathcal{M}_I^{(1)}}[0,+\infty[$ and we endow it with the standard Skorokhod topology.

Theorem 1.2.2 (Large Population Limit)
Suppose that the initial condition $\overline{\mathcal{Z}_0^N}$ converges in distribution to a deterministic limit $\mathcal{Z}_0 \in \mathcal{M}_I^{(1)}$ as $N \longrightarrow +\infty$, and that there exists $K_0 \in \mathbb{N}$ such that for every N sufficiently large,

$$\left\langle \overline{\mathcal{Z}_0^N}, \mathbb{1}_{\{l \geq K_0\}} \right\rangle = 0. \tag{1.17}$$

Then the process $\overline{\mathcal{Z}^N}$ converges in distribution in $D_{\mathcal{M}_I^{(1)}}[0,+\infty[$ as $N \longrightarrow +\infty$, to a deterministic limit $\mathcal{Z} = (\mathcal{Z}_t)_{t\in\mathbb{R}_+}$ which can be characterised as follows.

If for every $(T,l) \in \mathcal{I}$ we write $k_t(T,l)$ for the mass $\mathcal{Z}_t(\{(T,l)\})$ given by $\mathcal{Z}_t$ to the point (T,l), then:

$$k_0(T,l) = \mathcal{Z}_0(\{(T,l)\}),$$

and

$$
\begin{cases}
\frac{d}{dt}k_t(M,1) &= \lambda^- \sum_{l=1}^{+\infty}(k_t(S,l)+k_t(F,l)) - \lambda^+ k_t(M,1)\sum_{l=1}^{+\infty} k_t(S,l) - \Phi_P k_t(M,1)\\
\frac{d}{dt}k_t(P,1) &= \Phi_P k_t(M,1) - \lambda_F^+ k_t(P,1)\sum_{l=1}^{+\infty} k_t(F,l),\\
\frac{d}{dt}k_t(S,1) &= \Phi_F^- k_t(F,1) + \lambda^- k_t(S,2) - (\lambda^+ k_t(M,1) + \lambda^- + \Phi_F^+)k_t(S,1)\\
\frac{d}{dt}k_t(F,1) &= \Phi_F^+ k_t(S,1) + \lambda^- k_t(F,2) - (\lambda_F^+ k_t(P,1) + \lambda^- + \Phi_F^-)k_t(F,1)\\
\text{and } \forall l \geq 2,&\\
\frac{d}{dt}k_t(S,l) &= \lambda^+ k_t(M,1)k_t(S,l-1) + \Phi_F^- k_t(F,l) + \lambda^- k_t(S,l+1) - (\lambda^+ k_t(M,1) + \lambda^- + \Phi_F^+)k_t(S,l)\\
\frac{d}{dt}k_t(F,l) &= \lambda_F^+ k_t(P,1)k_t(F,l-1) + \Phi_F^+ k_t(S,l) + \lambda^- k_t(F,l+1) - (\lambda_F^+ k_t(P,1) + \lambda^- + \Phi_F^-)k_t(F,l).
\end{cases}
$$

Remark 1.2.3

i - Ideally, we would like to replace the assumption that $\mathcal{Z}_0 \in \mathcal{M}_{\mathcal{I}}^{(1)}$ by $\mathcal{Z}_0 \in \mathcal{M}_{\mathcal{I}}^*$ and obtain the convergence of the sequence $\left(\overline{\mathcal{Z}^N}\right)_{N\geq 1}$ in $D_{\mathcal{M}_{\mathcal{I}}^*}[0,+\infty[$. However, the fact that $\mathcal{Z}_t \in \mathcal{M}_{\mathcal{I}}^*$ for all $t \geq 0$ is not guaranteed by the convergence in distribution of $\overline{\mathcal{Z}_t^N}$ seen as a finite measure on $\mathcal{I}$, since the function $p_l : (T,l) \mapsto l$ is not bounded continuous on $\mathcal{I}$ and so the property that $\langle \mathcal{Z}_t, p_l\rangle = 1$ needs not hold in the limit. In Remark 1.4.2, we argue that in the limiting system, at any time $t \geq 0$ we have $\langle \mathcal{Z}_t, p_l\rangle < \infty$. Unfortunately, we were not able to prove that this integral is actually equal to 1. For this, one would have to be able either to compute the dynamics of $(\langle \mathcal{Z}_t, p_l\rangle)_{t\in\mathbb{R}_+}$ from the system of equations stated in Theorem 1.2.2, or to finely control the total length of polymers in the stochastic pre-limiting system, and in particular to control the amount of polymers of length larger than a given $m \in \mathbb{N}$ uniformly in N (for each given time $t \geq 0$). Indeed, although the probability that very long polymers form in a finite time t is very small (see the paragraph surrounding Equation (1.47)), a vanishing proportion of the polymer lengths may drift to infinity fast enough for their contribution to $\left\langle \overline{\mathcal{Z}_t^N}, p_l \right\rangle$ (which is equal to 1 by construction, for finite N) to remain lower bounded by some $\epsilon > 0$ as $N \longrightarrow +\infty$. Since in the limiting measures $\mathcal{Z}_t$ all polymers of "infinite length" are lost (recall that $\mathcal{Z}_t$ is a measure on $\mathcal{I} \subset \{M,P,S,F\}\times\mathbb{N}$), this would lead to $\langle \mathcal{Z}_t, p_l\rangle \leq 1-\epsilon$. Numerical experiments tend to show that indeed $(\mathcal{Z}_t)_{t\in\mathbb{R}_+}$ lives in $\mathcal{M}_{\mathcal{I}}^*$ for a large range of parameter values, with the caveat that very long polymers must also be very rare (by (1.47)) and therefore they are very difficult to capture by numerical simulations.

ii - Notice that the assumption that the measures $\overline{\mathcal{Z}_0^N}$ should have uniformly bounded supports imposes that $\overline{\mathcal{Z}_0^N}$ should asymptotically give no weight to polymers whose sizes tend to infinity with N, so that no mass is lost on infinitely long polymers in the limit. If for instance the initial population was made of a few polymers of size $\mathcal{O}(N)$ (and free monomers, so that (1.11) would be satisfied), we would rather have to scale time and polymer lengths by N as in Section 2.2 to obtain a non-trivial limit and this limit would be the obvious analogue for several polymers of the homogenisation result stated in Theorem 1.2.1 for a single polymer. Since our main motivation for this work was to understand the dynamics of the distribution in lengths and types of the large population of polymers of length $\mathcal{O}(1)$ present in the cellular cortex under standard conditions, we chose to focus on initial populations with many $\mathcal{O}(1)$-long polymers. But in this regime, a single "short" polymer is liable to disappear before having reached the timescale on which the homogenisation effect of the switching dynamics appears, and to be able to compare the long-term effect of mode switching on a single polymer and on a population of polymers competing for resources, in Section 2.2 we had to impose that the polymer of interest should initially be long enough for the switching to have enough time to act before the polymer length reaches 0. This explains the discrepancy between the set-ups of Sections 2.2 and 2.3. Note however that experimental approaches to measure the length of actin filaments, based on elec-

tron micrographs (e.g. [SVMB97]) or more recently on cry-electron tomography (e.g. [MSC+20]), typically yield estimates that vary from a few hundreds nanometers [BSK+02, CLZ91] to a few micrometers [SHA95, XCP99], covering at least one order of magnitude.

The rest of the paper is organised as follows. In Section 3, we prove Theorem 1.2.1 on the asymptotic behaviour of a single (long) polymer. In Section 4, we prove Theorem 1.2.2 on the asymptotic behaviour of a population of actin polymers. Corollary 1.4.3 allows to compare these two results and highlights the effects of the competition between polymers for G-actin on the evolution of the population. Finally, some simulations are discussed in Section 5.

3 - Homogenisation Result for a Single Actin Polymer

In this section, we prove Theorem 1.2.1. To this end, in Section 3.1 we prove that the sequence of processes $\left(\overline{L^N}\right)_{N\geq 1}$ is C-tight, and in Section 3.2 we show that any limit point satisfies the deterministic system of equations stated in Theorem 1.2.1. Before proceeding to these proofs, we establish some useful preliminary properties.

Let us consider the evolution of L^N for some fixed N. Each of its coordinates takes its values in the finite set $[N]$, and so $\left(L^N(t), M(t)\right)_{t\in\mathbb{R}_+}$ is a Markov jump process with values in a finite state space. This allows us to write that for every $t \geq 0$,

$$L_0^N(t) = L_0^N(0) + \int_0^t \mathbb{1}_{\{L_1^N(s)>0\}}\lambda^- ds - \int_0^t \mathbb{1}_{\{M(s)=0\}}\mathbb{1}_{\{L_1^N(s)>0\}}\lambda^+ \frac{L_0^N(s)}{N} ds - \int_0^t \Phi_P \frac{L_0^N(s)}{N} ds + X_0^N(t), \tag{1.18a}$$

$$L_1^N(t) = L_1^N(0) + \int_0^t \mathbb{1}_{\{M(s)=0\}}\mathbb{1}_{\{L_1^N(s)>0\}}\lambda^+ \frac{L_0^N(s)}{N} ds - \int_0^t \mathbb{1}_{\{L_1^N(s)>0\}}\lambda^- ds + \int_0^t \mathbb{1}_{\{M(s)=1\}}\mathbb{1}_{\{L_1^N(s)>0\}}\lambda_F^+ \frac{L_2^N(s)}{N} ds + X_1^N(t), \tag{1.18b}$$

$$L_2^N(t) = L_2^N(0) + \int_0^t \Phi_P \frac{L_0^N(s)}{N} ds - \int_0^t \mathbb{1}_{\{M(s)=1\}}\mathbb{1}_{\{L_1^N(s)>0\}}\lambda_F^+ \frac{L_2^N(s)}{N} ds + X_2^N(t), \tag{1.18c}$$

where X_0^N, X_1^N and X_2^N are mean-zero martingales with respective predictable quadratic variations

$$\left\langle X_0^N \right\rangle(t) = \int_0^t \mathbb{1}_{\{L_1^N(s)>0\}}\lambda^- ds + \int_0^t \mathbb{1}_{\{M(s)=0\}}\mathbb{1}_{\{L_1^N(s)>0\}}\lambda^+ \frac{L_0^N(s)}{N} ds + \int_0^t \Phi_P \frac{L_0^N(s)}{N} ds, \tag{1.19a}$$

$$\left\langle X_1^N \right\rangle(t) = \int_0^t \mathbb{1}_{\{M(s)=0\}}\mathbb{1}_{\{L_1^N(s)>0\}}\lambda^+ \frac{L_0^N(s)}{N} ds + \int_0^t \mathbb{1}_{\{L_1^N(s)>0\}}\lambda^- ds + \int_0^t \mathbb{1}_{\{M(s)=1\}}\mathbb{1}_{\{L_1^N(s)>0\}}\lambda_F^+ \frac{L_2^N(s)}{N} ds, \tag{1.19b}$$

$$\left\langle X_2^N \right\rangle(t) = \int_0^t \Phi_P \frac{L_0^N(s)}{N} ds + \int_0^t \mathbb{1}_{\{M(s)=1\}}\mathbb{1}_{\{L_1^N(s)>0\}}\lambda_F^+ \frac{L_2^N(s)}{N} ds. \tag{1.19c}$$

Remark 1.3.1

Notice that, for simplicity, we use the same process $(M(t))_{t\in\mathbb{R}_+}$ to define the evolution of every $(L^N(t))_{t\in\mathbb{R}_+}$. It has no impact on the result of Theorem 1.2.1 since the convergence result we obtain for $\left(\overline{L^N}\right)_{N\in\mathbb{N}}$ is in distribution.

If we now accelerate time by a factor N and scale down the coordinates of $\boldsymbol{L}^N$ by the same quantity (see (1.4)), a simple change of variables gives us that:

$$\overline{L_0^N}(t) = \overline{L_0^N}(0) + \int_0^t \mathbb{1}_{\{\overline{L_1^N}(u)>0\}}\lambda^- du - \int_0^t \mathbb{1}_{\{M(Nu)=0\}}\mathbb{1}_{\{\overline{L_1^N}(u)>0\}}\lambda^+\overline{L_0^N}(u)du - \int_0^t \Phi_P\overline{L_0^N}(u)du + \overline{X_0^N}(t), \tag{1.20a}$$

$$\overline{L_1^N}(t) = \overline{L_1^N}(0) + \int_0^t \mathbb{1}_{\{M(Nu)=0\}}\mathbb{1}_{\{\overline{L_1^N}(u)>0\}}\lambda^+\overline{L_0^N}(u)du - \int_0^t \mathbb{1}_{\{\overline{L_1^N}(u)>0\}}\lambda^- du + \int_0^t \mathbb{1}_{\{M(Nu)=1\}}\mathbb{1}_{\{\overline{L_1^N}(u)>0\}}\lambda_F^+\overline{L_2^N}(u)du + \overline{X_1^N}(t), \tag{1.20b}$$

$$\overline{L_2^N}(t) = \overline{L_2^N}(0) + \int_0^t \Phi_P\overline{L_0^N}(u)du - \int_0^t \mathbb{1}_{\{M(Nu)=1\}}\mathbb{1}_{\{\overline{L_1^N}(u)>0\}}\lambda_F^+\overline{L_2^N}(u)du + \overline{X_2^N}(t), \tag{1.20c}$$

where for $i \in \{0,1,2\}$ and $t \geq 0$,

$$\overline{X_i^N}(t) = \frac{X_i^N(Nt)}{N}.$$

We now have all the ingredients to prove Theorem 1.2.1.

3.1 - Tightness

The precise result that we prove in this section is the following.

Proposition 1.3.2 (Tightness)
The sequence of processes $\left(\overline{L^N}\right)_{N\geq 1}$ is C-tight in $D_\Delta[0,+\infty[$.

Proof of Proposition 1.3.2. The proof is rather standard. Since Δ is compact, all we have to check is that for every $T > 0$ and every $\varepsilon > 0$, there exists $\delta > 0$ such that

$$\limsup_{N\to\infty} \mathbb{P}\left(\omega\left(\overline{L^N}, \delta, T\right) > \varepsilon\right) \leq \varepsilon, \tag{1.21}$$

where

$$\omega\left(\overline{L^N}, \delta, T\right) = \sup_{\substack{0\leq s<t\leq T \\ |t-s|\leq\delta}} \left|\overline{L^N}(t) - \overline{L^N}(s)\right|.$$

Theorem 15.1 in [Bil99] will then enable us to conclude.

We proceed coordinate by coordinate, starting with $\overline{L_0^N}$. Let thus $T > 0$, $\varepsilon > 0$ and $\delta > 0$. Using (1.20a) and the fact that $\overline{L_0^N}(t) \leq 1$ for all $t \geq 0$, it is straightforward to show that for every $0 \leq s < t \leq T$ such that $|t-s| \leq \delta$, we have:

$$\left|\overline{L_0^N}(t) - \overline{L_0^N}(s)\right| \leq (\lambda^- + \lambda^+ + \Phi_P)\delta + \left|\overline{X_0^N}(t)\right| + \left|\overline{X_0^N}(s)\right|. \tag{1.22}$$

Choosing $\delta_0 = \varepsilon/(9(\lambda^- + \lambda^+ + \Phi_P))$ and $\delta < \delta_0$, we obtain that the first term on the r.h.s. of (1.22) is less than $\varepsilon/9$ with probability one. Next, by the Markov inequality we have:

$$\mathbb{P}\left(\sup_{u\in[0,T]}\left|\overline{X_0^N}(u)\right| > \frac{\varepsilon}{9}\right) \leq \frac{81}{\varepsilon^2 N^2}\,\mathbb{E}\left[\sup_{u\in[0,T]}\left(X_0^N(Nu)\right)^2\right].$$

Using Doob's maximal inequality and (1.19a), we can then write that:

$$\mathbb{E}\left[\sup_{u\in[0,T]}(X_0^N(Nu))^2\right] \leq 2\mathbb{E}\left[(X_0^N(NT))^2\right] = 2\mathbb{E}\left[\left\langle X_0^N\right\rangle(NT)\right] \leq 2(\lambda^- + \lambda^+ + \Phi_P)NT.$$

Combining the above, we obtain that:

$$\mathbb{P}\left(\sup_{u\in[0,T]}\left|\overline{X_0^N}(u)\right| > \varepsilon/9\right) \leq \frac{162(\lambda^- + \lambda^+ + \Phi_P)T}{\varepsilon^2 N}, \tag{1.23}$$

which converges to 0 as $N \longrightarrow \infty$. Consequently, with our choice of $\delta < \delta_0$, we obtain that:

$$\limsup_{N\to\infty}\mathbb{P}\left(\sup_{\substack{0\leq s<t\leq T\\ |t-s|\leq\delta}}\left|\overline{L_0^N}(t) - \overline{L_0^N}(s)\right| > \frac{\varepsilon}{3}\right) = 0.$$

The same reasoning applies to the two other coordinates $\overline{L_1^N}$ and $\overline{L_2^N}$ and yields similar thresholds $\delta_1, \delta_2 > 0$, which is more than enough to prove that if we take $\delta < \min(\delta_0, \delta_1, \delta_2)$,

$$\limsup_{N\to\infty}\mathbb{P}\left(\omega\left(\overline{L^N}, \delta, T\right) > \varepsilon\right) = 0.$$

This shows (1.21) and the proof of Proposition 1.3.2 is therefore complete. ✻

3.2 - Uniqueness of the Limit

Now that we have proved that the sequence of normalised processes is tight, it remains to show that the deterministic system described in Theorem 1.2.1 is its only possible limit as $N \longrightarrow +\infty$. This is what we do in this section. To do so, we shall use the following lemmas, whose proofs are postponed until the end of the section.

Lemma 1.3.3
For every $t > 0$, we have

$$\int_0^t (\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0))\overline{L_0^N}(s)ds \xrightarrow[N\to+\infty]{\mathbb{P}} 0.$$

Lemma 1.3.4
For every $t > 0$, we have

$$\int_0^t (\mathbb{1}_{\{M(Ns)=1\}} - \pi_M(1))\overline{L_2^N}(s)ds \xrightarrow[N\to+\infty]{\mathbb{P}} 0.$$

Proof of Theorem 1.2.1. Suppose that the sub-sequence $(\overline{L^{N_k}})_{k\in\mathbb{N}}$ converges weakly to a process

$$(l_0(t), l_1(t), l_2(t))_{t\in\mathbb{R}_+}.$$

By Proposition 1.3.2, we know that this process has continuous paths a.s. Furthermore, the estimates (1.23) (and their analogues for $\overline{X_1^N}$ and $\overline{X_2^N}$) guarantee that the limit of the quadratic variation process of each $\overline{L_i^N}$ is zero and $(l_0(t), l_1(t), l_2(t))_{t\in\mathbb{R}_+}$ is therefore deterministic.

For every $N \in \mathbb{N}$, let t_0^N be the first time at which the polymer has length 0:

$$t_0^N = \min\left\{ t \in \mathbb{R}_+ \mid \overline{L_1^N}(t) = 0 \right\}.$$

Similarly, let us define t_0 as the first time at which $l_1(t)$ takes the value 0. In both cases, if the process considered does not reach 0, we set the corresponding $t_0^{(N)}$ to $+\infty$. Since 0 is an absorbing state for the process $\left(\overline{L_1^N}(t)\right)_{t\in\mathbb{R}_+}$, we have:

$$\begin{cases} \overline{L_1^N}(t) > 0 & \text{if } t < t_0^N, \\ \overline{L_1^N}(t) = 0 & \text{if } t \geq t_0^N. \end{cases}$$

Therefore, the system of equations (1.20) can be rewritten as follows: for all $t < t_0^N$,

$$\begin{aligned}
\overline{L_0^N}(t) &= \overline{L_0^N}(0) + \lambda^- t - \lambda^+ \int_0^t \mathbb{1}_{\{M(Nu)=0\}} \overline{L_0^N}(u)du - \Phi_P \int_0^t \overline{L_0^N}(u)du + \overline{X_0^N}(t), \\
\overline{L_1^N}(t) &= \overline{L_1^N}(0) - \lambda^- t + \lambda^+ \int_0^t \mathbb{1}_{\{M(Nu)=0\}} \overline{L_0^N}(u)du + \lambda_F^+ \int_0^t \mathbb{1}_{\{M(Nu)=1\}} \overline{L_2^N}(u)du + \overline{X_1^N}(t), \\
\overline{L_2^N}(t) &= \overline{L_2^N}(0) + \Phi_P \int_0^t \overline{L_0^N}(u)du - \lambda_F^+ \int_0^t \mathbb{1}_{\{M(Nu)=1\}} \overline{L_2^N}(u)du + \overline{X_2^N}(t),
\end{aligned}$$

and for all $t \geq t_0^N$,

$$\begin{aligned}
\overline{L_0^N}(t) &= \overline{L_0^N}(0) + \lambda^- t_0^N - \lambda^+ \int_0^{t_0^N} \mathbb{1}_{\{M(Nu)=0\}} \overline{L_0^N}(u)du - \Phi_P \int_0^t \overline{L_0^N}(u)du + \overline{X_0^N}(t), \\
\overline{L_1^N}(t) &= 0, \\
\overline{L_2^N}(t) &= \overline{L_2^N}(0) + \Phi_P \int_0^t \overline{L_0^N}(u)du - \lambda_F^+ \int_0^{t_0^N} \mathbb{1}_{\{M(Nu)=1\}} \overline{L_2^N}(u)du + \overline{X_2^N}(t).
\end{aligned}$$

Splitting the integral $\int_0^{\min(t,t_0^{N_k})}$ into $\int_0^t - \int_{\min(t,t_0^{N_k})}^t$ and decomposing

$$\mathbb{1}_{\{M(N_k u)=0\}} = \pi_M(0) + (\mathbb{1}_{\{M(N_k u)=0\}} - \pi_M(0)),$$

we can rewrite the above system of equations for $t < t_0$ as:

$$\begin{aligned}
\overline{L_0^{N_k}}(t) &= \overline{L_0^{N_k}}(0) + \lambda^- t - (\lambda^+ \pi_M(0) + \Phi_P) \int_0^t \overline{L_0^{N_k}}(u)du + \overline{X_0^{N_k}}(t) \\
&\quad - \lambda^-(t - \min(t, t_0^{N_k})) + \lambda^+ \pi_M(0) \int_{\min(t,t_0^{N_k})}^t \overline{L_0^{N_k}}(u)du - \lambda^+ \int_0^{\min(t,t_0^{N_k})} (\mathbb{1}_{\{M(N_k u)=0\}} - \pi_M(0)) \overline{L_0^{N_k}}(u)du, \\
\overline{L_1^{N_k}}(t) &= \overline{L_1^{N_k}}(0) - \lambda^- t + \lambda^+ \pi_M(0) \int_0^t \overline{L_0^{N_k}}(u)du + \lambda_F^+ \pi_M(1) \int_0^t \overline{L_2^{N_k}}(u)du + \overline{X_1^{N_k}}(t) \\
&\quad + \lambda^-(t - \min(t, t_0^{N_k})) + \lambda^+ \int_0^{\min(t,t_0^{N_k})} (\mathbb{1}_{\{M(N_k u)=0\}} - \pi_M(0)) \overline{L_0^{N_k}}(u)du \\
&\quad + \lambda_F^+ \int_0^{\min(t,t_0^{N_k})} (\mathbb{1}_{\{M(N_k u)=1\}} - \pi_M(1)) \overline{L_2^{N_k}}(u)du - \lambda^+ \pi_M(0) \int_{\min(t,t_0^{N_k})}^t \overline{L_0^{N_k}}(u)du \\
&\quad - \lambda_F^+ \pi_M(1) \int_{\min(t,t_0^{N_k})}^t \overline{L_2^{N_k}}(u)du, \\
\overline{L_2^{N_k}}(t) &= \overline{L_2^{N_k}}(0) + \Phi_P \int_0^t \overline{L_0^{N_k}}(u)du - \lambda_F^+ \pi_M(1) \int_0^t \overline{L_2^{N_k}}(u)du + \overline{X_2^{N_k}}(t) \\
&\quad + \lambda_F^+ \pi_M(1) \int_{\min(t,t_0^{N_k})}^t \overline{L_2^{N_k}}(u)du - \lambda_F^+ \int_0^{\min(t,t_0^{N_k})} (\mathbb{1}_{\{M(N_k u)=1\}} - \pi_M(1)) \overline{L_2^{N_k}}(u)du.
\end{aligned}$$

First, for every $t < t_0$ we have the convergence in probability:

$$t - \min(t, t_0^{N_k}) \xrightarrow[k\to+\infty]{\mathbb{P}} 0, \tag{1.24}$$

and

$$\int_{\min(t,t_0^{N_k})}^{t} \overline{L_0^{N_k}}(u)du \xrightarrow[k\to+\infty]{\mathbb{P}} 0 \qquad \text{and} \qquad \int_{\min(t,t_0^{N_k})}^{t} \overline{L_2^{N_k}}(u)du \xrightarrow[k\to+\infty]{\mathbb{P}} 0. \tag{1.25}$$

Indeed, since for any $i \in \{0,2\}$ and any $u \le t$ we have $\overline{L_i^{N_k}}(u) \le 1$, then,

$$0 \le \mathbb{E}\left[\int_{\min(t,t_0^{N_k})}^{t} \overline{L_i^{N_k}}(u)du\right] \le t\mathbb{P}\left(t_0^{N_k} < t\right).$$

But

$$\limsup_{k\to\infty} \mathbb{P}\left(t_0^{N_k} < t\right) = \limsup_{k\to\infty} \mathbb{P}\left(\overline{L_1^{N_k}}(t) = 0\right) \le \inf_{\varepsilon>0} \mathbb{P}\left(l_1(t) \le \varepsilon\right) = 0,$$

where the last but first inequality comes from the fact that $(\overline{L_1^{N_k}}(t))_{k\ge1}$ converges in distribution to $l_1(t)$ and the last equality uses our assumption that $t < t_0$. This last result proves (1.24). The Markov inequality then gives us the two convergence results stated in (1.25).

Second, by assumption we have the convergence in distribution of $(\overline{L^{N_k}})_{k\ge1}$ to the deterministic process $(l_0(t), l_1(t), l_2(t))_{t\in\mathbb{R}_+}$ and therefore,

$$\left(\overline{L_0^{N_k}}(0), \overline{L_1^{N_k}}(0), \overline{L_2^{N_k}}(0)\right) \xrightarrow[k\to+\infty]{\mathbb{P}} \left(l_0(0), l_1(0), l_2(0)\right),$$

and

$$\left(\int_0^t \overline{L_0^{N_k}}(u)du\,,\ \int_0^t \overline{L_2^{N_k}}(u)du\right) \xrightarrow[k\to+\infty]{\mathbb{P}} \left(\int_0^t l_0(u)du\,,\ \int_0^t l_2(u)du\right). \tag{1.26}$$

Thirdly, by (1.23) the sequence $(\overline{X_0^N}(t))_{N\ge1}$ converges in probability to 0 (in fact this convergence is uniform over $[0,t]$) and equivalent results hold for $(\overline{X_1^N}(t))_{N\ge1}$ and $(\overline{X_2^N}(t))_{N\ge1}$.

Finally, Lemmas 1.3.3 and 1.3.4 ensure the convergence in probability:

$$\int_0^t (\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0))\overline{L_0^N}(s)ds \xrightarrow[N\to+\infty]{\mathbb{P}} 0$$

and

$$\int_0^t (\mathbb{1}_{\{M(Ns)=1\}} - \pi_M(1))\overline{L_2^N}(s)ds \xrightarrow[N\to+\infty]{\mathbb{P}} 0.$$

Combining the above and using Slutsky's theorem, we can conclude that for every $t < t_0$,

$$\begin{aligned}
l_0(t) &= l_0(0) + \lambda^- t - (\lambda^+\pi_M(0) + \Phi_P)\int_0^t l_0(u)du,\\
l_1(t) &= l_1(0) - \lambda^- t + \lambda^+\pi_M(0)\int_0^t l_0(u)du + \lambda_F^+\pi_M(1)\int_0^t l_2(u)du,\\
l_2(t) &= l_2(0) + \Phi_P\int_0^t l_0(u)du - \lambda_F^+\pi_M(1)\int_0^t l_2(u)du.
\end{aligned}$$

By the same type of arguments, we can prove that for every $t > t_0$,

$$\begin{aligned} l_0(t) &= l_0(0) + \lambda^- t_0 - \lambda^+ \pi_M(0) \int_0^{t_0} l_0(u)du - \Phi_P \int_0^t l_0(u)du, \\ l_1(t) &= 0, \\ l_2(t) &= l_2(0) + \Phi_P \int_0^t l_0(u)du - \lambda_F^+ \pi_M(1) \int_0^{t_0} l_2(u)du. \end{aligned}$$

As l_0 and l_2 are two continuous functions, the functions $t \mapsto \int_0^t l_0(u)du$ and $t \mapsto \int_0^t l_2(u)du$ are differentiable and for any $t \le t_0$, we have:

$$\begin{aligned} l_0'(t) &= \lambda^- - \pi_M(0)\lambda^+ l_0(t) - \Phi_P\, l_0(t), \\ l_1'(t) &= \pi_M(0)\lambda^+ l_0(t) + \pi_M(1)\lambda_F^+ l_2(t) - \lambda^-, \\ l_2'(t) &= \Phi_P l_0(t) - \pi_M(1)\lambda_F^+ l_2(t), \end{aligned} \tag{1.27}$$

while for any $t > t_0$,

$$l_0'(t) = -\Phi_P l_0(t), \qquad l_1'(t) = 0, \qquad l_2'(t) = \Phi_P l_0(t).$$

Solving these systems of equations, we obtain that the limiting process is necessarily the one given in Theorem 1.2.1 and therefore this limit is unique. Since $(\overline{L^N})_{N\ge1}$ is tight by Proposition 1.3.2, we can conclude that the full sequence converges in distribution and Theorem 1.2.1 is proved. ✻

It remains to prove Lemmas 1.3.3 and 1.3.4. As the proofs are similar, we only give the proof of Lemma 1.3.3.

Proof of Lemma 1.3.3. Let $t > 0$. We split the interval $[0, t]$ into intervals of length $\varepsilon > 0$. This gives us:

$$\begin{aligned} &\left| \int_0^t \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) \overline{L_0^N}(s)ds \right| \\ &\le \left| \sum_{k=1}^{\lfloor \frac{t}{\varepsilon} \rfloor} \int_{(k-1)\varepsilon}^{k\varepsilon} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) \overline{L_0^N}(s)ds \right| + \left| \int_{\lfloor \frac{t}{\varepsilon} \rfloor \varepsilon}^{t} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) \overline{L_0^N}(s)ds \right|, \\ &\le \sum_{k=1}^{\lfloor \frac{t}{\varepsilon} \rfloor} \left| \int_{(k-1)\varepsilon}^{k\varepsilon} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) \overline{L_0^N}\left((k-1)\varepsilon\right) ds \right| \\ &+ \sum_{k=1}^{\lfloor \frac{t}{\varepsilon} \rfloor} \left| \int_{(k-1)\varepsilon}^{k\varepsilon} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) \left(\overline{L_0^N}(s) - \overline{L_0^N}\left((k-1)\varepsilon\right) \right) ds \right| + \left| \int_{\lfloor \frac{t}{\varepsilon} \rfloor \varepsilon}^{t} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) \overline{L_0^N}\left(\lfloor t/\varepsilon \rfloor \varepsilon\right) ds \right| \\ &+ \left| \int_{\lfloor \frac{t}{\varepsilon} \rfloor \varepsilon}^{t} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) \left(\overline{L_0^N}(s) - \overline{L_0^N}\left(\lfloor t/\varepsilon \rfloor \varepsilon\right) \right) ds \right|. \end{aligned} \tag{1.28}$$

Now since $\overline{L_0^N}\left((k-1)\varepsilon\right) \in [0,1]$ for every k, we have:

$$\left| \int_{(k-1)\varepsilon}^{k\varepsilon} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) \overline{L_0^N}\left((k-1)\varepsilon\right) ds \right| \le \left| \int_{(k-1)\varepsilon}^{k\varepsilon} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) ds \right|, \tag{1.29}$$

and likewise:

$$\left| \int_{(k-1)\varepsilon}^{k\varepsilon} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) \left(\overline{L_0^N}(s) - \overline{L_0^N}\left((k-1)\varepsilon\right) \right) ds \right| \le \int_{(k-1)\varepsilon}^{k\varepsilon} \left| \overline{L_0^N}(s) - \overline{L_0^N}\left((k-1)\varepsilon\right) \right| ds. \tag{1.30}$$

We now control each of these terms.

First, since M is an irreducible Markov jump process on a finite state space, the ergodic theorem applies and gives us that for every $0 < s_1 < s_2$, we have the almost sure convergence:

$$\int_{s_1}^{s_2} \mathbb{1}_{\{M(Ns)=0\}} ds = \frac{1}{N} \int_{Ns_1}^{Ns_2} \mathbb{1}_{\{M(s)=0\}} ds \xrightarrow[N\to+\infty]{a.s.} \pi_M(0)(s_2 - s_1).$$

Applying this result with $s_1 = (k-1)\varepsilon$ and $s_2 = k\varepsilon$ (or $s_2 = t$ when $k = \lfloor t/\varepsilon \rfloor + 1$) for each of the finitely many values of k of interest, we obtain that:

$$\sum_{k=1}^{\lfloor \frac{t}{\varepsilon} \rfloor} \left| \int_{(k-1)\varepsilon}^{k\varepsilon} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) ds \right| + \left| \int_{\lfloor \frac{t}{\varepsilon} \rfloor \varepsilon}^{t} \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) ds \right| \xrightarrow[N\to+\infty]{a.s.} 0. \tag{1.31}$$

Second, recall from (1.20a) that for every $0 \le s' \le s$, we have:

$$\left| \overline{L_0^N}(s) - \overline{L_0^N}(s') \right| \le (\lambda^- + \lambda^+ + \Phi_P)(s - s') + \left| \overline{X_0^N}(s) - \overline{X_0^N}(s') \right|.$$

Applying this result with $s' = (k-1)\varepsilon$ and $s \in [(k-1)\varepsilon, k\varepsilon]$ and combining them with (1.23) (with $T = t$), we obtain, by choosing $C = \lambda^- + \lambda^+ + \Phi_P + 2/9$,

$$\lim_{N\to\infty} \mathbb{P}\left(\int_{(k-1)\varepsilon}^{k\varepsilon} \left| \overline{L_0^N}(s) - \overline{L_0^N}((k-1)\varepsilon) \right| ds \ge C\varepsilon^2 \right) = 0.$$

Consequently,

$$\begin{aligned}
&\lim_{N\to\infty} \mathbb{P}\left(\sum_{k=1}^{\lfloor \frac{t}{\varepsilon} \rfloor} \int_{(k-1)\varepsilon}^{k\varepsilon} \left| \overline{L_0^N}(s) - \overline{L_0^N}\left((k-1)\varepsilon\right) \right| ds + \int_{\lfloor \frac{t}{\varepsilon} \rfloor \varepsilon}^{t} \left| \overline{L_0^N}(s) - \overline{L_0^N}\left(\lfloor t/\varepsilon \rfloor \varepsilon\right) \right| ds \ge C\varepsilon^2 \left(\left\lfloor \frac{t}{\varepsilon} \right\rfloor + 1 \right) \right) \\
&\le \lim_{N\to\infty} \left(\sum_{k=1}^{\lfloor \frac{t}{\varepsilon} \rfloor} \mathbb{P}\left(\int_{(k-1)\varepsilon}^{k\varepsilon} \left| \overline{L_0^N}(s) - \overline{L_0^N}((k-1)\varepsilon) \right| ds \ge C\varepsilon^2 \right) + \mathbb{P}\left(\int_{\lfloor \frac{t}{\varepsilon} \rfloor \varepsilon}^{t} \left| \overline{L_0^N}(s) - \overline{L_0^N}(\lfloor t/\varepsilon \rfloor \varepsilon) \right| ds \ge C\varepsilon^2 \right) \right) \\
&= 0
\end{aligned} \tag{1.32}$$

Coming back to (1.28), combining (1.29), (1.30), (1.31) and (1.32), and observing that:

$$C\varepsilon^2 \left(\left\lfloor \frac{t}{\varepsilon} \right\rfloor + 1 \right) = \mathcal{O}(C\varepsilon t),$$

as ε tends to 0, we obtain that:

$$\lim_{N\to\infty} \mathbb{P}\left(\left| \int_0^t \left(\mathbb{1}_{\{M(Ns)=0\}} - \pi_M(0) \right) \overline{L_0^N}(s) ds \right| \ge (Ct + 1)\varepsilon \right) = 0.$$

Since ε was arbitrary, this proves Lemma 1.3.3. ✻

4 - Large Population Asymptotics

In this section, we prove Theorem 1.2.2. Recall that for every $N \in \mathbb{N}$, the Markov process $\overline{\mathcal{Z}^N}$ takes its values in the space $\mathcal{M}_{\mathcal{I}}^* \subset \mathcal{M}_{\mathcal{I}}^{(1)}$, respectively defined in (1.15) and (1.16). Let $\mathcal{D}$ be the set of all functions on $\mathcal{M}_{\mathcal{I}}^{(1)}$ of the form:

$$\varphi_f : Z \mapsto \varphi(\langle Z, f \rangle), \tag{1.33}$$

with f a bounded measurable function on $\mathcal{I}$ and φ a function of class C^1 on $\mathbb{R}$.

Since the rescaled process $\left(\overline{\mathcal{Z}_t^N}\right)_{t\in\mathbb{R}_+}$ is a Markov jump process with bounded rates for every $N \in \mathbb{N}$, its infinitesimal generator $\mathcal{G}^N$ applied to any $\varphi_f \in \mathcal{D}$ is given by: for every $Z \in \mathcal{M}_{\mathcal{I}}^{(1)}$,

$$\begin{aligned}
\mathcal{G}^N\varphi_f(Z) = \lambda^+ Z(M,1)\sum_{l=1}^{N-1} NZ(S,l)\left(\varphi_f\left(Z+\frac{\delta_{(S,l+1)}}{N}-\frac{\delta_{(S,l)}}{N}-\frac{\delta_{(M,1)}}{N}\right)-\varphi_f(Z)\right)\\
+\lambda_F^+ Z(P,1)\sum_{l=1}^{N-1} NZ(F,l)\left(\varphi_f\left(Z+\frac{\delta_{(F,l+1)}}{N}-\frac{\delta_{(F,l)}}{N}-\frac{\delta_{(P,1)}}{N}\right)-\varphi_f(Z)\right)\\
+\lambda^-\sum_{T\in\{S,F\}}\sum_{l=1}^{N} NZ(T,l)\left(\varphi_f\left(Z+\frac{\delta_{(T,l-1)}}{N}\mathbb{1}_{\{l\geq 2\}}-\frac{\delta_{(T,l)}}{N}+\frac{\delta_{(M,1)}}{N}\right)\varphi_f(Z)\right)\\
+\Phi_P NZ(M,1)\left(\varphi_f\left(Z+\frac{\delta_{(P,1)}}{N}-\frac{\delta_{(M,1)}}{N}\right)-\varphi_f(Z)\right)\\
+\Phi_F^+\sum_{l=1}^{N} NZ(S,l)\left(\varphi_f\left(Z+\frac{\delta_{(F,l)}}{N}-\frac{\delta_{(S,l)}}{N}\right)-\varphi_f(Z)\right)\\
+\Phi_F^-\sum_{l=1}^{N} NZ(F,l)\left(\varphi_f\left(Z+\frac{\delta_{(S,l)}}{N}-\frac{\delta_{(F,l)}}{N}\right)-\varphi_f(Z)\right),
\end{aligned} \tag{1.34}$$

where for simplicity we have written $Z(T,l)$ for $Z(\{(T,l)\})$. Furthermore, $\left(\overline{\mathcal{Z}_t^N}\right)_{t\in\mathbb{R}_+}$ satisfies the property that for every $\varphi_f \in \mathcal{D}$, the process $\left(\overline{\mathcal{Y}_t^N}(\varphi_f)\right)_{t\in\mathbb{R}_+}$ defined by:

$$\left(\overline{\mathcal{Y}_t^N}(\varphi_f)\right)_{t\in\mathbb{R}_+} \overset{def}{=} \left(\varphi_f\left(\overline{\mathcal{Z}_t^N}\right)-\varphi_f\left(\overline{\mathcal{Z}_0^N}\right)-\int_0^t \mathcal{G}^N\varphi_f\left(\overline{\mathcal{Z}_s^N}\right)ds\right)_{t\in\mathbb{R}_+} \tag{1.35}$$

is a mean-zero martingale with predictable quadratic variation given for all $t \geq 0$ by

$$\left\langle \overline{\mathcal{Y}^N}(\varphi_f)\right\rangle(t) = \int_0^t y_s^N(\varphi_f)ds, \tag{1.36}$$

where, writing this time $\overline{K_t^N}(T,l)$ for $\overline{\mathcal{Z}_t^N}(\{(T,l)\})$,

$$\begin{aligned}
y_t^N(\varphi_f) =&\lambda^+\overline{K_t^N}(M,1)\sum_{l=1}^{N-1} N\overline{K_t^N}(S,l)\left(\varphi_f\left(\overline{\mathcal{Z}_t^N}+\frac{\delta_{(S,l+1)}}{N}-\frac{\delta_{(S,l)}}{N}-\frac{\delta_{(M,1)}}{N}\right)-\varphi_f\left(\overline{\mathcal{Z}_t^N}\right)\right)^2\\
&+\lambda_F^+\overline{K_t^N}(P,1)\sum_{l=1}^{N-1} N\overline{K_t^N}(F,l)\left(\varphi_f\left(\overline{\mathcal{Z}_t^N}+\frac{\delta_{(F,l+1)}}{N}-\frac{\delta_{(F,l)}}{N}-\frac{\delta_{(P,1)}}{N}\right)-\varphi_f\left(\overline{\mathcal{Z}_t^N}\right)\right)^2\\
&+\lambda^-\sum_{T\in\{S,F\}}\sum_{l=1}^{N} N\overline{K_t^N}(T,l)\left(\varphi_f\left(\overline{\mathcal{Z}_t^N}+\frac{\delta_{(T,l-1)}}{N}\mathbb{1}_{\{l\geq 2\}}-\frac{\delta_{(T,l)}}{N}+\frac{\delta_{(M,1)}}{N}\right)-\varphi_f\left(\overline{\mathcal{Z}_t^N}\right)\right)^2\\
&+\Phi_P N\overline{K_t^N}(M,1)\left(\varphi_f\left(\overline{\mathcal{Z}_t^N}+\frac{\delta_{(P,1)}}{N}-\frac{\delta_{(M,1)}}{N}\right)-\varphi_f\left(\overline{\mathcal{Z}_t^N}\right)\right)^2\\
&+\Phi_F^+\sum_{l=1}^{N} N\overline{K_t^N}(S,l)\left(\varphi_f\left(\overline{\mathcal{Z}_t^N}+\frac{\delta_{(F,l)}}{N}-\frac{\delta_{(S,l)}}{N}\right)-\varphi_f\left(\overline{\mathcal{Z}_t^N}\right)\right)^2\\
&+\Phi_F^-\sum_{l=1}^{N} N\overline{K_t^N}(F,l)\left(\varphi_f\left(\overline{\mathcal{Z}_t^N}+\frac{\delta_{(S,l)}}{N}-\frac{\delta_{(F,l)}}{N}\right)-\varphi_f\left(\overline{\mathcal{Z}_t^N}\right)\right)^2.
\end{aligned}$$

Hence, the semi-martingale decomposition of $\varphi_f(\overline{\mathcal{Z}^N})$ reads:

$$\varphi_f\left(\overline{\mathcal{Z}_t^N}\right) = \varphi_f\left(\overline{\mathcal{Z}_0^N}\right) + \overline{\mathcal{V}_t^N}(\varphi_f) + \overline{\mathcal{Y}_t^N}(\varphi_f), \qquad t \geq 0 \tag{1.37}$$

with

$$\overline{\mathcal{V}_t^N}(\varphi_f) \overset{def}{=} \int_0^t \mathcal{G}^N \varphi_f\left(\overline{\mathcal{Z}_s^N}\right) ds.$$

4.1 - Tightness

As a first step in the proof of Theorem 1.2.2, we show the tightness of $(\overline{\mathcal{Z}^N})_{N\in\mathbb{N}}$.

Let $\mathcal{I}^\Delta$ be the one-point compactification of $\mathcal{I}$. In this subsection, we use the classical trick of seeing our sequence of processes as taking values in the space $\mathcal{M}_{\mathcal{I}^\Delta}$ of all finite measures on $\mathcal{I}^\Delta$. In the next subsection, we shall show that any limit point actually takes its values in $\mathcal{M}_{\mathcal{I}}^{(1)}$.

Proposition 1.4.1 (Tightness)
The sequence $\left(\overline{\mathcal{Z}^N}\right)_{N\in\mathbb{N}}$ is tight in $D_{\mathcal{M}_{\mathcal{I}^\Delta}}[0,+\infty[$.

Proof of Proposition 1.4.1. Since the total mass of each $\overline{\mathcal{Z}_t^N}$ is bounded by 1 and since the set of all measures on $\mathcal{I}^\Delta$ with total mass bounded by 1 is compact (hence the fact that we consider the one-point compactification of $\mathcal{I}$), the compact containment condition holds for $(\overline{\mathcal{Z}^N})_{N\geq 1}$ seen as a sequence in $D_{\mathcal{M}_{\mathcal{I}^\Delta}}[0,+\infty[$. By Theorem 3.9.1 in [EK86], tightness of $\left(\overline{\mathcal{Z}^N}\right)_{N\geq 1}$ will follow from the tightness of $\left(\varphi_f\left(\overline{\mathcal{Z}^N}\right)\right)_{N\geq 1}$ for every $\varphi_f \in \mathcal{D}$.

Let thus f be a bounded measurable function on $\mathcal{I}^\Delta$ and $\varphi \in \mathcal{C}^1(\mathbb{R})$. We use the standard Aldous-Rebolledo criterion [Ald78, Reb80]. First, the tightness of $\left(\varphi_f\left(\overline{\mathcal{Z}_t^N}\right)\right)_{N\geq 1}$ for every $t \geq 0$ is obvious from the boundedness of f and the fact that the total mass of $\overline{\mathcal{Z}_t^N}$ is bounded by 1.

Second, we have to prove that for any $T > 0$, any sequence of stopping times $(T_N)_{N\geq 1}$ bounded by T and for every $\varepsilon > 0$, there exists $\delta > 0$ and $N_0 \in \mathbb{N}$ such that for all $\forall N \geq N_0$ and $\theta \in [0,\delta]$,

$$\mathbb{P}\left(\left|\int_{T_N}^{T_N+\theta} \mathcal{G}^N \varphi_f\left(\overline{\mathcal{Z}_t^N}\right) dt\right| > \varepsilon\right) \leq \varepsilon, \tag{1.38}$$

and

$$\mathbb{P}\left(\left|\int_{T_N}^{T_N+\theta} y_t^N(\varphi_f) dt\right| > \varepsilon\right) \leq \varepsilon. \tag{1.39}$$

Plugging the definition of $\varphi_f(Z) = \varphi(\langle Z, f\rangle)$ in the expression for $\mathcal{G}^N\varphi_f$, Taylor expanding φ and writing $\|\varphi'\|_f$ for the supremum of φ' over $[-\|f\|_\infty, \|f\|_\infty]$, we obtain that:

$$\begin{aligned}\left|\mathcal{G}^N \varphi_f\left(\overline{\mathcal{Z}_t^N}\right)\right| \leq & \|\varphi'\|_f \|f\|_\infty \Bigg(3\lambda^+ \overline{K_t^N}(M,1) \sum_{l=1}^{N-1} \overline{K_t^N}(S,l) + 3\lambda_F^+ \overline{K_t^N}(P,1) \sum_{l=1}^{N-1} \overline{K_t^N}(F,l) \\ & + 3\lambda^- \sum_{T\in\{S,F\}} \sum_{l=1}^{N} \overline{K_t^N}(T,l) + 2\Phi_P \overline{K_t^N}(M,1) + 2\Phi_F^+ \sum_{l=1}^{N} \overline{K_t^N}(S,l) + 2\Phi_F^- \sum_{l=1}^{N} \overline{K_t^N}(F,l) \Bigg) + o(1),\end{aligned} \tag{1.40}$$

where the remainder term tends to 0 uniformly in N and t. By construction (since the total mass of each $\overline{\mathcal{Z}_t^N}$ is bounded by 1), for every type $T \in \{S, F\}$ we have:

$$\overline{K_t^N}(M,1) \le 1, \qquad \overline{K_t^N}(P,1) \le 1, \qquad \sum_{l=1}^{N} \overline{K_t^N}(T,l) \le 1,$$

and hence, there exists a constant $C_1 > 0$ such that:

$$\left|\mathcal{G}^N \varphi_f\left(\overline{\mathcal{Z}_s^N}\right)\right| \le \|\varphi'\|_f \|f\|_\infty \left(3\lambda^+ + 3\lambda_F^+ + 6\lambda^- + 2\Phi_P + 2\Phi_F^+ + 2\Phi_F^-\right) + o(1) = C_1 + o(1). \tag{1.41}$$

Therefore, taking $\delta = \varepsilon/(2C_1)$, we see that there exists $N_0 \in \mathbb{N}$ such that for every $N \ge N_0$ and $\theta \in [0, \delta]$,

$$\mathbb{P}\left(\left|\int_{T_N}^{T_N+\theta} \mathcal{G}^N \varphi_f\left(\overline{\mathcal{Z}_t^N}\right) dt\right| > \varepsilon\right) = 0$$

and (1.38) is satisfied.

We prove (1.39) in exactly the same way, noticing that the squared increments in the expression for $y_t^N(\varphi_f)$ give rise to a bound on $\left|y_t^N(\varphi_f)\right|$ of the form $C_2/N + o(1/N)$, where again the remainder term is uniform in t. Proceeding this way, we obtain a stronger result than (1.39), namely that the predictable quadratic variation of $\varphi_f\left(\overline{\mathcal{Z}^N}\right)$ vanishes as $N \to \infty$, uniformly over compact time intervals: by the Markov inequality and then Doob's maximal inequality,

$$\begin{aligned}
\mathbb{P}\left(\sup_{t\in[0,T]} \left|\overline{\mathcal{Y}_t^N}(\varphi_f)\right| > \varepsilon\right) &\le \frac{1}{\varepsilon^2}\mathbb{E}\left[\sup_{t\in[0,T]} \left|\overline{\mathcal{Y}_t^N}(\varphi_f)\right|^2\right] \\
&\le \frac{2}{\varepsilon^2}\mathbb{E}\left[\left\langle \overline{\mathcal{Y}^N}(\varphi_f)\right\rangle(T)\right] \\
&\le \frac{2C_2 T}{N\varepsilon^2} + o\left(\frac{1}{N}\right),
\end{aligned} \tag{1.42}$$

which tends to 0 as $N \to +\infty$. As a corollary to this result, we obtain that any potential limit for $\left(\overline{\mathcal{Z}^N}\right)_{N\ge1}$ must be deterministic. Since (1.42) implies (1.39), all the points in the Aldous-Rebolledo criterion have now been checked and we can conclude that $\left(\varphi_f\left(\overline{\mathcal{Z}^N}\right)\right)_{N\ge1}$ is tight in $D_{\mathbb{R}}[0,+\infty[$ for every $\varphi_f \in \mathcal{D}$, and finally that the sequence $\left(\overline{\mathcal{Z}^N}\right)_{N\ge1}$ is tight in $D_{\mathcal{M}_{I^\Delta}}[0,+\infty[$. ❋

4.2 - Large Population Limit

We now complete the proof of Theorem 1.2.2 by checking that all the assumptions of Theorem 4.8.10 in [EK86] are met.

Proof of Theorem 1.2.2. Tightness in $D_{\mathcal{M}_{I^\Delta}}[0,+\infty[$ was obtained in the previous subsection and we now want to prove that, if we define the operator $\mathcal{G}^\infty$ on $\mathcal{D}$ (the set of all functions of the form $\varphi(\langle\cdot, f\rangle)$ with

$\varphi \in \mathcal{C}^1(\mathbb{R})$ and $f : \mathcal{I} \longrightarrow \mathbb{R}$ bounded measurable) by: for every $Z \in \mathcal{M}_{\mathcal{I}}$,

$$\begin{aligned}\mathcal{G}^{\infty}\varphi_f(Z) \overset{def}{=} \varphi'(\langle Z, f\rangle)\Bigg(& \lambda^{+} Z(M,1)\sum_{l=1}^{+\infty} Z(S,l)\left(f(S,l+1)-f(S,l)-f(M,1)\right)\\ &+\lambda_F^{+} Z(P,1)\sum_{l=1}^{+\infty} Z(F,l)\left(f(F,l+1)-f(F,l)-f(P,1)\right)\\ &+\lambda^{-}\sum_{T\in\{S,F\}}\sum_{l=1}^{+\infty} Z(T,l)\left(f(T,l-1)\mathbb{1}_{\{l\geq 2\}}-f(T,l)+f(M,1)\right)\\ &+\Phi_P Z(M,1)\left(f(P,1)-f(M,1)\right)\\ &+\Phi_F^{+}\sum_{l=1}^{+\infty} Z(S,l)\left(f(F,l)-f(S,l)\right)+\Phi_F^{-}\sum_{l=1}^{+\infty} Z(F,l)\left(f(S,l)-f(F,l)\right)\Bigg),\end{aligned} \tag{1.43}$$

then for every $\varphi_f \in \mathcal{D}$ and $t, s > 0$, every $k \in \mathbb{N}$, $0 \leq t_1 < \cdots < t_k \leq t < t+s$ and $\beta_i \in C_b(\mathcal{M}_{\mathcal{I}})$, we have:

$$\lim_{N\to\infty} \mathbb{E}\left[\left(\varphi_f\left(\overline{\mathcal{Z}^N_{t+s}}\right)-\varphi_f\left(\overline{\mathcal{Z}^N_t}\right)-\int_t^{t+s}\mathcal{G}^{\infty}\varphi_f\left(\overline{\mathcal{Z}^N_u}\right)du\right)\left(\prod_{i=1}^{k}\beta_i\left(\overline{\mathcal{Z}^N_{t_i}}\right)\right)\right]=0. \tag{1.44}$$

Once this result is shown, we shall prove that any potential limiting process takes its value in $\mathcal{M}_{\mathcal{I}}^{(1)}$, on which every $\mathcal{G}^{\infty}\varphi_f$ is a continuous and bounded function, which will also prove that tightness of $(\overline{\mathcal{Z}^N})_{N\geq 1}$ holds in $D_{\mathcal{M}_{\mathcal{I}}^{(1)}}[0,+\infty[$ (by Corollary 3.9.3 in [EK86]). Finally, we shall prove that there is at most one possible limit for $(\overline{\mathcal{Z}^N})_{N\geq 1}$, which is the deterministic solution to the $D_{\mathcal{M}_{\mathcal{I}}^{(1)}}[0,+\infty[$-martingale problem associated to $(\mathcal{G}^{\infty}, \delta_{Z_0})$ or, equivalently, the $\mathcal{M}_{\mathcal{I}}^{(1)}$-valued solution to the system of equations stated in Theorem 1.2.2. Combining these arguments, we shall be able to use Theorem 4.8.10 in [EK86] to conclude that the limiting process exists and $(\overline{\mathcal{Z}^N})_{N\geq 1}$ indeed converges to it in $D_{\mathcal{M}_{\mathcal{I}}^{(1)}}[0,+\infty[$.

Let us thus start by showing (1.44). Using the decomposition (1.37), we have for every $N \in \mathbb{N}$,

$$\mathbb{E}\left[\left(\varphi_f\left(\overline{\mathcal{Z}^N_{t+s}}\right)-\varphi_f\left(\overline{\mathcal{Z}^N_t}\right)-\int_t^{t+s}\mathcal{G}^{N}\varphi_f\left(\overline{\mathcal{Z}^N_u}\right)du\right)\left(\prod_{i=1}^{k}\beta_i\left(\overline{\mathcal{Z}^N_{t_i}}\right)\right)\right]=0. \tag{1.45}$$

It thus remains to prove that:

$$\lim_{N\to\infty}\mathbb{E}\left[\int_t^{t+s}\left|\mathcal{G}^{N}\varphi_f\left(\overline{\mathcal{Z}^N_u}\right)-\mathcal{G}^{\infty}\varphi_f\left(\overline{\mathcal{Z}^N_u}\right)\right|du\right]=0, \tag{1.46}$$

since then,

$$\begin{aligned}&\left|\mathbb{E}\left[\left(\varphi_f\left(\overline{\mathcal{Z}^N_{t+s}}\right)-\varphi_f\left(\overline{\mathcal{Z}^N_t}\right)-\int_t^{t+s}\mathcal{G}^{\infty}\varphi_f\left(\overline{\mathcal{Z}^N_u}\right)du\right)\left(\prod_{i=1}^{k}\beta_i\left(\overline{\mathcal{Z}^N_{t_i}}\right)\right)\right]\right|\\ &\qquad=\left|\mathbb{E}\left[\left(\int_t^{t+s}\left(\mathcal{G}^{N}\varphi_f\left(\overline{\mathcal{Z}^N_u}\right)-\mathcal{G}^{\infty}\varphi_f\left(\overline{\mathcal{Z}^N_u}\right)\right)du\right)\left(\prod_{i=1}^{k}\beta_i\left(\overline{\mathcal{Z}^N_{t_i}}\right)\right)\right]\right|\\ &\qquad\leq\left(\prod_{i=1}^{k}\|\beta_i\|_{\infty}\right)\mathbb{E}\left[\int_t^{t+s}\left|\mathcal{G}^{N}\varphi_f\left(\overline{\mathcal{Z}^N_u}\right)-\mathcal{G}^{\infty}\varphi_f\left(\overline{\mathcal{Z}^N_u}\right)\right|du\right]\underset{N\to+\infty}{\longrightarrow}0\end{aligned}$$

and we obtain (1.44).

Coming back to the definition of $\mathcal{G}^N\varphi_f$ and performing a Taylor expansion of φ, it is straightforward to obtain that there exists a sequence $(\varepsilon_N)_{N\geq 1}$ tending to 0 such for any $Z \in \mathcal{M}_I^{(1)}$,

$$\left|\mathcal{G}^N\varphi_f(Z) - \mathcal{G}^\infty\varphi_f(Z)\right| \leq \varepsilon_N.$$

Using this result, the fact that for every $N \in \mathbb{N}$ and $t \geq 0$ we have $\left\langle \overline{\mathcal{Z}_t^N}, 1 \right\rangle \leq 1$, the bound (1.41) and the dominated convergence theorem, we can conclude that (1.46) holds true and the proof of (1.44) is complete.

Second, let us show that any potential limit point $(z_t)_{t\in\mathbb{R}_+}$ for $(\overline{\mathcal{Z}^N})_{N\geq 1}$ satisfies that $z_t \in \mathcal{M}_I^{(1)}$ for every $t \geq 0$. We already know from the subsection on tightness that each z_t is deterministic and that its total mass is bounded by 1. It only remains to show that z_t gives no mass to the "infinity" point of $\mathcal{I}^\Delta$ (i.e. to polymers of infinite length). Let thus $t \geq 0$ and N large enough for (1.17) to hold. Since the elongation rate of each polymer is bounded by $\lambda^* = \max\left(\lambda^+, \lambda_F^+\right)$, the probability that at time t a given polymer is of length at least $K_0 + k$ is bounded by the probability that the sum of k i.i.d exponential random variables with parameter λ^* is less than t, that is:

$$\mathbb{P}\left(\sum_{i=1}^k E_i < t\right) = \mathbb{P}\left(\exp\left(-\sum_{i=1}^k E_i\right) > e^{-t}\right) \leq e^t\left(\frac{\lambda^*}{\lambda^*+1}\right)^k, \tag{1.47}$$

where the last inequality uses the Markov inequality. Since by construction $\overline{\mathcal{Z}_t^N}$ has at most N atoms (or "polymers"), another use of the Markov inequality gives us that for every $k \in \mathbb{N}$, $\varepsilon \in (0,1)$ and N large enough (independently of k and ε),

$$\mathbb{P}\left(\left\langle \overline{\mathcal{Z}_t^N}, \mathbb{1}_{\{l\geq K_0+k\}}\right\rangle > \varepsilon\right) \leq \frac{1}{\varepsilon}\mathbb{E}\left[\left\langle \overline{\mathcal{Z}_t^N}, \mathbb{1}_{\{l\geq K_0+k\}}\right\rangle\right] \leq \frac{1}{\varepsilon N} N e^t\left(\frac{\lambda^*}{\lambda^*+1}\right)^k.$$

Since we suppose that $\overline{\mathcal{Z}_t^N}$ converges in distribution to z_t, we can take the limit as $N \longrightarrow +\infty$ in the above to obtain:

$$\mathbb{P}\left(\left\langle z_t, \mathbb{1}_{\{l\geq K_0+k\}}\right\rangle > \varepsilon\right) \leq \frac{1}{\varepsilon} e^t\left(\frac{\lambda^*}{\lambda^*+1}\right)^k. \tag{1.48}$$

Together with the fact that z_t is not random, we obtain that for every $\varepsilon \in (0,1)$, there exists $k(\varepsilon) \in \mathbb{N}$ such that $\varepsilon^{-1}e^t(\lambda^*/(\lambda^*+1))^{k(\varepsilon)} < 1$ and so:

$$\left\langle z_t, \mathbb{1}_{\{l\geq K_0+k(\varepsilon)\}}\right\rangle \leq \varepsilon.$$

Taking ε to 0, we can conclude that z_t puts no mass on infinitely long polymers, i.e. $z_t \in \mathcal{M}_I^{(1)}$.

Remark 1.4.2

The estimate (1.47) allows us to argue that for every $t \geq 0$, we have:

$$\langle z_t, p_l\rangle < \infty.$$

Indeed, in the stochastic pre-limiting model, the length of each of the at most N polymers is stochastically bounded by a pure birth model with birth rate λ^*, started at K_0, and therefore by (1.47) we have for every $t \geq 0$ and $m \in \mathbb{N}$,

$$\mathbb{E}\left[\left\langle \overline{\mathcal{Z}_t^N}, \mathbb{1}_{\{l=K_0+m\}}\right\rangle\right] \leq e^t\left(\frac{\lambda^*}{\lambda^*+1}\right)^m.$$

Passing to the limit as $N \longrightarrow +\infty$ yields:

$$k_t(S, K_0 + m) + k_t(F, K_0 + m) = \left\langle \mathcal{Z}_t, \mathbb{1}_{\{l=K_0+m\}} \right\rangle \leq e^t \left(\frac{\lambda^*}{\lambda^* + 1} \right)^m.$$

Summing over m for a fixed t, we obtain that:

$$\langle \mathcal{Z}_t, p_l \rangle = k_t(M, 1) + k_t(P, 1) + \sum_{T \in \{S,F\}} \sum_{l=1}^{K_0} l k_t(T, l) + \sum_{m=1}^{\infty} (K_0 + m) \left(k_t(S, K_0 + m) + k_t(F, K_0 + m) \right) < \infty. \tag{1.49}$$

Note however that these arguments are not sufficient to conclude that $\langle \mathcal{Z}_t, p_l \rangle = 1$ and therefore that the convergence in Theorem 1.2.2 holds in $\mathcal{M}_{\mathcal{I}}^*$ rather than in $\mathcal{M}_{\mathcal{I}}^{(1)}$.

⌞ ✤

Thirdly, let us check that any potential limit $\mathcal{Z}$ for $(\overline{\mathcal{Z}^N})_{N \geq 1}$ satisfies the system of equations stated in Theorem 1.2.2. As recalled earlier, we know from (1.42) that $\mathcal{Z}$ is deterministic (since we assumed that the limit of the initial value $\overline{\mathcal{Z}_0^N}$ was deterministic). Using (1.44) along the subsequence of $\left(\overline{\mathcal{Z}^N}\right)_{N \geq 1}$ converging to $\mathcal{Z}$, we can conclude that for every bounded measurable function f on $\mathcal{I}$ (taking $\varphi = \mathrm{Id}$),

$$\left(\langle \mathcal{Z}_t, f \rangle - \langle \mathcal{Z}_0, f \rangle - \int_0^t \mathcal{G}^{\infty} \mathrm{Id}_f(\mathcal{Z}_s) ds \right)_{t \in \mathbb{R}_+}$$

is a martingale, which is thus constant equal to 0. Taking now $f = \mathbb{1}_{\{(T,l)\}}$ for $(T, l) \in \mathcal{I}$, we obtain the integrated version of the differential equation for

$$k_t(T, l) = \mathcal{Z}_t(\{(T, L)\})$$

given in Theorem 1.2.2, namely:

$$k_t(T, L) = k_0(T, l) + \int_0^t \mathcal{G}^{\infty} \mathrm{Id}_{\mathbb{1}_{(T,l)}}(\mathcal{Z}_s) ds. \tag{1.50}$$

Since the integrand in the above expression is bounded, the function $t \mapsto k_t(T, l)$ is differentiable and we can therefore differentiate it to obtain the corresponding line in the system of Theorem 1.2.2.

Finally, let us prove that this system has at most one solution with trajectories in $\mathcal{M}_{\mathcal{I}}^{(1)}$. Let thus $(z_t)_{t \in \mathbb{R}_+}$ and $(\tilde{z}_t)_{t \in \mathbb{R}_+}$ be two solutions to the limiting system of equations in Theorem 1.2.2 such that $z_0 = \tilde{z}_0$ and $z_t, \tilde{z}_t \in \mathcal{M}_{\mathcal{I}}^{(1)}$ for all $t \geq 0$. Let f be a bounded measurable function on $\mathcal{I}$ such that $\|f\|_\infty \leq 1$. From the above, for every $t \geq 0$, we have:

$$|\langle z_t, f \rangle - \langle \tilde{z}_t, f \rangle| = \left| \int_0^t \left(\mathcal{G}^{\infty} \mathrm{Id}_f(z_s) - \mathcal{G}^{\infty} \mathrm{Id}_f(\tilde{z}_s) \right) ds \right|. \tag{1.51}$$

Let us consider each term in $\mathcal{G}^{\infty} \mathrm{Id}_f(z_s) - \mathcal{G}^{\infty} \mathrm{Id}_f(\tilde{z}_s)$ separately to obtain a bound on the integral on the r.h.s. of (1.51). We have:

$$\begin{aligned}
&\left| \lambda^+ \int_0^t \left(\sum_{l=1}^{\infty} (z_s(M, 1) z_s(S, l) - \tilde{z}_s(M, 1) \tilde{z}_s(S, l)) f(S, l + 1) \right) ds \right| \\
&\quad = \lambda^+ \left| \int_0^t \left(\sum_{l=1}^{\infty} (z_s(M, 1)(z_s(S, l) - \tilde{z}_s(S, l)) + (z_s(M, 1) - \tilde{z}_s(M, 1)) \tilde{z}_s(S, l)) f(S, l + 1) \right) ds \right| \\
&\quad = \lambda^+ \left| \int_0^t \left(z_s(M, 1) \left\langle z_s - \tilde{z}_s, \mathbb{1}_{(S,\cdot)} f(S, \cdot + 1) \right\rangle + \left\langle z_s - \tilde{z}_s, \mathbb{1}_{(M,1)} \right\rangle \sum_{l=1}^{\infty} \tilde{z}_s(S, l) f(S, l + 1) \right) ds \right|.
\end{aligned} \tag{1.52}$$

By assumption, the total mass of z_s and $\tilde{z}_s$ is bounded by 1, and so:

$$0 \leq z_s(M,1) \leq 1 \qquad \text{and} \qquad \left|\sum_{l=1}^{\infty} \tilde{z}_s(S,l) f(S,l+1)\right| \leq \|f\|_\infty \langle \tilde{z}_s, 1\rangle \leq 1.$$

As a consequence, the r.h.s. of (1.52) is bounded by:

$$\lambda^+ \int_0^t \left(\left| \left\langle z_s - \tilde{z}_s, \mathbb{1}_{(S,\cdot)} f(S,\cdot+1) \right\rangle \right| + \left| \left\langle z_s - \tilde{z}_s, \mathbb{1}_{(M,1)} \right\rangle \right| \right) ds \leq 2\lambda^+ \int_0^t \sup_{\|\phi\|_\infty \leq 1} |\langle z_s, \phi\rangle - \langle \tilde{z}_s, \phi\rangle| \, ds.$$

Proceeding in the same way for all the other terms in the generator, we obtain that:

$$|\langle z_t, f\rangle - \langle \tilde{z}_t, f\rangle| \leq (6\lambda^+ + 6\lambda_F^+ + 6\lambda^- + 2\Phi_P + 2\Phi_F^+ + 2\Phi_F^-) \int_0^t \sup_{\|\phi\|_\infty \leq 1} |\langle z_s, \phi\rangle - \langle \tilde{z}_s, \phi\rangle| \, ds,$$

and since this inequality holds for every bounded measurable f such that $\|f\|_\infty \leq 1$, we can write that for every $t \geq 0$,

$$\sup_{\|\phi\|_\infty \leq 1} |\langle z_t, \phi\rangle - \langle \tilde{z}_t, \phi\rangle| \leq (6\lambda^+ + 6\lambda_F^+ + 6\lambda^- + 2\Phi_P + 2\Phi_F^+ + 2\Phi_F^-) \int_0^t \sup_{\|\phi\|_\infty \leq 1} |\langle z_s, \phi\rangle - \langle \tilde{z}_s, \phi\rangle| \, ds.$$

By Gronwall's lemma,

$$\sup_{\|\phi\|_\infty \leq 1} |\langle z_t, \phi\rangle - \langle \tilde{z}_t, \phi\rangle| \leq \left(\sup_{\|\phi\|_\infty \leq 1} |\langle z_0, \phi\rangle - \langle \tilde{z}_0, \phi\rangle| \right) e^{(6\lambda^+ + 6\lambda_F^+ + 6\lambda^- + 2\Phi_P + 2\Phi_F^+ + 2\Phi_F^-)t} = 0.$$

Since every bounded measurable function can be written as a multiple of a function ϕ such that $\|\phi\|_\infty \leq 1$, we obtain that $\langle z_t, f\rangle = \langle \tilde{z}_t, f\rangle$ for every bounded measurable f, and hence $z_t = \tilde{z}_t$ for all $t \geq 0$ and the system of Theorem 1.2.2 has at most one solution with trajectories in $\mathcal{M}_I^{(1)}$.

We have now checked that all the conditions of Theorem 4.8.10 in [EK86] are satisfied and we can therefore conclude that, as N tends to infinity, $\left(\overline{\mathcal{Z}^N}\right)_{N\geq 1}$ converges in distribution in $D_{\mathcal{M}_I^{(1)}}[0,+\infty[$ to the unique solution with values in $\mathcal{M}_I^{(1)}$ of the system stated in Theorem 1.2.2. ✻

We end this section with some heuristics that are confirmed by the simulations discussed in the next section. In order to compare the large population limit of Theorem 1.2.2 with the fluid limit of Theorem 1.2.1, recall that informally, for a given N, L_0^N in the single polymer model corresponds to $K^N(M,1)$, L_2^N corresponds to $K^N(P,1)$ and L_1^N corresponds to M_{inF}^N (defined in (1.13)). The same correspondence can be made between the limiting objects, if we define:

$$m_{inF}(t) \overset{def}{=} \sum_{l=1}^{+\infty} l(k_t(S,l) + k_t(F,l)).$$

(This quantity is finite by Remark 1.4.2.) Moreover, writing for any $t \geq 0$,

$$\overline{F^\infty}(t) \overset{def}{=} \lim_{N\to+\infty} \overline{F^N}(t) = \sum_{l=1}^{+\infty} k_t(S,l) + \sum_{l=1}^{+\infty} k_t(F,l)$$

for the amount of "real" polymers at time t in the limiting population process, we see that the quantity $\sum_{l=1}^{+\infty} k_t(S,l)/\overline{F^\infty}(t)$ corresponds to the proportion of those polymers which are simple and we expect it to be close to $\pi_M(0)$ when t is large (recall that $\pi_M(0)$ is the proportion of time that a single polymer spends not bound with a formin at stationarity). Although we have no proof of this fact, Figures 1.6 and 1.7 show two examples where this convergence happens.

Finally, summing the corresponding equations in the system of ODE stated in Theorem 1.2.2, we obtain the following system which can be more easily compared with the result of Theorem 1.2.1.

Corollary 1.4.3 (Limit distribution in 3 sub-populations)
In the limit as $N \longrightarrow +\infty$, the distribution of monomers among the different sub-populations (free monomers, inserted in a polymer or in a G-actin/profilin complex) is described by the following system of ODE:

$$\begin{cases} \frac{d}{dt}k_t(M,1) &= \lambda^- \overline{F^\infty}(t) - \lambda^+ k_t(M,1)\sum_{l=1}^{+\infty} k_t(S,l) - \Phi_P k_t(M,1) \\ \frac{d}{dt}m_{inF}(t) &= \lambda^+ k_t(M,1)\sum_{l=1}^{+\infty} k_t(S,l) + \lambda_F^+ k_t(P,1)\sum_{l=1}^{+\infty} k_t(F,l) - \lambda^- \overline{F^\infty}(t) \\ \frac{d}{dt}k_t(P,1) &= \Phi_P k_t(M,1) - \lambda_F^+ k_t(P,1)\sum_{l=1}^{+\infty} k_t(F,l). \end{cases}$$

5 - Simulations

5.1 - Single Actin Polymer

In this section, we exhibit an interesting example of behaviour for a single long polymer, that highlights the role of the different proteins. The parameters chosen in these simulations are fairly arbitrary and do not come from biological measurements. Investigating more biologically-driven examples and the effect of sudden parameter changes on the behaviour of the system will be the object of future work.

Figure 1.3 shows the evolution of the system with $N = 1000$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_P = 10$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$, starting with a unique actin polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. That is, initially a third of the monomers in the system are free, another third are associated with a profilin and the rest sits in the polymer.

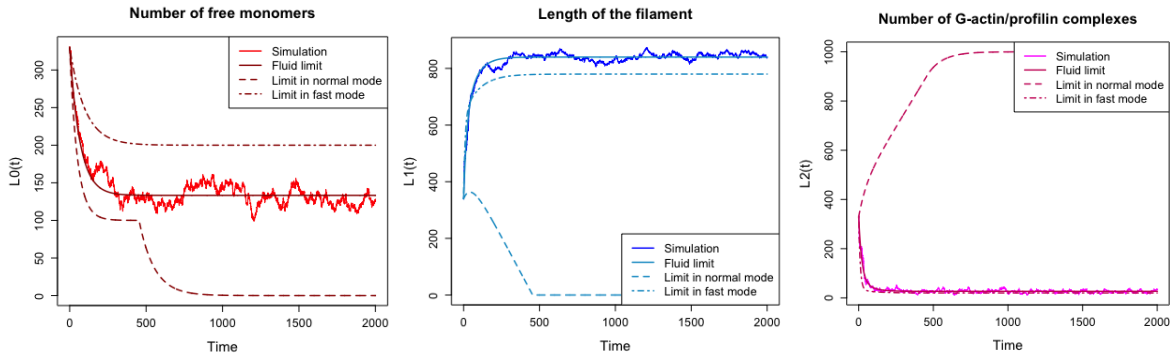


Figure 1.3 Evolution of the system described by the model for a single actin polymer with $N = 1000$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_P = 10$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$. Initially, there are a (unique) polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. Left: evolution of the number of free monomers. Middle: evolution of the length of the polymer. Right: evolution of the number of G-actin/profilin complexes. On each graph, we plot a single realisation of the stochastic system for $N = 1000$, the fluid limit given in Theorem 1.2.1 (i.e. N times $l_i(t)$), the fluid limit one would obtain by assuming that the system stays in normal mode and the fluid limit one would obtain by assuming that the system remains in fast mode all the time.

Figure 1.3 is divided into three graphs. On the left, we show the evolution in time of the number of free monomers, $L_0^N(t)$. The graph on the middle presents the evolution of the length of the polymer, $L_1^N(t)$, and the graph on the right corresponds to the evolution of the number of G-actin/profilin complexes, $L_2^N(t)$. Each graph displays four plots. The oscillating solid line shows a single realisation

of the stochastic system of Section 2.2.1 with $N = 1000$ and the parameters expounded above. The smooth solid line represents N times the fluid limit described in Theorem 1.2.1 (so that we compare numbers instead of proportions of monomers). The dashed line represents (N times) the fluid limit that we would obtain if we assumed that the polymer was constantly in normal mode. Finally, the alternated dashed line represents (N times) the fluid limit that we would obtain if we assumed that the polymer was always in fast mode.

In this particular example, we observe that if the system stays in normal mode, the length of the polymer converges quickly to 0 because the polymer looses monomers faster than it can gain them. This can be explained by the fact that the polymer and profilins compete for the same pool of free monomers with density dependent rates while the polymer also looses monomers through depolymerisation at a constant rate, so that once the pool of free monomers has fallen below a certain level because some fraction of monomers have been captured in G-actin/profilin complexes, polymerisation happens more slowly than depolymerisation. In fact, this is a general result, valid for any parameter values: if the polymer stays in normal mode all the time, then $t_0 < +\infty$. Indeed, in this case the system of equations (1.27) reads:

$$\begin{aligned} l_0'(t) &= \lambda^- - \lambda^+ l_0(t) - \Phi_P\, l_0(t), \\ l_1'(t) &= \lambda^+ l_0(t) - \lambda^-, \\ l_2'(t) &= \Phi_P l_0(t), \end{aligned} \tag{1.53}$$

for all $t \le t_0$. Solving the above, we obtain that for every $t \le t_0$,

$$\begin{aligned} l_0(t) &= \left(l_0(0) - \frac{\lambda^-}{\lambda^+ + \Phi_P} \right) e^{-(\lambda^+ + \Phi_P)t} + \frac{\lambda^-}{\lambda^+ + \Phi_P}, \\ l_1(t) &= 1 - l_0(t) - l_2(t), \\ l_2(t) &= l_2(0) + \frac{\Phi_P}{\lambda^+ + \Phi_P} \left(l_0(0) - \frac{\lambda^-}{\lambda^+ + \Phi_P} \right) \left(1 - e^{-(\lambda^+ + \Phi_P)t} \right) + \frac{\lambda^- \Phi_P}{\lambda^+ + \Phi_P} t. \end{aligned} \tag{1.54}$$

Supposing that $t_0 = +\infty$ would imply that $\lim_{t \to +\infty} l_2(t) = +\infty$, which is absurd since $l_2(t) \le 1$ for all $t \ge 0$.

Coming back to the example in Figure 1.3 we see that if, on the other hand, the polymer is constantly in fast mode or if it switches modes, then its length converges to a non-zero equilibrium. For other parameter values, it is possible to observe that the length of the polymer converges to a non-zero equilibrium when mode switching is allowed, while it converges to 0 if the system stays in fast mode. Writing the analogue of Condition (1.8) when the polymer is always in fast mode, we obtain:

$$1 - \frac{\lambda^-}{\Phi_P} - \frac{\lambda^-}{\lambda_F^+} < 0, \tag{1.55}$$

which tells us that, for instance, $\Phi_P \le \lambda^-$ is a sufficient condition for $t_0 < +\infty$ in this regime. Our numerical exploration did not enable us to find biologically reasonable parameter values for which the length of the polymer with mode switching converges to 0.

Noticeably, the polymer length at equilibrium is longer in the case of mode switching than in the case when the polymer remains in the state with the highest rate of polymerisation. As can be remarked on the two other graphs, this phenomenon is due to the fact that the switching between modes enables the polymer to use the two types of "resources" (free monomers and G-actin/profilin complexes) more efficiently, allowing the pool of free monomers not to be depleted (so that they can be consumed during the elongation of the polymer in normal mode, or be used to replenish the pool of complexes). More generally, in cases where $t_0 = +\infty$, the length of the polymer at equilibrium is larger with mode switching than in fast mode if and only if:

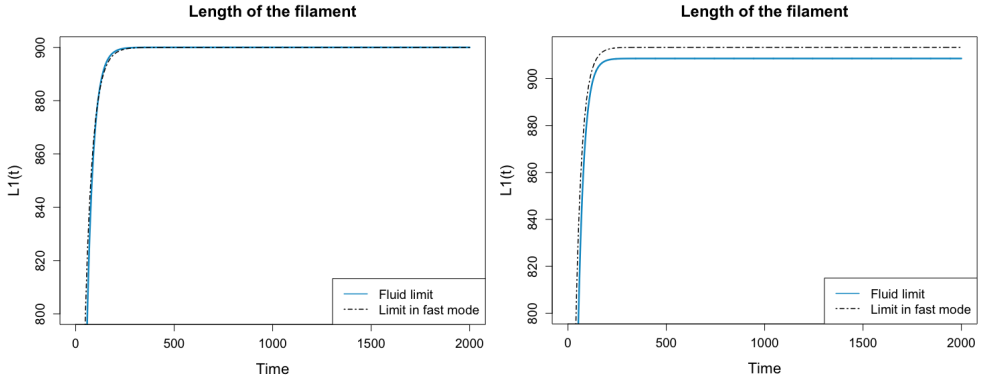
$$\Phi_P \in \left[0, \frac{1}{2} \left(\pi_M(1)\lambda^+ + \sqrt{\pi_M(1)\lambda^+ (\pi_M(1)\lambda^+ + 4\lambda_F^+)} \right) \right]. \tag{1.56}$$

Indeed, using Equation (1.7), we see that the length at equilibrium is $l_S^\infty = 1 - \frac{\lambda^-}{C_0} - \frac{\lambda^- \Phi_P}{C_0 C_2}$ and $l_F^\infty = 1 - \frac{\lambda^-}{\Phi_P} - \frac{\lambda^-}{\lambda_F^+}$ respectively with mode switching and in fast mode. This gives us

$$l_S^\infty > l_F^\infty \quad \Longleftrightarrow \quad -\Phi_P^2 + \pi_M(1)\lambda^+ \Phi_P + \pi_M(1)\lambda^+ \lambda_F^+ > 0, \tag{1.57}$$

and since by assumption $\Phi_P \geq 0$, we find Condition (1.56).

Figure 1.4 illustrates this criterion. Using the same parameters as in Figure 1.3, Condition (1.56) reads $\Phi_P \in [0, 25]$. In Figure 1.4a we set $\Phi_P = 25$ and we observe that the two limiting polymer lengths (in fast and alternating modes) are identical. In Figure 1.4b, we set $\Phi_P = 30$ and the length at equilibrium is larger if the system stays in fast mode.



(A) $\Phi_P = 25$ (B) $\Phi_P = 30$

Figure 1.4 Evolution of the length of the filament with $N = 1000$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$ for two values of Φ_P. Initially, there are a (unique) polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. On each graph, we plot the fluid limit given in Theorem 1.2.1 (i.e. N times $l_i(t)$) and the fluid limit assuming that the system stays in fast mode all the time.

5.2 - Population of Actin Polymers

Figure 1.5 displays the evolution of a population of actin polymers with the same set of parameters as in Figure 1.3, namely $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_P = 10$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$. In order to compare the limiting models for a single polymer and for a population of competing polymers, we start with an analogous initial distribution of monomers into free monomers, and monomers sequestered in polymers or complexes: 33, 3% of free monomers, 33, 4% of monomers included in a polymer and 33, 3% of monomers in G-actin/profilin complexes. Moreover, to match the conditions of Theorem 1.2.2 we start with 50% of the polymers being simple and 50% being associated with a formin, all of them being of length 30. Finally, we use the Euler explicit method with a time pitch $dt = 0.001$. Note that here we only compare the limits obtained in Theorems 1.2.1 and 1.2.2, and therefore we directly consider $N = +\infty$.

In Figure 1.5, the same quantities are shown as in Figure 1.3, but for a population of actin polymers. Figure 1.5 is divided into three graphs. On the left, we show the evolution in time of the proportion of free monomers in the population, $k_t(M, 1)$. The graph on the middle presents the evolution of the proportion of monomers included in a polymer,

$$m_{inF}(t) = \sum_{l=1}^{+\infty} l(k_t(S, l) + k_t(F, l)),$$

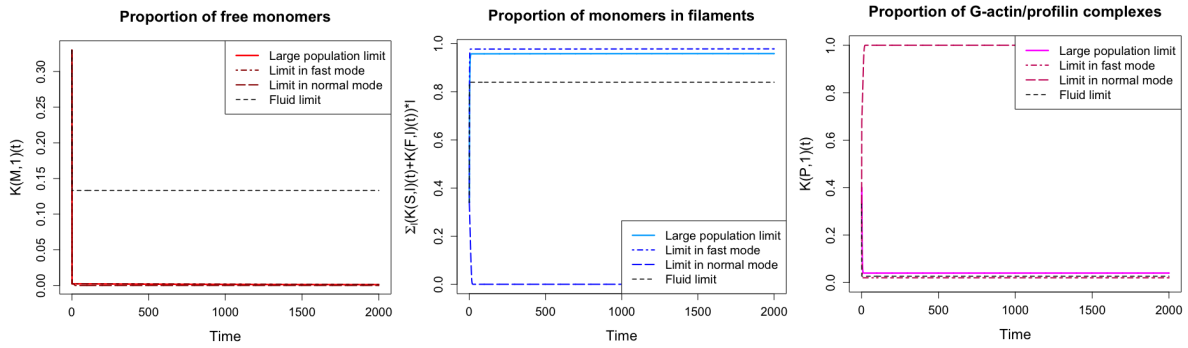


Figure 1.5 Evolution of the system described by the limiting model for a population of actin polymers with $dt = 0.001$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_P = 10$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$, starting with 33% of free monomers, 33% of G-actin/profilin complexes and 34% of monomers included within polymers of length 30. Left: evolution of the proportion of free monomers. Middle: evolution of the proportion of monomers included in a polymer. Right: evolution of the proportion of G-actin/profilin complexes. On each graph, we plot the fluid limit given in Theorem 1.2.1, the large population limit given in Theorem 1.2.2, the large population limit obtained by assuming that the system stays in normal mode and the large population limit obtained by assuming that the system stays in fast mode.

and the graph on the right gives the evolution in time of the proportion G-actin/profilin complexes available in the population, $k_t(P,1)$. On each graph, we display four plots. The solid line represents the large population limit described in Theorem 1.2.2. The coloured dashed line (resp. coloured alternated dashed line) represents the large population limit that we would obtain if we assumed that all polymers were stuck in normal mode (resp. fast mode — in these cases, initially all the polymers are simple, resp. associated with a formin). Finally, the thin black dashed line represents the equilibrium state in the fluid limit obtained in the single polymer model and shown in Figure 1.3.

In this particular example, we can observe that the qualitative behaviour of a population of polymers bears some similarity with the fluid limit for a single long polymer. Indeed, if all polymers are blocked in normal mode, the proportion of monomers inserted in a polymer converges quickly to 0 because a large fraction of free monomers is quickly captured by profilins and cannot be converted again into free monomers (since there is no accelerated polymerisation), so that they can no longer be used in polymer elongation. In contrast, when the polymers can switch modes or when they all remain in fast mode, the proportion of monomers inserted in a polymer stabilises to a non-zero value. However, this time the "equilibrium" is lower in the case with switching than in the constantly fast mode. This can be explained by the fact that the large number of polymers in the system compete for the same order of magnitude of free monomers (themselves also captured by profilins) and so the pool of free monomers is constantly kept at a low level, which impedes the elongation of the simple polymers. In this case, mode switching does not allow a better use of the different pools of monomers.

Although we are not able to prove analytically that, in the limiting process obtained in Theorem 1.2.2, the proportion of polymers bound with a formin approaches the stationary probability $\pi_M(1)$ that a single polymer is in "fast" mode (in the terminology of the single polymer model of Section 2.2), we checked this intuition by simulation. In Figure 1.6, we plot the evolution of the quantities:

$$\frac{\sum_{l=1}^{+\infty} k_t(S,l)}{\overline{F^\infty}(t)} \qquad \text{and} \qquad \frac{\sum_{l=1}^{+\infty} k_t(F,l)}{\overline{F^\infty}(t)},$$

where as earlier,

$$\overline{F^\infty}(t) = \sum_{l=1}^{+\infty} k_t(S,l) + \sum_{l=1}^{+\infty} k_t(F,l)$$

is the total amount of "real" polymers (thus excluding monomers and complexes) in the system at time t. The parameters and initial state are the same as before. Since $\Phi_F^+ = \Phi_F^-$, in this case we have $\pi_M(0) = 0.5 = \pi_M(1)$ and the initial proportion of polymers bound with a formin is also 0.5. As we see in Figure 1.6, after an initial very small shift (whose origin is not clear), the proportion of polymers converges again to the value 0.5, which confirms our intuition.

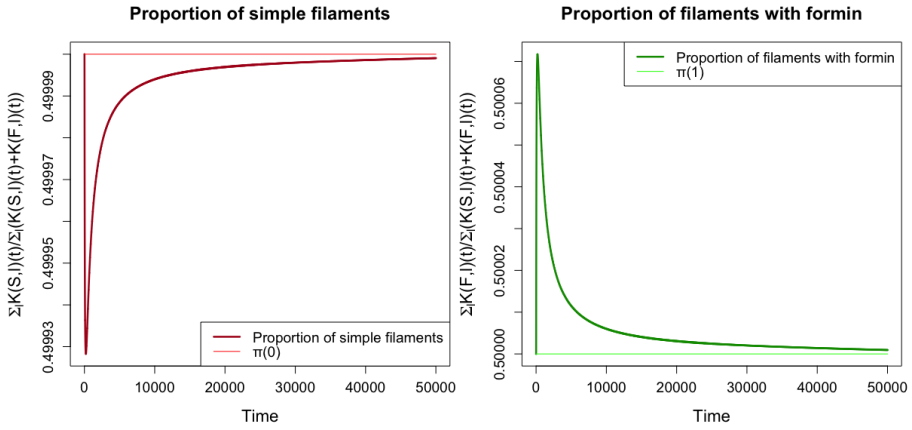


Figure 1.6 Convergence of the proportion of simple polymers (*resp.*, polymers bound with a formin) towards $\pi_M(0)$ (resp. $\pi_M(1)$) with $dt = 0.001$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_P = 10$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$, starting with 33% of free monomers, 33% of G-actin/profilin complexes and 34% of monomers included in polymers. Initially, we start with 50% of simple polymers and 50% of polymers bound with a formin, all of length 30.

Using this approximation, the system of ODE proposed in Corollary 1.4.3 can be rewritten for large times as:

$$\begin{cases} \frac{d}{dt}k_t(M,1) &= \lambda^-\overline{F^\infty}(t) - \lambda^+\pi_M(0)\overline{F^\infty}(t)k_t(M,1) - \Phi_P k_t(M,1) \\ \frac{d}{dt}m_{inF}(t) &= \lambda^+\pi_M(0)\overline{F^\infty}(t)k_t(M,1) + \lambda_F^+\pi_M(1)\overline{F^\infty}(t)k_t(P,1) - \lambda^-\overline{F^\infty}(t) \\ \frac{d}{dt}k_t(P,1) &= \Phi_P k_t(M,1) - \lambda_F^+\pi_M(1)\overline{F^\infty}(t)k_t(P,1). \end{cases} \tag{1.58}$$

If we compare this system to the limiting evolution obtained in (1.27) for a single polymer, we observe that in the above, the elongation rates (spontaneous or with a formin) and the depolymerisation rate are multiplied by the amount of polymers $\overline{F^\infty}$. Therefore, it appears that elongation and depolymerisation are slowed down by the competition between polymers whereas the production of G-actin/profilin complexes is not impacted.

Since in Figure 1.6 the initial proportion of polymers bound with a formin was set to be equal to its (supposed) equilibrium value, we explored what happens in other cases by choosing a different set of parameters. Figure 1.7 displays the same quantities as Figure 1.6 but with the following set of parameters: $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_P = 10$, $\Phi_F^+ = 8$ and $\Phi_F^- = 2$. We start again with the same initial state (33% of free monomers, 34% of monomers included in a polymer, 33% of monomers in G-actin/profilin complexes, 50% of polymers being simple, 50% being associated with a formin and all polymers being of length 30). By (1.3), this time we have $(\pi_M(0), \pi_M(1)) = (0.2, 0.8)$ and Figure 1.7 confirms again the suspected convergence (with another initial overshoot by a very small amount that we cannot explain).

Since the main difference between the systems (1.27) and (1.58) lies in the presence of the coefficient $\overline{F^\infty}$ in the latter, it is natural to investigate the evolution in time of the amount of "real" polymers in the limiting population model. In Figure 1.8, we plot $\left(\overline{F^\infty}(t)\right)_{t\in\mathbb{R}_+}$ using the same parameter values as in Figure 1.5. We observe in particular that the amount of polymers seems to decrease towards a limit

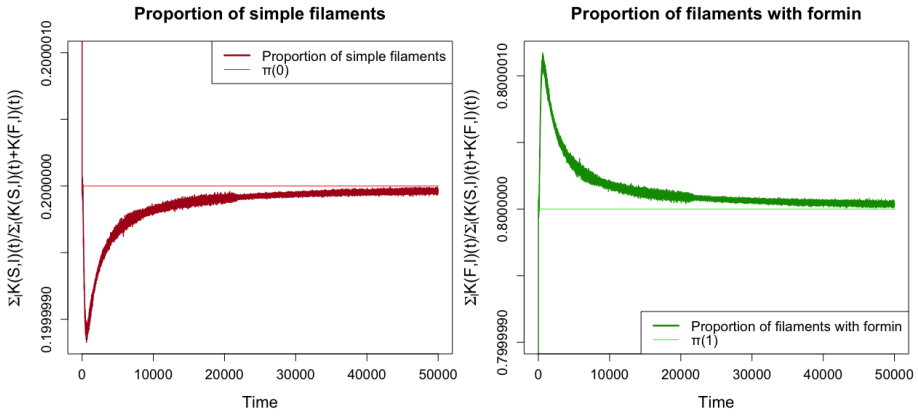


Figure 1.7 Convergence of the proportion of simple polymers (resp. polymers bound with a formin) towards $\pi_M(0)$ (resp. $\pi_M(1)$) with $dt = 0.001$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_P = 10$, $\Phi_F^+ = 8$ and $\Phi_F^- = 2$, starting with 33% of free monomers, 33% of G-actin/profilin complexes and 34% of monomers included in polymers. Initially, we start with 50% of simple polymers and 50% of polymers with formin, all of length 30.

of the order of 10^{-3}. Of course this (empirical) stabilisation to a non-zero value is only one example of potential limit for the decreasing process $\overline{F^\infty}$ and other sets of parameters may lead to a limiting amount of "real" polymers equal to 0.

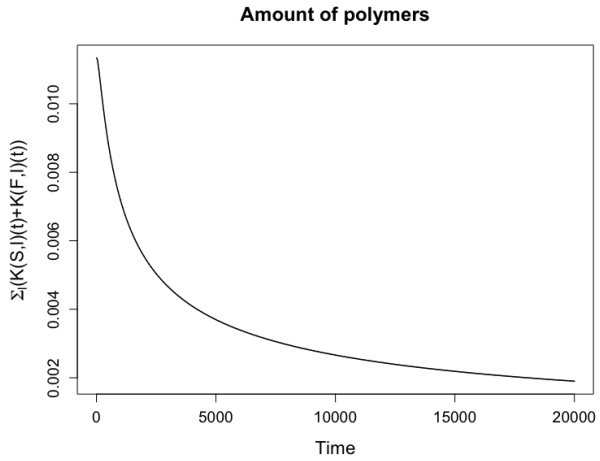


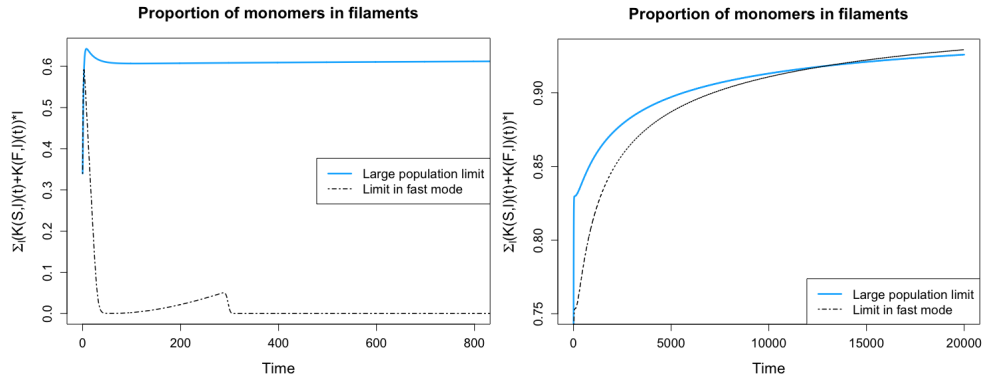
Figure 1.8 Evolution of the amount of "real" polymers $\overline{F^\infty}$ with $dt = 0.001$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_P = 10$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$, starting with 33% of free monomers, 33% of G-actin/profilin complexes and 34% of monomers included within polymers of length 30.

As in the previous section, we can also try to use the approximation (1.58) to compare the amount of monomers inserted in a polymer (as a proxy for the average polymer length) when the polymers can switch modes and when they are always in fast mode. We only proceed heuristically, assuming that $\overline{F^\infty}(t) \underset{t\to\infty}{\longrightarrow} f_\infty > 0$. Finding an analogue of the criterion (1.56) to characterise the set of parameters for which the fraction of monomers included in a polymer at equilibrium is larger with mode switching than if the system stays in fast mode is trickier than in the case of a single polymer. Indeed, the limit f_∞ itself depends on the value of Φ_P. However, noticing that the system (1.58) can be obtained from (1.27) by replacing λ^- by $\lambda^- f_\infty$, λ^+ by $\lambda^+ f_\infty$ and λ_F^+ by $\lambda_F^+ f_\infty$, the candidate criterion derived from (1.56) can be written:

$$\Phi_P \in \left[0, \frac{1}{2}\left(\pi_M(1)\lambda^+ f_\infty + \sqrt{\pi_M(1)\lambda^+ f_\infty^2(\pi_M(1)\lambda^+ + 4\lambda_F^+)}\right)\right]. \tag{1.59}$$

In particular, we see that when the limiting amount of "real" polymers f_∞ is small, the proportion of

monomers included in a polymer at equilibrium is greater with mode switching than if the system stays in the fast mode only when Φ_P is itself very small. Figure 1.9 confirms that such a regime is numerically possible, but it is not clear that values of Φ_P that are so low compared to other parameter values are biologically relevant.



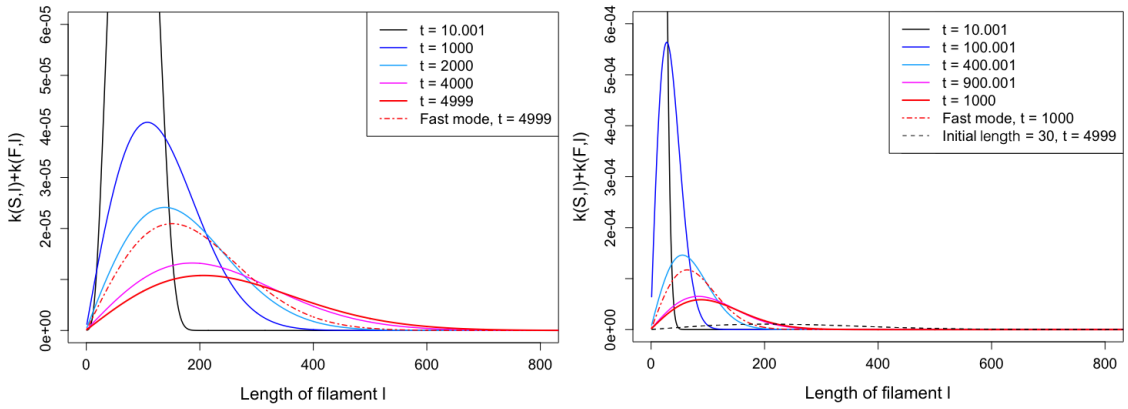
(A) $\Phi_P = 0.001$ (B) $\Phi_P = 0.01$

Figure 1.9 Time evolution of the proportion of monomers included in a polymer with $dt = 0.001$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$ for two values of Φ_P, starting with 33% of free monomers, 33% of G-actin/profilin complexes and 34% of monomers included within polymers of length 30. On each graph, we plot the large population limit given in Theorem 1.2.2 and the large population limit obtained by assuming that the system stays in fast mode.

Let us now consider the polymer lengths in our large population limit to see what is the impact of mode switching on their distribution. Figure 1.10 presents the evolution of the distribution of the polymer lengths with the same set of parameters as in Figure 1.5. We start with the same proportion as previously, 33, 3% of free monomers, 33, 3% of G-actin/profilin complexes and 33, 4% of monomers included in a polymer. In Figure 1.10a we start with polymers of length 30, whereas on Figure 1.10b we start with polymers of length 3. In each graph, we plot the distribution of the polymer lengths obtained in Theorem 1.2.2 at different times and in dashed line we plot the distribution at which the system seems to stabilise (in the simulations) when we assume that all polymers are blocked in fast mode.

First, comparing Figure 1.10a and Figure 1.10b, we see that the distribution in length depends on the initial condition. Indeed, if the polymers are initially shorter, at later times they are globally shorter but more numerous. In contrast, if we start with longer polymers, at later times they are longer, the variance in polymer length is larger but there are fewer polymers.

Second, let us compare the distributions of the polymer lengths at different times, the distribution after stabilisation (in red) and the distribution assuming that all polymers remain in fast mode (red dashed line) in Figure 1.10a. As in Figure 1.5, we see that the proportion of polymers is larger when the polymers are blocked in fast mode. But we can also observe that the polymer lengths are globally shorter than when polymers switch modes. We can therefore conclude that mode switching allows to have longer polymers but limits the proportion of monomers included in polymers, and thus it limits the number of polymers.



(A) Initial polymers of length 30

(B) Initial polymers of length 3

Figure 1.10 Evolution of the distribution of the polymer lengths with $dt = 0.001$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 100$, $\Phi_P = 10$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$, starting with 33% of free monomers, 33% of G-actin/profilin complexes and 34% of monomers included in polymers. In each graph, we plot the distribution at different times. The red dashed line represents the distribution after stabilisation when assuming that all polymers are blocked in fast mode. To ease the comparison between the two graphs, the black dashed line in (B) corresponds to the red solid line in (A).

Chapter II

Model for a Single Branched Polymer

Contents

This chapter presents an adaptation of the model developed by Patric Karl Glöde in his PhD thesis entitled *Dynamics of Genealogical Trees for Autocatalytic Branching Processes* [Glö13]. Some passages are thus very similar, but others have been completely revised. We can note two main changes. Firstly, our elongation process is not deterministic. As each extremity can grow randomly and independently of each other, the genealogical distance is no longer an ultrametric. Our framework is therefore quite different. Secondly, we consider an offspring distribution whose mean is not equal to 1. Choosing $p_N = \frac{1}{2} \pm \frac{a}{N}$ for the offspring distribution (define in Section 3.1.1 page 74) is a very small change in the model but it implies several changes in the results. For example, the total mass is no longer a martingale and we had to revise the proof of tightness as N tends to infinity. It also implies some modifications in the expression for the generator describing the binding dynamics. So, we need to rewrite all the technical lemmas in order to prove our convergence result. In most cases, we could use similar tricks as those of [Glö13], but we had to adapt the model and rewrite the majority of the proofs.

1 - Overview

In this model, we consider a single branched polymer of actin that interacts with two accessory proteins: Arp2/3 and cofilin. As a reminder, an Arp2/3 complex can bind to any monomer of an actin polymer and then create a new branch. Cofilin also binds to actin polymers but causes the fragmentation of the polymer.

Let N be the order of magnitude of the total number of monomers in the population. In this chapter, we will not assume that the total number of monomers in the system is constrained. Let $\lambda^+ > 0$ be a positive constant that characterises the rate of spontaneous elongation, $\beta > 0$ be a positive constant that characterises the binding rate of accessory proteins and $a \in \mathbb{R}$ be a constant that will allow us to give an advantage to one of the accessory proteins over the other. A priori, the constants defined above are individual rates of events occurring to a given branch of the polymer. Because we want to consider very large polymers (containing of the order of N monomers), in what follows we will be interested in a time-scale of the form $(Nt)_{t\in\mathbb{R}_+}$ on which events will therefore happen N times faster.

We see the branched polymer as a tree, modelled thanks to a metric measure triple (see Definition 2.2.2). In this chapter, we will consider a polymer with a large number of extremities (of the order of N) and short branches (of the order of 1). Thereby, we can assume that Arp2/3 complexes and cofilin do not have enough space to bind to an internal branch. Therefore, we assume that binding of proteins occurs only on the extremities.

Three types of events can occur, illustrated in Figure 2.1:

1. **Elongation:** Each branch can elongate by catching a free monomer at rate $N\lambda^+$, independently of each other.
2. **Binding of a protein:** Each branch can become bound to an Arp2/3 complex or a cofilin at rate $N\beta$, independently of each other. A binding event changes the structure of the polymer by creating a new branch or removing the chosen branch.
   (a) **Branching:** With probability $p_2^N = \frac{1}{2} + \frac{a}{N}$, the protein which binds to the polymer is an

Arp2/3 complex and a new branch of length 0 is created. In that case, a new extremity is created.

(b) **Fragmentation:** With probability $p_0^N = \frac{1}{2} - \frac{a}{N}$, the protein which binds to the polymer is a cofilin and an extremity as well as its entire branch disappear.

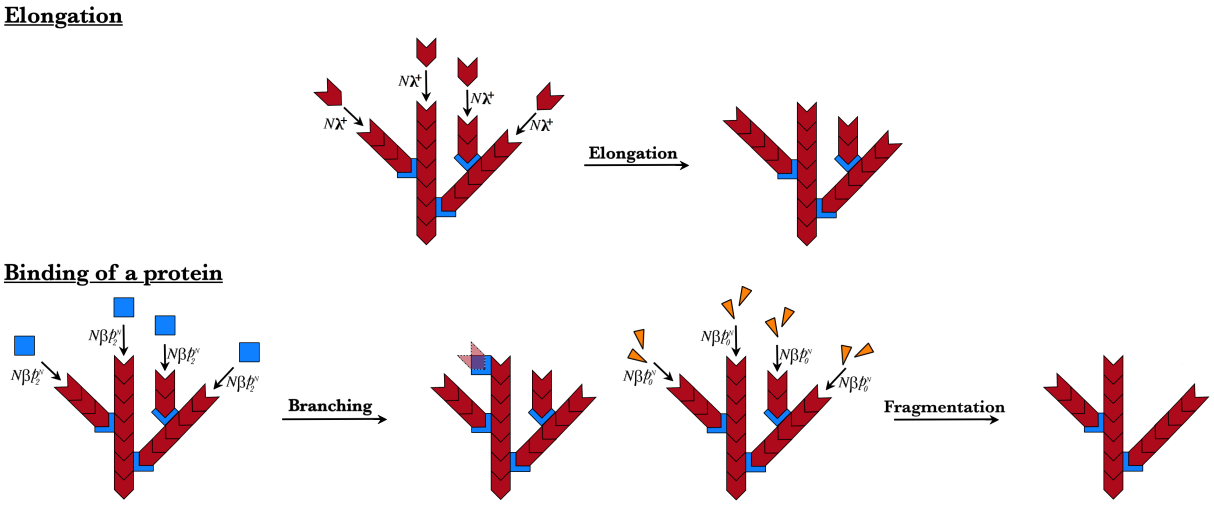


Figure 2.1 Schematic representation of the model for the dynamics of a single branched actin polymer and its interaction with Arp2/3 and cofilin. Red single units represent monomers, blue squares represent Arp2/3 complexes and pairs of orange triangles represent cofilin. Sets of red units and blue squares represent branched polymers which can be seen as trees. For each possible event (elongation, branching and fragmentation), the rate is recalled on the left-hand-side and the event affects one of the branches on the r.h.s. For the branching event, the transparent unit corresponds to a site that can receive a monomer (but there is no monomer yet on the new Arp2/3 complex — see the second branch starting from the left in the tree on the r.h.s.).

2 - Basic Definitions

In this whole chapter, we will use the formalism of metric measure spaces — introduced by A. Greven, P. Pfaffelhuber and A. Winter in [GPW09] and studied by P.K. Glöde [Glö13] and later by A. Depperschmidt *et al.* [DGP12, DG19] — to study tree-valued Markov processes. We therefore begin by recalling the notions that are essential to understand what follows. We also introduce some basic definitions and general notation that will be useful in defining our tree-valued process.

If $(E, \mathcal{O})$ is a topological space, we denote the set of finite non-negative measures on the Borel-σ-algebra $\mathcal{B}(E)$ on E by $\mathcal{M}_f(E)$. The set of probability measures are denoted by $\mathcal{M}_1(E)$.

Let us first recall the definition of the image measure in order to fix the notation.

Definition 2.2.1 (Image Measure of μ)
Let $\mu \in \mathcal{M}_f(E)$. If $(E', \mathcal{O}')$ is another topological space and if $\varphi : E \longrightarrow E'$ is a measurable function, the *image measure of* μ with respect to φ, denoted by $\mu \circ \varphi^{-1}$, is defined, for all $B \in \mathcal{B}(E')$, by:

$$\mu \circ \varphi^{-1}(B) = \mu\left(\varphi^{-1}(B)\right).$$

We can now define a metric measure triple and metric measure spaces that we will use throughout this chapter.

Definition 2.2.2 (Metric Measure Triple)

i - Assume $E \subseteq \mathbb{R}^{\mathbb{N}}$. Let (E, d) be a complete and separable metric space and $\mu \in \mathcal{M}_f(E)$. Then, the triple (E, d, μ) is called a *metric measure triple* (also write *mm-triple*).

ii - Let (E, d, μ) and (E', d', μ') be metric measure triples. Then (E', d', μ') is said to be *isomorphic* to (E, d, μ) if there exists an isometry $i : supp(\mu) \longrightarrow supp(\mu')$ such that the image measure of μ with respect to i is equal to μ':

$$\mu \circ i^{-1} = \mu'.$$

We then write:

$$(E, d, \mu) \sim_i (E', d', \mu').$$

Definition 2.2.3 (Metric Measure Space)

i - Let (E, d, μ) be a metric measure triple. We denote the isomorphism class of (E, d, μ) by $[E, d, \mu]$. Moreover, we define:

$$\mathbb{M} \overset{def}{=} \{\mathfrak{e} \mid \exists (E, d, \mu) \text{ a mm-triple such that } [E, d, \mu] = \mathfrak{e}\}, \tag{2.1}$$

and the elements of $\mathbb{M}$ are called *metric measure spaces* (or *mm-spaces*).

ii - We denote the null-measure by $\mathbf{0}_{\mathcal{M}}$ and the null space in $\mathbb{M}$ by:

$$\mathbf{0}_{\mathbb{M}} \overset{def}{=} [\{1\}, d_1, \mathbf{0}_{\mathcal{M}}], \tag{2.2}$$

where d_1 is the trivial metric on $\{1\}$.

iii - We also set:

$$\mathbb{M}_{>0} \overset{def}{=} \{\mathfrak{e} = [E, d, \mu] \in \mathbb{M} \mid \mu(E) > 0\}, \tag{2.3}$$

and

$$\mathbb{M}_1 \overset{def}{=} \{\mathfrak{e} = [E, d, \mu] \in \mathbb{M} \mid \mu(E) = 1\}. \tag{2.4}$$

If $\mu(E) = 1$, we call (E, d, μ) a *metric probability measure triple* (also write *mp-triple*) and $[E, d, \mu]$ a *metric probability measure space* (or *mp-space*).

iv - Finally, if $\#supp(\mu) < \infty$, we call $\mathfrak{e} = [E, d, \mu] \in \mathbb{M}$ a *finite metric measure space* and we define:

$$\#\mathfrak{e} \overset{def}{=} \#supp(\mu),$$

where we have used the notation $\#S$ to denote the cardinality of the set S.

Remark 2.2.4

If (E, d, μ) is a metric measure **triple**, we assume that $E = supp(\mu)$ (that is, no Borel subset of E with non-empty interior has μ-measure 0).

To study convergence of metric measure spaces, we will use the Gromov-Prokhorov metric. This metric includes the idea of the Gromov-Hausdorff distance (to take into account the metric aspect) as well as the idea of the Prokhorov metric (to take into account the measure aspect). Let us recall a few useful definitions.

Definition 2.2.5 (Gromov-Hausdorff Distance)

i - Let E_1 and E_2 be two subsets of a metric space (E, d). The *Hausdorff Distance* between E_1 and E_2 is defined as:

$$d_H(E_1, E_2) \overset{def}{=} \inf\{\varepsilon > 0 \mid E_1 \subseteq V_\varepsilon(E_2) \text{ and } E_2 \subseteq V_\varepsilon(E_1)\}, \tag{2.5}$$

where

$$V_\varepsilon(B) \overset{def}{=} \left\{ e \in E \mid \inf_{e' \in B} d(e, e') < \varepsilon \right\}.$$

ii - Let (E, d) and (E', d') be two metric spaces. The *Gromov-Hausdorff Distance* between these two spaces is defined as:

$$d_{GH}(E, E') \overset{def}{=} \inf\left\{ d_H\left(\varphi_E(E), \varphi_{E'}(E')\right) \mid \varphi_E : E \hookrightarrow Z \text{ and } \varphi_{E'} : E' \hookrightarrow Z \text{ are isometries} \right\}, \tag{2.6}$$

where the infimum is taken over all isometric embeddings φ_E and $\varphi_{E'}$ of E and E' into some common metric space (Z, d_Z) and d_H is the Hausdorff metric.

Intuitively, we will represent a tree as a metric space (E, d), where the space E is the set of leaves and the distance, d, corresponds to some genealogical distance between the leaves. Therefore, to obtain the Gromov-Hausdorff distance between two trees, we transform the two trees to find the best way to make them coincide and we take the Hausdorff distance between the two transformations. However, we also need to put a mass on each leaf. To take the measure aspect into account while keeping the idea of the Gromov-Hausdorff distance, we use the Gromov-Prokhorov distance.

Definition 2.2.6 (Gromov-Prokhorov Distance)

i - Let (E, d) be a metric space with its Borel-σ-algebra $\mathcal{B}(E)$. For μ and μ', two probability measures on the measurable space $(E, \mathcal{B}(E))$, the *Prokhorov distance* is defined as:

$$d_P(\mu, \mu') \overset{def}{=} \inf\left\{ \varepsilon > 0 \mid \mu(B) \leq \mu'(V_\varepsilon(B)) + \varepsilon \text{ and } \mu'(B) \leq \mu(V_\varepsilon(B)) + \varepsilon \text{ for any closed set } B \right\}, \tag{2.7}$$

where $V_\varepsilon(B) = \left\{ e \in E \mid \inf_{e' \in B} d(e, e') < \varepsilon \right\}$ is the ε- neighbourhood of B defined earlier.

ii - For $\mathfrak{e} = [E, d, \mu] \in \mathbb{M}_1$ and $\mathfrak{e}' = [E', d', \mu'] \in \mathbb{M}_1$, the *Gromov-Prokhorov distance* is defined as:

$$d_{GP}(\mathfrak{e}, \mathfrak{e}') \overset{def}{=} \inf\left\{ d_P\left(\mu \circ \varphi_E^{-1}, \mu \circ \varphi_{E'}^{-1}\right) \mid \varphi_E : E \hookrightarrow Z \text{ and } \varphi_{E'} : E' \hookrightarrow Z \text{ are isometries} \right\}, \tag{2.8}$$

where the infimum is taken over all isometric embeddings φ_E and $\varphi_{E'}$ of E and E' into some common metric space (Z, d_Z) and d_P is the Prokhorov metric.

Now that we have set our framework, we can build our process.

3 - Construction of the Tree-Valued Process

To build our tree-valued process, we need to define three processes:

1. the total mass process, which represents the evolution of the number of extremities (or leaves) in the tree over time,

2. the lexicographic process, which allows to number the extremities,

3. the metric-valued process, which provides a distance over the set of indices of extremities.

We describe these three processes in the next three sections. In what follows, we shall see the extremities of the branches of the polymer as individuals in a population, and we will use "extremities" and "individuals" interchangeably.

3.1 - The Total Mass Process

3.1.1 - Basic branching system

Let $\boldsymbol{p^N} = (p_k^N)_{k\in\mathbb{N}}$ be the probability distribution on $\mathbb{N}$ such that:

$$p_k^N = \begin{cases} \frac{1}{2} - \frac{a}{N} & \text{if } k = 0, \\ \frac{1}{2} + \frac{a}{N} & \text{if } k = 2, \\ 0 & \text{otherwise.} \end{cases}$$

We use this distribution in order to determine whether a given "binding" event is a branching event due to Arp2/3 or a fragmentation event due to cofilin.

Let $\Xi_1^N = \{\xi_n^N, \theta_n^N, v_n^N \mid n \in \mathbb{N}\}$ be a family of independent random variables on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ such that:

$$\xi_0^N \in \{1, 2, \cdots, N\} \text{ a.s. and } \theta_0^N \sim \text{Exp}(\beta),$$

and, for all $n \in \mathbb{N}^*$,

$$\xi_n^N + 1 \sim \boldsymbol{p^N}, \quad \theta_n^N \sim \text{Exp}(\beta) \text{ and } v_n^N \sim \text{Unif}([0,1]).$$

The variable ξ_0^N is the initial number of individuals (i.e. the initial number of extremities in the branched polymer). $(\xi_n^N + 1)$ determines the number of offspring at the n-th binding. In other words, the variable $\xi_n^N \in \{-1, 1\}$ is the increment of the total number of individual during event n. The sequence $(\theta_n^N)_{n\in\mathbb{N}}$ will be used to determine the sequence of times of binding. Finally, we will use the variable v_n^N to choose on which extremity the n-th binding will occur. In the following, we will call *branch i*, the branch which carries the extremity i, that is the edge between the leaf i and its parent.

3.1.2 - Total number of extremities

We let X_n^N denote the total number of extremities (individuals) after the n-th binding. The sequence $(X_n^N)_{n\in\mathbb{N}}$ is easily defined by induction by:

$$\begin{cases} X_0^N = \xi_0^N, \\ X_n^N = X_{n-1}^N + \xi_n^N & \text{if } X_{n-1}^N > 0, \\ X_n^N = 0 & \text{if } X_{n-1}^N = 0. \end{cases}$$

So, by writing $n_0 = \min\{n \in \mathbb{N} \mid X_n^N = 0\}$, we have for every $n \in \mathbb{N}$:

$$X_n^N = \begin{cases} \sum_{i=0}^n \xi_i^N & \text{if } n < n_0, \\ 0 & \text{if } n \geq n_0. \end{cases}$$

Let $(\Theta_n^N)_{n\in\mathbb{N}}$ be our notation for the sequence of times of binding. As we will later scale down the total number of extremities by the factor N, we assume that the number of extremities evolves at a speed proportional to N. Therefore, each extremity of the branched polymer is subject to a binding event at the rate $N\beta$. Without this assumption, the rescaled process for the total number of extremities would not allow us to observe an non-trivial behaviour in the limit as $N \longrightarrow \infty$.

The total rate of binding after the n-th event is $\beta N X_n^N$ and $\Theta_{n+1}^N - \Theta_n^N \sim \mathrm{Exp}(\beta N X_n^N)$. By property of the exponential distribution, we have, conditionally on X_n^N,

$$\tilde{\theta}_n^N = \frac{\theta_n^N}{NX_n^N} \sim \mathrm{Exp}(\beta N X_n^N).$$

We then define the sequence $(\Theta_n^N)_{n\in\mathbb{N}}$ by induction by:

$$\begin{cases} \Theta_0^N = 0, \\ \Theta_n^N = \Theta_{n-1}^N + \tilde{\theta}_{n-1}^N. \end{cases}$$

So, we have for every $n \in \mathbb{N}$:

$$\Theta_n^N = \sum_{i=0}^{n-1} \tilde{\theta}_i^N.$$

We can now define the continuous process which represents the evolution of the scaled number of extremities of the polymer $\tilde{\boldsymbol{X}}^{\boldsymbol{N}} = (\tilde{X}_t^N)_{t\in\mathbb{R}_+}$ by, for every $t \in \mathbb{R}_+$:

$$\tilde{X}_t^N \overset{def}{=} \frac{X_n^N}{N} \text{ when } t \in \left[\Theta_n^N, \Theta_{n+1}^N\right[. \tag{2.9}$$

Since 0 is an absorbing state for $\tilde{\boldsymbol{X}}^{\boldsymbol{N}}$, we also define $\tau_{ex}^N \overset{def}{=} \inf\{t \geq 0 \mid \tilde{X}_t^N = 0\}$ the time of extinction of the population which corresponds to the disappearance of the polymer.

3.2 - The Lexicographic Process

3.2.1 - Tree structure

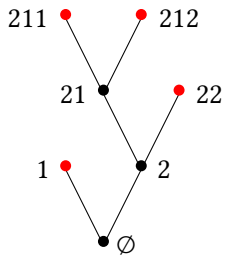
The binding of a protein leads either to the creation or to the disappearance of a branch. This then leads to a renumbering of extremities. We therefore need to retain the genealogical structure represented here by a plane binary tree.

Consider the set of labels:

$$\mathbb{I} = \bigcup_{n=0}^{\infty} \mathbb{N}^n,$$

with convention $\mathbb{N}^0 = \emptyset$. An element ι of $\mathbb{I}$ is a sequence $\iota = <\iota_1, \cdots, \iota_n>$ of elements of $\mathbb{N}$ where $|\iota| = n$ is the length of ι and represents the "generation" of ι. A plane binary tree is therefore a subset of $\mathbb{I}$.

Example 2.3.1
This plane binary tree,



is represented by the set:

$$\mathcal{T} = \{\emptyset, <1>, <2>, <2,1>, <2,2>, <2,1,1>, <2,1,2>\} \subset \mathbb{I},$$

and the set of alive individuals (or extremities, in our application) is:

$$\mathcal{I} = \{<1>, <2,2>, <2,1,1>, <2,1,2>\} \subset \mathbb{I}.$$

For $\iota, \iota' \in \mathbb{I}$, we define:

$$s(\iota, \iota') = \max\{k \in \mathbb{N} \mid \forall j \in \mathbb{N},\ j \leq k,\ \iota_j = \iota'_j\}.$$

For example, $s(<2,1,1>,<2,2>) = 1$ and we can notice that $s(\iota, \iota')$ represents the generation at which the "lineages" of ι and ι' diverge.

The intersection of $\iota, \iota' \in \mathbb{I}$ is defined by:

$$\iota \cap \iota' = \begin{cases} <\iota_1, \iota_2, \cdots, \iota_{s(\iota,\iota')}> & \text{if } s(\iota, \iota') > 0, \\ \emptyset & \text{if } s(\iota, \iota') = 0, \end{cases}$$

and the concatenation of $\iota = <\iota_1, \cdots, \iota_n> \in \mathbb{I}$ and $\iota' = <\iota'_1, \cdots, \iota'_m> \in \mathbb{I}$ is given by:

$$c(\iota, \iota') = <\iota_1, \cdots, \iota_n, \iota'_1, \cdots, \iota'_m>.$$

For example, $<2,1,1> \cap <2,2> = <2>$ and $c(<2,1,1>,<2,2>) = <2,1,1,2,2>$.

Finally, we will need an order on $\mathbb{I}$ which is defined by:

$$\iota \lhd \iota' \iff \iota_{s(\iota,\iota')+1} < \iota'_{s(\iota,\iota')+1}.$$

For instance, $<2,1,1> \lhd <2,2>$ and visually, $\iota \lhd \iota'$ means that the node ι is to the left of node ι'.

3.2.2 - Indexing the leaves

As for the process $\tilde{\boldsymbol{X}}^{\boldsymbol{N}}$, we start by constructing a discrete-time process $(\mathcal{I}_n^N)_{n\in\mathbb{N}}$ by induction.

- For $n = 0$, let
$$\mathcal{I}_0^N = \{\iota^{(1)}, \iota^{(2)}, \cdots, \iota^{(X_0^N)}\} = \{<1>, <2>, \cdots, <X_0^N>\}.$$
- For $n \in \mathbb{N}$, at the n-th binding of a protein, we proceed as follows:

- If $X_{n-1}^N = 0$, then $\mathcal{I}_{n-1}^N = \emptyset$ and $\mathcal{I}_n^N = \emptyset$.
- If $X_{n-1}^N > 0$, then, after ordering the individuals according to $\lhd$, we can write

$$\mathcal{I}_{n-1}^N = \{\iota^{(1)}, \iota^{(2)}, \cdots, \iota^{(X_{n-1}^N)}\}$$

where $\iota^{(1)} \lhd \iota^{(2)} \lhd \cdots \lhd \iota^{(X_{n-1}^N)}$.

We now need a random variable to chose on which extremity the n-th binding will occurs. Let $\tilde{v}_n^N = \lceil v_n^N \cdot X_{n-1}^N \rceil$ where v_n^N was defined in section 3.1.1. As $v_n^N \sim \text{Unif}([0,1])$, $\tilde{v}_n^N$ is uniformly distributed on the set $\{1, 2, \cdots, X_{n-1}^N\}$ and the next binding will occur on the branch $\tilde{v}_n^N$.

$\mathcal{I}_n^N$ is then defined as follows:

  - If $\xi_n = -1$, it is a cofilin that binds to the polymer and a fragmentation occurs. So, the individual $\iota^{\tilde{v}_n}$ is removed from the population without producing offspring and

$$\mathcal{I}_n^N = \mathcal{I}_{n-1}^N \backslash \{\iota^{(\tilde{v}_n)}\}.$$

  - If $\xi_n = 1$, it is an Arp2/3 that binds to the polymer and a branching occurs. So, the individual $\iota^{\tilde{v}_n}$ is replaced by two offspring and

$$\mathcal{I}_n^N = \mathcal{I}_{n-1}^N \backslash \{\iota^{(\tilde{v}_n)}\} \cup \left\{ c(\iota^{(\tilde{v}_n)}, <1>), c(\iota^{(\tilde{v}_n)}, <2>) \right\}.$$

We can now define the continuous lexicographic process $\tilde{\mathcal{I}}^N = (\tilde{\mathcal{I}}_t^N)_{t \in \mathbb{R}_+}$ by, for every $t \in \mathbb{R}_+$:

$$\tilde{\mathcal{I}}_t^N \overset{def}{=} \mathcal{I}_n^N \text{ when } t \in \left[\Theta_n^N, \Theta_{n+1}^N\right[. \tag{2.10}$$

We can also define ancestors at time s of individuals from $\tilde{\mathcal{I}}_t^N$.

Definition 2.3.2 (Ancestor)
Let $0 \leq s \leq t < \tau_{ex}^N$. An individual $\iota \in \tilde{\mathcal{I}}_s^N$ is an *ancestor* of $\iota' \in \tilde{\mathcal{I}}_t^N$ if and only if $\iota \cap \iota' = \iota$. In this case, we write:

$$\tilde{\mathcal{A}}_s^N(\iota', t) = \iota.$$

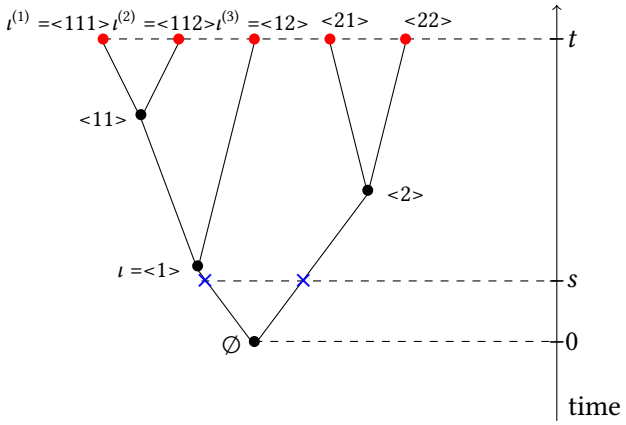
Moreover, the *number of ancestors* at time s of individuals from $\tilde{\mathcal{I}}_t^N$ is denoted by:

$$\#\tilde{\mathcal{A}}_s^N(t) = \#\left\{ \tilde{\mathcal{A}}_s^N(\iota', t) \mid \iota' \in \tilde{\mathcal{I}}_t^N \right\}.$$

Example 2.3.3
If at time t we have:

- $N\tilde{X}_t^N = 5$,
- $\tilde{\mathcal{I}}_t^N = \{<1,1,1>, <1,1,2>, <1,2>, <2,1>, <2,2>\}$.

Then we have the following binary tree:



and:

- $\tilde{\mathcal{A}}_s^N(\iota^{(1)}, t) = \tilde{\mathcal{A}}_s^N(\iota^{(2)}, t) = \tilde{\mathcal{A}}_s^N(\iota^{(3)}, t) = \iota$,
- $\#\tilde{\mathcal{A}}_s^N(t) = 2$.

3.3 - The Metric-Valued Process

3.3.1 - The Pseudo-metric-valued process

Now that the tree structure is built, we can define a pseudo-metric on it. The pseudo-distance between two individuals of $\tilde{\boldsymbol{\mathcal{I}}}^{\boldsymbol{N}}$ corresponds to the genealogical distance expressed in number of monomers. This is a pseudo-distance since, immediately after a branching event, the two new extremities are not distinguishable. We must first define the process of elongation of alive branches. As before, we start by constructing a discrete-time process.

Let $\Xi_2^N = \{\varphi_n^N, v_n^N \mid n \in \mathbb{N}\}$ be a family of independent random variables on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ such that, for all $n \in \mathbb{N}$,

$$\varphi_n^N \sim \text{Exp}(\lambda^+) \text{ and } v_n^N \sim \text{Unif}([0,1]).$$

The sequence $(\varphi_n^N)_{n\in\mathbb{N}}$ will be used to determine the sequence of times of elongation and we will use the variable v_n^N to choose on which extremity the n-th elongation will occur.

We denote by $m \in \mathbb{N}$, the number of bindings of protein that occurred when the n-th elongation takes place. Therefore, there are X_m^N extremities and the structure of the tree is given by $\mathcal{I}_m^N = \{\iota^{(1)}, \iota^{(2)}, \cdots, \iota^{(X_m^N)}\}$ with $\iota^{(1)} \lhd \iota^{(2)} \lhd \cdots \lhd \iota^{(X_m^N)}$ just before the n-th elongation. We will use this notation and this order for $\mathcal{I}_m^N$ throughout this section. We let $\rho_{m,n}^N$ denote the pseudo-metric on $\mathcal{I}_m^N$ at the n-th elongation such that $\rho_{m,n}^N(\iota^{(i)}, \iota^{(j)})$ is the number of monomers between extremities $\iota^{(i)}$ and $\iota^{(j)}$ along the tree.

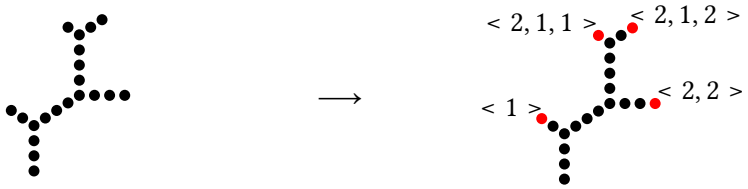
Remark 2.3.4

In an equivalent way, we can define $\mathcal{D}_{m,n}^N = (d_{i,j}^N(m,n))_{1\le i,j\le X_m^N} \in \mathcal{S}^{X_m^N}$ the pseudo-distance matrix after m bindings of protein and n-th elongation such that:

$$d_{i,j}^N(m,n) = \rho_{m,n}^N(\iota^{(i)}, \iota^{(j)}).$$

Example 2.3.5

If the branched actin polymer is as follows:



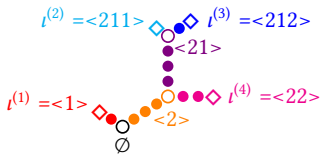
there are $X = 4$ extremities, the set of extremities is:

$$\mathcal{I} = \{< 1 >, < 2, 1, 1 >, < 2, 1, 2 >, < 2, 2 >\} = \{\iota^{(1)}, \iota^{(2)}, \iota^{(3)}, \iota^{(4)}\},$$

and the pseudo-distance matrix is given by:

$$\mathcal{D} = \begin{pmatrix} 0 & 12 & 13 & 10 \\ 12 & 0 & 4 & 9 \\ 13 & 4 & 0 & 10 \\ 10 & 9 & 10 & 0 \end{pmatrix}$$

We can notice that the root branch is not taken into account. As we have chosen a model with a lot of short branches ($\mathcal{O}(N)$ branches of length $\mathcal{O}(1)$), this will have no impact on the mean behaviour. With one color per branch, the polymer must be represented like this:



Let $\rho^N_{0,0}$ be a pseudo-metric on the space $\mathcal{I}^N_0$. We define the process $(\rho^N_{m,n})_{m,n\in\mathbb{N}}$ by induction on $m \in \mathbb{N}$, the number of bindings of protein and on $n \in \mathbb{N}$, the number of elongation.

- At the $(n+1)$-th elongation, by assuming that m bindings of protein have occurred, we define the pseudo-metric $\rho^N_{m,n+1}$ as follows:
  - Let $\tilde{v}^N_{n+1} = \lceil \nu^N_{n+1} \cdot X^N_m \rceil$. The next elongation will occur on the extremity $\tilde{v}^N_{n+1}$.
  - When the extremity $\tilde{v}^N_{n+1}$ grows, the branch $\tilde{v}^N_{n+1}$ gains a monomer and thus the distance between this extremity and any other increases by 1 (while the others distances remains unchanged). If we consider this transformation on the pseudo-distance matrix, this is equivalent to adding 1 to the $\tilde{v}^N_{n+1}$-th row and to the $\tilde{v}^N_{n+1}$-th column except to the element $(\tilde{v}^N_{n+1}, \tilde{v}^N_{n+1})$ which stays null. The pseudo-distance matrix after the $(n+1)$-th elongation, $\mathcal{D}^N_{m,n+1}$, is then defined by:

$$\mathcal{D}^N_{m,n+1} = \mathcal{D}^N_{m,n} + \sum_{i=1}^{X^N_m} \mathbf{J}_{i,\tilde{v}_{n+1}} + \sum_{j=1}^{X^N_m} \mathbf{J}_{\tilde{v}_{n+1},j} - 2\mathbf{J}_{\tilde{v}_{n+1},\tilde{v}_{n+1}},$$

    where $\mathbf{J}_{i,j}$ is the single-entry matrix with a one on the i-th row and j-th column and zeros everywhere else.

    In an equivalent way, we can define $\rho^N_{m,n+1}$, the pseudo-distance associated with the matrix $\mathcal{D}^N_{m,n+1}$, for $\iota^{(i)}, \iota^{(j)} \in \mathcal{I}^N_m$ by:

$$\rho^N_{m,n+1}(\iota^{(i)},\iota^{(j)}) = \begin{cases} \rho^N_{m,n}(\iota^{(i)},\iota^{(j)}) + 1 & \text{if } (i = \tilde{v}_{n+1} \text{ or } j = \tilde{v}_{n+1}) \text{ and } i \neq j \\ \rho^N_{m,n}(\iota^{(i)},\iota^{(j)}) & \text{otherwise.} \end{cases}$$

- At the $(m+1)$-th binding, by assuming that n elongations have occurred, we define the pseudo-metric $\rho^N_{m+1,n}$ as follows:
  - As described in Section 3.2.2, the next binding will occur on the branch $\tilde{v}_{m+1}$.
  - If $\xi_{m+1} = -1$, it is a fragmentation and the individual $\iota^{(\tilde{v}_{m+1})}$ is removed from the population. We then obtain the pseudo-distance matrix $\mathcal{D}^N_{m+1,n}$ by removing the $\tilde{v}_{m+1}$-th row and the $\tilde{v}_{m+1}$-th column of the matrix $\mathcal{D}^N_{m,n}$.
    The pseudo-distance $\rho^N_{m+1,n}$ is defined for $\iota^{(i)},\iota^{(j)} \in \mathcal{I}^N_{m+1}$ with $1 \leq i \leq j \leq X^N_m - 1$ by:

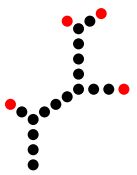
$$\rho^N_{m+1,n}(\iota^{(i)},\iota^{(j)}) = \begin{cases} \rho^N_{m,n}(\iota^{(i)},\iota^{(j)}) & \text{if } i < j < \tilde{v}_{m+1}, \\ \rho^N_{m,n}(\iota^{(i)},\iota^{(j+1)}) & \text{if } i < \tilde{v}_{m+1},\ j \geq \tilde{v}_{m+1} \text{ and } i < j, \\ \rho^N_{m,n}(\iota^{(i+1)},\iota^{(j+1)}) & \text{if } \tilde{v}_{m+1} \leq i < j, \\ 0 & \text{if } i = j. \end{cases}$$

  - If $\xi_{m+1} = 1$, it is a branching and the individual $\iota^{(\tilde{v}_{m+1})}$ is removed from the population and replaced by $c\left(\iota^{(\tilde{v}_{m+1})}, <1>\right)$ and $c\left(\iota^{(\tilde{v}_{m+1})}, <2>\right)$. In this case, we obtain the pseudo-distance matrix $\mathcal{D}^N_{m+1,n}$ from $\mathcal{D}^N_{m,n}$ by duplicating the $\tilde{v}_{m+1}$-th row and the $\tilde{v}_{m+1}$-th column and by setting the element $(\tilde{v}_{m+1},\tilde{v}_{m+1}+1)$, and the element $(\tilde{v}_{m+1}+1,\tilde{v}_{m+1})$ to zero. The pseudo-distance $\rho^N_{m+1,n}$ is then defined for $\iota^{(i)},\iota^{(j)} \in \mathcal{I}^N_{m+1}$ with $1 \leq i \leq j \leq X^N_m - 1$ by:

$$\rho^N_{m+1,n}(\iota^{(i)},\iota^{(j)}) = \begin{cases} \rho^N_{m,n}(\iota^{(i)},\iota^{(j)}) & \text{if } i < j \leq \tilde{v}_{m+1}, \\ \rho^N_{m,n}(\iota^{(i)},\iota^{(j-1)}) & \text{if } i < \tilde{v}_{m+1},\ j > \tilde{v}_{m+1} \text{ and } i < j, \\ 0 & \text{if } i = \tilde{v}_{m+1} \text{ and } j = \tilde{v}_{m+1}+1, \\ \rho^N_{m,n}(\iota^{(i)},\iota^{(j-1)}) & \text{if } i = \tilde{v}_{m+1} \text{ and } j > \tilde{v}_{m+1}+1, \\ \rho^N_{m,n}(\iota^{(i-1)},\iota^{(j-1)}) & \text{if } \tilde{v}_{m+1} < i < j, \\ 0 & \text{if } i = j. \end{cases}$$

Example 2.3.6

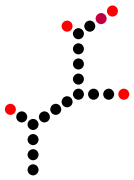
We return to Example 2.3.5. Our branched actin polymer is as follows:



and the pseudo-distance matrix is given by:

$$\mathcal{D} = \begin{pmatrix} 0 & 12 & 13 & 10 \\ 12 & 0 & 4 & 9 \\ 13 & 4 & 0 & 10 \\ 10 & 9 & 10 & 0 \end{pmatrix}$$

- If an elongation occurs on the third extremity, we have:

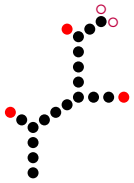


$$\mathcal{D}' = \begin{pmatrix} 0 & 12 & 14 & 10 \\ 12 & 0 & 5 & 9 \\ 14 & 5 & 0 & 11 \\ 10 & 9 & 11 & 0 \end{pmatrix}$$

- If a fragmentation occurs on the third extremity, we have:

$$\mathcal{D}' = \begin{pmatrix} 0 & 12 & 10 \\ 12 & 0 & 9 \\ 10 & 9 & 0 \end{pmatrix}$$

- If a branching event occurs on the third extremity, we have:



$$\mathcal{D}' = \begin{pmatrix} 0 & 12 & 13 & 13 & 10 \\ 12 & 0 & 4 & 4 & 9 \\ 13 & 4 & 0 & 0 & 10 \\ 13 & 4 & 0 & 0 & 10 \\ 10 & 9 & 10 & 10 & 0 \end{pmatrix}$$

We can now proceed in the same way as for $\tilde{\boldsymbol{X}}^{\boldsymbol{N}}$. Let $(\Phi_n^N)_{n\in\mathbb{N}}$ be our notation for the sequence of times of elongation and,

$$\tilde{\varphi}_n^N = \frac{\varphi_n^N}{NX_m^N} \sim \mathrm{Exp}(\lambda^+ NX_m^N) \text{ conditionnally on } X_m^N.$$

We have for every $n \in \mathbb{N}$:

$$\Phi_n^N = \sum_{i=0}^{n-1} \tilde{\varphi}_i^N,$$

and we can now define the continuous process which represents the evolution of the pseudo-metric $\tilde{\boldsymbol{\rho}}^{\boldsymbol{N}} = (\tilde{\rho}_t^N)_{t\in\mathbb{R}_+}$ by, for every $t \in \mathbb{R}_+$:

$$\tilde{\rho}_t^N \overset{def}{=} \frac{\rho_{m,n}^N}{N} \text{ when } t \in \left[\Theta_m^N, \Theta_{m+1}^N\right[\cap \left[\Phi_n^N, \Phi_{n+1}^N\right[. \tag{2.11}$$

3.3.2 - Metric on equivalent classes

In order to have a metric instead of a pseudo-metric, we need to consider two undistinguishable points as a single point. For this, we pass to the quotient space with respect to $\tilde{\rho}_t^N$. More precisely, for $t \in [0, \tau_{ex}^N[$ and $\iota \in \tilde{\mathcal{I}}_t^N$, we define the equivalent class of ι by:

$$[\iota]_0 = \left\{ \iota' \in \tilde{\mathcal{I}}_t^N \mid \tilde{\rho}_t^N(\iota, \iota') = 0 \right\},$$

and we define the quotient space process $\mathfrak{I}^{\boldsymbol{N}} = (\mathfrak{I}_t^N)_{t\in\mathbb{R}_+}$ by, for all $t \in \mathbb{R}_+$:

$$\mathfrak{I}_t^N \overset{def}{=} \tilde{\mathcal{I}}_t^N / \tilde{\rho}_t^N = \left\{ \mathfrak{i} = [\iota]_0 \mid \iota \in \tilde{\mathcal{I}}_t^N \right\}. \tag{2.12}$$

Hence, the lexicographic order on $\mathfrak{I}_t^N$ is that of the smallest representatives of each equivalent class. For every $t \in \mathbb{R}_+$, we can now define a metric $\mathfrak{r}_t^N$ on $\mathfrak{I}_t^N$ by:

$$\mathfrak{r}_t^N([\iota]_0, [\iota']_0) \overset{def}{=} \tilde{\rho}_t^N(\iota, \iota') \quad \forall \iota, \iota' \in \tilde{I}_t^N, \tag{2.13}$$

and finally, we write $\mathfrak{r}^N = (\mathfrak{r}_t^N)_{t\in\mathbb{R}_+}$ for the metric-valued process.

Remark 2.3.7

In the following, we will need the notion of ancestors in the quotient space $\mathfrak{I}_t^N$, for every $t \in \mathbb{R}_+$. We adapt definition 2.3.2 as follows:

> Let $0 \le s \le t < \infty$. An individual $\mathfrak{i} \in \mathfrak{I}_s^N$ is an *ancestor* of $\mathfrak{i}' \in \mathfrak{I}_t^N$ if and only if there are $\iota \in \mathfrak{i}$ and $\iota' \in \mathfrak{i}'$ such that $\iota \cap \iota' = \iota$. In this case, we write:
>
> $$\mathfrak{A}_s^N(\mathfrak{i}', t) = \mathfrak{i}. \tag{2.14}$$
>
> Moreover, the *number of ancestors* at time s of individuals from $\mathfrak{I}_t^N$ is denoted by:
>
> $$\#\mathfrak{A}_s^N(t) = \#\left\{\mathfrak{A}_s^N(\mathfrak{i}', t) \mid \mathfrak{i}' \in \mathfrak{I}_t^N\right\}. \tag{2.15}$$

Note that by an abuse of terminology, we refer to the elements of $\mathfrak{I}_t^N$ as "individuals" rather than as "classes of individuals".

Also note that for $t \ge \tau_{ex}^N$, $\mathfrak{I}_t^N = \emptyset$. Hence, there is no $\mathfrak{i}' \in \mathfrak{I}_t^N$ and so, no ancestors either. In particular, for all $t \ge \tau_{ex}^N$, $\#\mathfrak{A}_s^N(t) = 0$.

3.4 - The Tree-Valued Process

By construction, for every $t \in [0, \tau_{ex}^N[$, $\left(\mathfrak{I}_t^N, \mathfrak{r}_t^N\right)$ is a metric space. We now want to add a measure on $\mathfrak{I}_t^N$ in order to obtain a metric measure triple as defined in Definition 2.2.2.(i).

We simply construct a finite measure that give the same mass to each extremity of the polymer. As we want a non-trivial asymptotic behaviour when $N \longrightarrow +\infty$, we need to give a mass $\frac{1}{N}$ to each individual in order to control the total mass. Such a measure on the extremities (and thus on the set $\tilde{I}_t^N$) is written $\frac{1}{N}\sum_{\iota\in\tilde{I}_t^N}\delta_\iota$. As we want a measure on classes of individuals (i.e. on the set $\mathfrak{I}_t^N$), we need to multiply the Dirac mass of a class of individuals $\mathfrak{i} \in \mathfrak{I}_t^N$ by the number of individuals in this class. Thus, we define the measure $\mathfrak{u}_t^N$ on $\mathfrak{I}_t^N$ by, for all $t \in [0, \tau_{ex}^N[$:

$$\mathfrak{u}_t^N \overset{def}{=} \frac{1}{N}\sum_{\mathfrak{i}\in\mathfrak{I}_t^N} \#\mathfrak{i} \cdot \delta_{\mathfrak{i}}, \tag{2.16}$$

where $\#\mathfrak{i}$ is the cardinal of the equivalence class $\mathfrak{i}$, and we write $\mathfrak{u}^N = (\mathfrak{u}_t^N)_{t\in[0,\tau_{ex}^N[}$ the measure-valued process.

At last, for all $t \in \mathbb{R}_+$, $\left(\mathfrak{I}_t^N, \mathfrak{r}_t^N, \mathfrak{u}_t^N\right)$ is a metric measure triple and by passing to the isomorphism class of $\left(\mathfrak{I}_t^N, \mathfrak{r}_t^N, \mathfrak{u}_t^N\right)$ (according to definition 2.2.2.(ii)), we have constructed a process which take values in $\mathbb{M}$.

The tree-valued branching process at scale N derived from (Ξ_1, Ξ_2, N) is denoted by $\mathfrak{T}^N = (\mathfrak{T}_t^N)_{t\in\mathbb{R}_+}$ and defined by, for every $t \in [0, \tau_{ex}^N[$:

$$\mathfrak{T}_t^N = \left[\mathfrak{I}_t^N, \mathfrak{r}_t^N, \mathfrak{u}_t^N\right], \tag{2.17}$$

where $\mathfrak{I}_t^N$, $\mathfrak{r}_t^N$ and $\mathfrak{u}_t^N$ are respectively defined in (2.12), (2.13) and (2.16).

In order to define $\mathfrak{T}^N$ over $\mathbb{R}_+$ as a whole, we need to find the right metric space in which to study our tree-valued process. In particular, we need to find a null state for $\mathfrak{T}^N$ after the extinction time, and have a metric space suitable for studying the convergence when N goes to infinity. This is the point of the next section.

For the entire definition of the process $\mathfrak{T}^N$, see Definition 2.4.6 page 86.

4 - State-Space of the Tree-Valued Process

As seen in Section 2, the metric space $(\mathbb{M}, d_{GP})$ is the appropriate framework to define our tree-valued process. However, this metric space is not convenient to study the convergence of the process as N goes to infinity. In this section, we introduce some definitions, notation and the main results needed to find the right state-space for $\mathfrak{T}^N$ to study this convergence.

Firstly, we will see that metric probability spaces are adequate spaces to study a tree-valued process. However, the measure associated with a tree-valued process is not necessarily a probability measure. In the second subsection, we use the polar decomposition (decomposition of a mm-space as a positive constant and a mp-space) to extend the results of the first subsection. Finally, we introduce finite metric measure spaces to study discrete tree-valued processes.

4.1 - Metric Probability Measure Spaces

4.1.1 - Distance matrix distributions

The first important result is a theorem due to Gromov [Gro99] that ensures that a metric probability measure triple (E, d, μ) may be characterised through the distribution of the distances between points which have been randomly sampled according to μ. In order to formulate this Gromov's result, we need first to define the distance matrix map.

Definition 2.4.1 (Distance Matrix Map)

i - For all $n \in \mathbb{N}$ and $n \geq 2$, let

$$\mathbb{U}^n \overset{def}{=} \{\boldsymbol{u^n} = (u^n_{i,j})_{1\leq i<j\leq n} \mid u^n_{i,j} \in \mathbb{R}_+, u^n_{i,j} \leq u^n_{i,k} + u^n_{k,j}, \forall 1 \leq i < j \leq n\}.$$

Intuitively, $\mathbb{U}^n$ is the set of (upper-triangular) distance matrices of size n.

ii - Let (E, d, μ) be a mp-triple and $n \in \mathbb{N}$, $n \geq 2$. We define the *distance matrix map* of degree n by:

$$\begin{array}{rccc} U^n_{(E,d)} : & E^n & \to & \mathbb{U}^n \\ & (e_1, e_2, \cdots, e_n) & \mapsto & \left\{ d(e_i, e_j) \right\}_{1\leq i<j\leq n} \end{array}$$

Therefore, $U^n_{(E,d)}$ maps a random sample of points of E of size n to its distance matrix.

Theorem 2.4.2 (see [Gro99])
Let $[E, d, \mu], [E', d'\mu'] \in \mathbb{M}_1$. Then

$$[E, d, \mu] = [E', d', \mu'] \Longleftrightarrow \forall n \in \mathbb{N},\ n \geq 2,\ \mu^{\otimes n} \circ \left(U^n_{(E,d)}\right)^{-1} = \mu'^{\otimes n} \circ \left(U^n_{(E',d')}\right)^{-1}.$$

In other words, an element $\mathfrak{e} = [E, d, \mu] \in \mathbb{M}_1$ is uniquely determined by the distributions of distance matrices of random samples.

It is now clear that distance matrix distributions have a key role in the study of mp-spaces, and so, we will need some more notation.

For $\mathfrak{e} = [E, d, \mu] \in \mathbb{M}_1$ and $n \in \mathbb{N}$, $n \geq 2$, we note:

$$\nu^n_{\mathfrak{e}} \overset{def}{=} \mu^{\otimes n} \circ \left(U^n_{(E,d)}\right)^{-1}. \tag{2.18}$$

the distance matrix distribution for a random sample of size n.

We have seen that a mp-space $\mathfrak{e} \in \mathbb{M}_1$ is uniquely determined by its distance matrix distributions. By the separation property of bounded continuous functions on $\mathcal{M}_1(\mathbb{U}^n)$, distance matrix distributions are uniquely determined by the first moment of bounded continuous functions. In order to use this property, we need to introduce polynomials.

4.1.2 - Polynomials on $\mathbb{M}_1$

Recall that, for a function $f : S_1 \longrightarrow S_2$ and for a measure μ , we note:

$$\langle f, \mu \rangle = \int_{S_1} f(s) d\mu(s),$$

the integral of f with respect to μ.

For $n \in \mathbb{N}$, $n \geq 2$, and for a real-valued bounded continuous function $\omega \in \mathcal{C}_b(\mathbb{U}^n)$, we define $\Omega^n_\omega : \mathbb{M}_1 \longrightarrow \mathbb{R}$, by:

$$\begin{aligned}
\Omega^n_\omega(\mathfrak{e}) &\overset{def}{=} \langle \omega, \nu^n_{\mathfrak{e}} \rangle, \\
&= \int_{\mathbb{U}^n} \omega(\boldsymbol{u^n}) \mathrm{d}\mu^{\otimes n} \left(\left(U^n_{(E,d)}\right)^{-1} (\boldsymbol{u^n}) \right), \\
&= \int_{E^n} \omega \left(\left\{ d(e_i, e_j) \right\}_{1 \leq i < j \leq n} \right) \mathrm{d}\mu^{\otimes n}(e_1, e_2, \cdots, e_n),
\end{aligned} \tag{2.19}$$

where $(E, d, \mu) \in \mathfrak{e}$.

We refer to functions of this form as Ω-polynomials of degree n and we denote the set of polynomials by:

$$\Pi_\Omega \overset{def}{=} \{\Omega^n_\omega \mid n \in \mathbb{N}, n \geq 2 \text{ and } \omega \in \mathcal{C}_b(\mathbb{U}^n)\}.$$

If we restrict ω to some subclass $C(\mathbb{U}^n) \subset \mathcal{C}_b(\mathbb{U}^n)$, we indicate this by writing:

$$\Pi_\Omega(C) \overset{def}{=} \{\Omega^n_\omega \in \Pi_\Omega \mid \omega \in C(\mathbb{U}^n)\}.$$

We also let ${}^S\Pi_\Omega(C)$ denote the linear span of $\Pi_\Omega(C)$ and ${}^{\mathcal{A}}\Pi_\Omega(C)$ the algebra generated by $\Pi_\Omega(C)$.

The next proposition, which corresponds to Proposition 2.3.4 of [Glö13], ensures that an element $\mathfrak{e} = [E, d, \mu] \in \mathbb{M}_1$ is uniquely determined by Ω-polynomials restricted to class $C(\mathbb{U}^n)$, if this class $C(\mathbb{U}^n) \subseteq \mathcal{C}_b(\mathbb{U}^n)$ is separating for $\mathcal{M}_1(\mathbb{U}^n)$.

Proposition 2.4.3 (see [Glö13])
Assume $C(\mathbb{U}^n) \subseteq \mathcal{C}_b(\mathbb{U}^n)$ is separating for $\mathcal{M}_1(\mathbb{U}^n)$. Then, $\Pi_\Omega(C)$ separates points on $\mathbb{M}_1$.

Since the measure-valued process $\mathfrak{u}^N$ of our tree-valued process $\mathfrak{T}^N$ is not a probability measure-valued process, the space $\mathbb{M}_1$ can not be a state-space for $\mathfrak{T}^N$, and we must generalise the previous result.

4.2 - Metric Measure Spaces

The main idea of this section is that we can transform each metric measure space with a positive total mass into a metric probability measure space by normalising its measure. Conversely, we can transform each mp-space into a mm-space with a positive total mass by multiplying the probability measure by a positive constant. This shows that every mm-space with positive mass can be uniquely represented by a mp-space (containing the information about the distribution of distances) and a positive constant (representing the mass of the space). We refer to this representation as *polar decomposition*.

4.2.1 - Polar decomposition

Definition 2.4.4 (Polar Decomposition Map)
For $\mathfrak{e} = [E, d, \mu] \in \mathbb{M}_{>0}$, we define the *total mass* by:

$$m_{\mathfrak{e}} \overset{def}{=} \mu(E),$$

the *normalised measure* by:

$$\hat{\mu} \overset{def}{=} \frac{\mu}{m_{\mathfrak{e}}} \tag{2.20}$$

and the *normalised metric measure space* by:

$$\hat{\mathfrak{e}} \overset{def}{=} [E, d, \hat{\mu}] \in \mathbb{M}_1.$$

Finally, we define the *polar decomposition map* by:

$$\begin{array}{rccc} \pi : & \mathbb{M}_{>0} & \longrightarrow & \mathbb{R}_+^* \times \mathbb{M}_1 \\ & \mathfrak{e} & \longmapsto & (m_{\mathfrak{e}}, \hat{\mathfrak{e}}) \end{array}$$

Thanks to this decomposition, we can easily find a distance on $\mathbb{M}_{>0}$. Indeed, considering the Euclidean metric, d_{eucl}, on $\mathbb{R}_+^*$ and the Gromov-Prokhorov metric, d_{GP}, on $\mathbb{M}_1$, we can define a metric on the product space $\mathbb{R}_+^* \times \mathbb{M}_1$ as follows: for $m, m' \in \mathbb{R}_+^*$ and for $\mathfrak{e}, \mathfrak{e}' \in \mathbb{M}_1$,

$$d_{eucl} \otimes d_{GP}\left((m, \mathfrak{e}), (m', \mathfrak{e}')\right) \overset{def}{=} |m - m'| + d_{GP}(\mathfrak{e}, \mathfrak{e}').$$

Moreover, since π is a bijection (see Lemma 2.4.1 of [Glö13]), we can define a distance, $d_{GP}^{\mathbb{M}_{>0}}$, on $\mathbb{M}_{>0}$ by: for $\mathfrak{e}, \mathfrak{e}' \in \mathbb{M}_{>0}$,

$$d_{GP}^{\mathbb{M}_{>0}}(\mathfrak{e}, \mathfrak{e}') \overset{def}{=} d_{eucl} \otimes d_{GP}\left(\pi(\mathfrak{e}), \pi(\mathfrak{e}')\right) = |m_{\mathfrak{e}} - m_{\mathfrak{e}'}| + d_{GP}(\hat{\mathfrak{e}}, \hat{\mathfrak{e}}').$$

However, the metric space $(\mathbb{M}_{>0}, d_{GP}^{\mathbb{M}_{>0}})$ does not contain the limits of sequences of $\mathbb{M}_{>0}$ with a total mass which tends to 0. To take into account mm-spaces with null total mass, we construct a space $\mathbb{M}_0$ which contains all the elements of $\mathbb{M}_1$ (i.e. with a total mass equal to 1) but associated with the constant 0 (instead of 1). To be more precise, we have seen previously that an element $\mathfrak{e}_1 \in \mathbb{M}_1$ is associated with the pair $(1, \mathfrak{e}_1)$. Thus, the elements of $\mathbb{M}_0$ are the elements associated with the pair $(0, \mathfrak{e}_1)$ for each $\mathfrak{e}_1 \in \mathbb{M}_1$.

We then consider the space $\mathbb{T}$ define by:

$$\mathbb{T} \overset{def}{=} \mathbb{M}_{>0} \sqcup \mathbb{M}_0,$$

and we can adapt the distance $d_{GP}^{\mathbb{M}_{>0}}$ to have a distance, $d_{GP}^{\mathbb{T}}$, on $\mathbb{T}$ by writing, for $\mathfrak{e}, \mathfrak{e}' \in \mathbb{T}$:

$$d_{GP}^{\mathbb{T}}(\mathfrak{e}, \mathfrak{e}') \overset{def}{=} \begin{cases} d_{GP}^{\mathbb{M}_{>0}}(\mathfrak{e}, \mathfrak{e}') & \text{if } \mathfrak{e}, \mathfrak{e}' \in \mathbb{M}_{>0}, \\ m_{\mathfrak{e}} + d_{GP}(\hat{\mathfrak{e}}, \mathfrak{e}') & \text{if } \mathfrak{e} \in \mathbb{M}_{>0} \text{ and } \mathfrak{e}' \in \mathbb{M}_0, \\ d_{GP}(\mathfrak{e}, \mathfrak{e}') & \text{if } \mathfrak{e}, \mathfrak{e}' \in \mathbb{M}_0. \end{cases}$$

Finally, we extend the bijection π by setting:

$$\pi(\mathfrak{e}) \overset{def}{=} (0, \mathfrak{e}) \text{ if } \mathfrak{e} \in \mathbb{M}_0$$

and π becomes an isometry between $\left(\mathbb{T}, d_{GP}^{\mathbb{T}}\right)$ and $(\mathbb{R}_+ \times \mathbb{M}_1, d_{eucl} \otimes d_{GP})$.

This lead to the next proposition, which corresponds to Proposition 2.4.3 in [Glö13], and which finishes to convince us that $\left(\mathbb{T}, d_{GP}^{\mathbb{T}}\right)$ is the right state-space for our tree-valued process $\mathfrak{T}^{\boldsymbol{N}}$.

Proposition 2.4.5 (see [Glö13])
The metric space $\left(\mathbb{T}, d_{GP}^{\mathbb{T}}\right)$ is Polish.

Now that we have found the right space for $\mathfrak{T}^{\boldsymbol{N}}$, we can define the null state and give a complete definition of our tree-valued branching process.

Definition 2.4.6 (Tree-Valued Branching Process)

i - We define the *special null-space* by:

$$\mathfrak{n} \overset{def}{=} \pi^{-1}\big((0, [\{1\}, d_1, \delta_1]) \big) \in \mathbb{T} \tag{2.21}$$

where d_1 is the trivial metric on $\{1\}$ and π is defined in Definition 2.4.4.

ii - The *tree-valued branching process* at scale N derived from (Ξ_1, Ξ_2, N) is denoted by $\mathfrak{T}^{\boldsymbol{N}} = (\mathfrak{T}_t^N)_{t \in \mathbb{R}_+}$ and defined by:

$$\mathfrak{T}_t^N \overset{def}{=} \begin{cases} \left[\mathfrak{I}_t^N, \mathfrak{r}_t^N, \mathfrak{u}_t^N\right] & \text{if } t \in [0, \tau_{ex}^N[, \\ \mathfrak{n} & \text{if } t \in [\tau_{ex}^N, +\infty[, \end{cases} \tag{2.22}$$

where $\mathfrak{I}_t^N$, $\mathfrak{r}_t^N$ and $\mathfrak{u}_t^N$ are respectively defined in (2.12), (2.13) and (2.16), and the special null-space is defined in (2.21).

iii - The process $\mathfrak{m}^{\boldsymbol{N}} = \left(\mathfrak{m}_t^N\right)_{t \in \mathbb{R}_+}$ defined by:

$$\mathfrak{m}_t^N \overset{def}{=} \begin{cases} \mathfrak{u}_t^N(\mathfrak{I}_t^N) & \text{if } t \in [0, \tau_{ex}^N[, \\ 0 & \text{if } t \in [\tau_{ex}^N, +\infty[, \end{cases} \tag{2.23}$$

is called the *total mass process* of $\mathfrak{T}^{\boldsymbol{N}}$.

iv - The normalised process

$$\hat{\mathfrak{T}}^{\boldsymbol{N}} \overset{def}{=} \left(\hat{\mathfrak{T}}_t^N\right)_{t\in\mathbb{R}_+} \tag{2.24}$$

is called the *genealogical process* of $\mathfrak{T}^{\boldsymbol{N}}$.

Remark 2.4.7

1. By construction, $\mathfrak{T}_t^N \in \mathbb{T}$ and $\hat{\mathfrak{T}}_t^N \in \mathbb{M}_1$ for all $t \in \mathbb{R}_+$.
2. By definition of the processes $\tilde{X}^{\boldsymbol{N}}$, $\mathfrak{I}^{\boldsymbol{N}}$ and $\mathfrak{u}^{\boldsymbol{N}}$, we have, for all $t \in \mathbb{R}_+$,

$$\mathfrak{m}_t^N = \tilde{X}_t^N$$

In the same way as before, we will now define classes of functions on $\mathbb{T}$ that separates points on $\mathbb{T}$.

4.2.2 - Polynomials on $\mathbb{T}$

For $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_b(\mathbb{R}_+)$, we define $\Psi^n_{\omega,\psi} : \mathbb{T} \longrightarrow \mathbb{R}$, by:

$$\begin{aligned}\Psi^n_{\omega,\psi}(\mathfrak{e}) &\overset{def}{=} \psi(m_{\mathfrak{e}})\Omega^n_\omega(\hat{\mathfrak{e}}),\\ &= \psi(m_{\mathfrak{e}}) \int_{E^n} \omega\left(\left\{d(e_i,e_j)\right\}_{1\leq i<j\leq n}\right) \mathrm{d}\hat{\mu}^{\otimes n}(e_1,e_2,\cdots,e_n),\end{aligned} \tag{2.25}$$

where $(E, d, \hat{\mu}) \in \hat{\mathfrak{e}}$.

We refer to functions of this forms as Ψ-polynomials of degree n and we denote the set of polynomials by:

$$\Pi_\Psi \overset{def}{=} \left\{\Psi^n_{\omega,\psi} \mid n \in \mathbb{N}, n \geq 2, \omega \in \mathcal{C}_b(\mathbb{U}^n) \text{ and } \psi \in \mathcal{C}_b(\mathbb{R}_+)\right\}.$$

If we restrict ω to some subclass $C_1(\mathbb{U}^n) \subset \mathcal{C}_b(\mathbb{U}^n)$ and ψ to some subclass $C_2(\mathbb{R}_+) \subset \mathcal{C}_b(\mathbb{R}_+)$, we indicate this by writing:

$$\Pi_\Psi(C_1, C_2) \overset{def}{=} \left\{\Psi^n_{\omega,\psi} \in \Pi_\Psi \mid \omega \in C_1(\mathbb{U}^n) \text{ and } \psi \in C_2(\mathbb{R}_+)\right\}.$$

We also let ${}^S\Pi_\Psi(C_1, C_2)$ denote the linear span of $\Pi_\Psi(C_1, C_2)$ and ${}^{\mathcal{A}}\Pi_\Psi(C_1, C_2)$ the algebra generated by $\Pi_\Psi(C_1, C_2)$.

The next proposition, which corresponds to Proposition 2.4.9 (i) of [Glö13], gives a sufficient condition on classes $C_1(\mathbb{U}^n)$ and $C_2(\mathbb{R}_+)$ for $\Pi_\Psi(C_1, C_2)$ to separates points on $\mathbb{T}$. We denote by $\mathbb{1}_S$ the indicator function of a set S.

Proposition 2.4.8 (see [Glö13])
Let $C_1(\mathbb{U}^n) \subseteq \mathcal{C}_b(\mathbb{U}^n)$ and $C_2(\mathbb{R}_+) \subseteq \mathcal{C}_b(\mathbb{R}_+)$. Assume that $C_1(\mathbb{U}^n)$ is separating for $\mathcal{M}_1(\mathbb{U}^n)$ and $C_2(\mathbb{R}_+)$ separates points on $\mathbb{R}_+$. Moreover, assume that $\mathbb{1}_{\mathbb{U}^n} \in C_1(\mathbb{U}^n)$ **or** $\mathbb{1}_{\mathbb{R}_+} \in C_2(\mathbb{R}_+)$. Then, $\Pi_\Psi(C_1, C_2)$ separates points on $\mathbb{T}$.

We now have all the notation and results necessary to study a tree-valued process, but only when the number of individuals is infinite. Indeed, the results of Sections 4.1 and 4.2 are based on the sampling of n elements of E according to $\mu^{\otimes n}$. When the support of μ is finite, we shall modify the sampling procedure to sample without replacement. This is the subject of the next section.

4.3 - Finite Metric Measure Space

For $N \in \mathbb{N}$, we define:

$$\mathbb{M}_{>0}^{N} \overset{def}{=} \{[E,d,\mu] \in \mathbb{M}_{>0} \mid \#supp(\mu) \in \mathbb{N} \text{ and } \forall e \in E,\ N\mu(\{e\}) \in \mathbb{N}\}$$

and

$$\mathbb{T}^{N} \overset{def}{=} \mathbb{M}_{>0}^{N} \cup \{\mathfrak{n}\},$$

where $\mathfrak{n}$ is the special null space defined in (2.21).

Finite metric measure spaces have simple representation. If $[E,d,\mu] \in \mathbb{M}_{>0}^{N}$, then by definition $supp(\mu)$ is a finite set and we can assume that $E = \{1,2,\cdots,\#supp(\mu)\}$. The sampling measure can then be written:

$$\mu = \sum_{e\in E} \mu(\{e\})\delta_e.$$

This representation amounts to thinking of $[E,d,\mu]$ as an urn containing $Nm_{\mathfrak{e}}$ balls of $\#E$ colors.

As in the previous case, distance matrix distributions have a key role in the study of finite metric measure spaces. The difference in this finite case is that we have to make sure that each $e \in E$ is sampled at most $N\mu(\{e\})$ times. Using the urn analogy, this corresponds to a sampling without replacement of balls.

Let $\mathfrak{e} = [E,d,\mu] \in \mathbb{T}^N$ and $n \in \mathbb{N}$. We define the n–fold measure without replacement on E^n by:

$$d\mu^{\otimes n,N}(e_1,e_2,\cdots,e_n) \overset{def}{=} \begin{cases} \bigotimes_{i=1}^{n}\left(\mu - \frac{1}{N}\sum_{j=1}^{i-1}\delta_{e_j}\right)(de_1,de_2,\cdots,de_n) & \text{if } m_{\mathfrak{e}} > 0, \\ \mathbf{0}_{\mathcal{M}} & \text{if } m_{\mathfrak{e}} = 0. \end{cases} \tag{2.26}$$

For $x \in \mathbb{R}$ and $n,N \in \mathbb{N}$, let

$$(x)^{n,N} \overset{def}{=} x\left(x-\frac{1}{N}\right)\left(x-\frac{2}{N}\right)\cdots\left(x-\frac{n-1}{N}\right). \tag{2.27}$$

Then the normalised n–fold measure without replacement on E^n is given by:

$$\widehat{d\mu^{\otimes n,N}}(e_1,e_2,\cdots,e_n) \overset{def}{=} \begin{cases} \frac{d\mu^{\otimes n,N}(e_1,e_2,\cdots,e_n)}{(m_{\mathfrak{e}})^{n,N}} & \text{if } m_{\mathfrak{e}} > 0 \text{ and } n \le Nm_{\mathfrak{e}} \\ \mathbf{0}_{\mathcal{M}} & \text{otherwise.} \end{cases} \tag{2.28}$$

For $\mathfrak{e} = [E,d,\mu] \in \mathbb{T}^N$ and $n \in \mathbb{N}$, $n \ge 2$, we define the distance matrix distribution without replacement of degree n by:

$$\nu_{\mathfrak{e}}^{n,N} \overset{def}{=} \widehat{\mu^{\otimes n,N}} \circ \left(U_{(E,d)}^{n}\right)^{-1}$$

Remark 2.4.9
We can note that $\widehat{\mu^{\otimes n,N}} \in \mathcal{M}_1(E^n) \cup \{\mathbf{0}_{\mathcal{M}}\}$ and $\nu_{\mathfrak{e}}^{n,N} \in \mathcal{M}_1(\mathbb{U}^n) \cup \{\mathbf{0}_{\mathcal{M}}\}$. Moreover, $\mu^{\otimes n,N} = \widehat{\mu^{\otimes n,N}} = \mathbf{0}_{\mathcal{M}}$ if $n > Nm_{\mathfrak{e}}$.

All that remains is to introduce polynomials for finite mm-spaces, which will once again form classes of function that separates points, but this time on $\mathbb{T}^N$.

For $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_b(\mathbb{R}_+)$, we define $\Omega_\omega^{n,N} : \mathbb{T}^N \longrightarrow \mathbb{R}$ by:

$$\begin{aligned}\Omega_\omega^{n,N}(\mathfrak{e}) &\stackrel{def}{=} \left\langle \omega, \nu_{\mathfrak{e}}^{n,N} \right\rangle, \\ &= \int_{E^n} \omega\left(\left\{ d(e_i, e_j)\right\}_{1\leq i<j\leq n}\right) \widehat{\mathrm{d}\mu^{\otimes n,N}}(e_1, e_2, \cdots, e_n),\end{aligned} \tag{2.29}$$

and $\Psi_{\omega,\psi}^{n,N} : \mathbb{T}^N \longrightarrow \mathbb{R}$ by:

$$\begin{aligned}\Psi_{\omega,\psi}^{n,N}(\mathfrak{e}) &\stackrel{def}{=} \psi(m_{\mathfrak{e}})\Omega_\omega^{n,N}(\mathfrak{e}), \\ &= \psi(m_{\mathfrak{e}}) \int_{E^n} \omega\left(\left\{ d(e_i, e_j)\right\}_{1\leq i<j\leq n}\right) \widehat{\mathrm{d}\mu^{\otimes n,N}}(e_1, e_2, \cdots, e_n).\end{aligned} \tag{2.30}$$

where $(E, d, \mu) \in \mathfrak{e}$.

We refer to functions of this forms as Ω–polynomials without replacement and Ψ–polynomials without replacement, respectively. We denote these sets by:

$$\Pi_\Omega^N \stackrel{def}{=} \left\{\Omega_\omega^{n,N} \mid n \in \mathbb{N}, n \geq 2 \text{ and } \omega \in \mathcal{C}_b(\mathbb{U}^n)\right\} \cup \left\{\mathbb{1}_{\mathbb{T}^N}\right\},$$

and

$$\Pi_\Psi^N \stackrel{def}{=} \left\{\Psi_{\omega,\psi}^{n,N} \mid \psi \in \mathcal{C}_b(\mathbb{R}_+) \text{ and } \Omega_\omega^{n,N} \in \Pi_\Omega^N\right\}.$$

If we restrict ω to some subclass $C_1(\mathbb{U}^n) \subset \mathcal{C}_b(\mathbb{U}^n)$ and ψ to some subclass $C_2(\mathbb{R}_+) \subset \mathcal{C}_b(\mathbb{R}_+)$, we indicate this by writing:

$$\Pi_\Omega^N(C_1) \stackrel{def}{=} \left\{\Omega_\omega^{n,N} \mid n \in \mathbb{N}, n \geq 2 \text{ and } \omega \in C_1(\mathbb{U}^n)\right\} \cup \left\{\mathbb{1}_{\mathbb{T}^N}\right\},$$

and

$$\Pi_\Psi^N(C_1, C_2) \stackrel{def}{=} \left\{\Psi_{\omega,\psi}^{n,N} \mid \psi \in C_2(\mathbb{R}_+) \text{ and } \Omega_\omega^{n,N} \in \Pi_\Omega^N(C_1)\right\}.$$

We also let ${}^S\Pi_\Omega^N(C_1)$ denote (respectively ${}^S\Pi_\Psi^N(C_1, C_2)$) the linear span of $\Pi_\Omega^N(C_1)$ (resp. of $\Pi_\Psi^N(C_1, C_2)$) and ${}^{\mathcal{A}}\Pi_\Omega^N(C_1)$ (resp. ${}^{\mathcal{A}}\Pi_\Psi^N(C_1, C_2)$) the algebra generated by $\Pi_\Omega^N(C_1)$ (resp. by $\Pi_\Psi^N(C_1, C_2)$).

Remark 2.4.10
The function $\mathbb{1}_{\mathbb{T}^N}$ is added to the set of Ω–polynomials without replacement to ensure that Π_Ψ^N separates $\mathfrak{n}$ from $\pi^{-1}\left(\left(\frac{1}{N}, [1, d_1, \delta_1]\right)\right)$. This makes it possible to distinguish a space whose total mass tends to 0 when N goes to infinity from the special null space $\mathfrak{n}$.

We finish this section with a statement, which correspond to Proposition 2.5.12 of [Glö13], about subclasses of polynomials which separates points on $\mathbb{T}^N$.

Proposition 2.4.11 (see [Glö13])
Let $C_1(\mathbb{U}^n) \subseteq \mathcal{C}_b(\mathbb{U}^n)$ and $C_2(\mathbb{R}_+) \subseteq \mathcal{C}_b(\mathbb{R}_+)$. Assume that C_2 satisfies either of the following assumptions:

i - C_2 separates points on $\frac{1}{N}\mathbb{N}$.

ii - For each $K \in \mathbb{N}$, there exists $\psi \in C_2$ such that $\psi(x) = 1$ for all $x \in [0, K]$. Moreover, for each $x \in \frac{1}{N}\mathbb{N}$, there exists $\psi \in C_2$ such that $\psi(0) \neq \psi(x)$.

Assume that C_1 is separating for $\mathcal{M}_1(\mathbb{U}^n)$. Then, $\Pi_\Psi^N(C_1, C_2)$ separates points on $\mathbb{T}^n$.

5 - Generators of the Discrete Tree-Valued Process

In order to study the convergence of the tree-valued process $\mathfrak{T}^N$ as N goes to infinity, we first need to establish the convergence of its infinitesimal generator. This generator corresponds to the sum of the "growth" generator and the "binding" generator. As explained in Section 4.3, Ψ-polynomials without replacement are the right class of functions to study the discrete tree-valued process. However, it is necessary to restrict this class of functions in order to be able to study the convergence of the growth and binding generators. We write $\mathcal{C}_b^{1bc}$ for the set of continuous bounded functions with first partial derivatives bounded and continuous and $\mathcal{C}_K^\infty$ for the set of smooth functions with compact support. For these functions, the generator of $\mathfrak{T}^N$ can be written as follows:

Definition 2.5.1 (Generator of $\mathfrak{T}^N$)
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b^{1bc}(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_K^\infty(\mathbb{R}_+)$. The *infinitesimal generator of the discrete tree-valued process* applied to the Ψ-polynomial without replacement $\Psi_{\omega,\psi}^{n,N}$ is given by:

$$\mathcal{G}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e}) = \mathcal{G}_{grow}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e}) + \mathcal{G}_{bind}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e}),$$

where $\mathcal{G}_{grow}^N$ and $\mathcal{G}_{bind}^N$ are defined respectively in (2.34) and in (2.40).

In this whole section, when we choose a mm-space $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, we also implicitly choose a mm-triple $(E, d, \mu) \in [E, d, \mu]$ which is a representative of the isomorphism class $[E, d, \mu]$. We assume that each point of this representative $x \in (E, d, \mu)$ (also written $x \in E$) has a mass $\mu(\{x\}) = \frac{1}{N}$, duplicating some of the points to account for multiplicities if necessary. As mentioned in Remark 2.2.4, we recall that the space E of the representative (E, d, μ) satisfies $E = supp(\mu)$.

Let us now define the growth generator and the binding generator.

5.1 - Growth Generator

Since the genealogical distance between extremities has been scaled down by the factor N in (2.11), we assume that each extremity (more precisely each branch attached to an extremity) grows at a speed proportional to N. Thus, each extremity of the branched polymer can grow at rate $N\lambda^+$.

Since the state-space of our discrete tree-valued process is $\mathbb{T}^N$, we need to define the growth rate of an element of $\mathbb{T}^N$. For $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, the total mass is denoted by $m_\mathfrak{e}$ and so, there are $Nm_\mathfrak{e}$ extremities that can be elongated. Therefore, the total rate of elongation is $N^2\lambda^+ m_\mathfrak{e}$.

When an elongation occurs, the extremity that will grow is chosen uniformly at random from the $Nm_\mathfrak{e}$ extremities and the distance between the chosen extremity and all the others increases by $\frac{1}{N}$. So, for $x, e, e' \in (E, d, \mu)$, where (E, d, μ) is the representative of the class $[E, d, \mu]$, if there is an elongation of the branch x, the distance between e and e' becomes:

$$\Delta_x^N(e, e') \overset{def}{=} \begin{cases} d(e, e') + \frac{1}{N} & \text{if } e \neq e' \text{ and } (e = x \text{ or } e' = x), \\ d(e, e') & \text{otherwise.} \end{cases} \tag{2.31}$$

and the element $[E, d, \mu] \in \mathbb{T}^N$ becomes $[E, \Delta_x^N, \mu]$. Moreover, the jump kernel for the growth process, denoted by Q_{grow}^N, is defined for $\mathfrak{e} \in \mathbb{T}^N$ as follows:

$$Q_{grow}^N(\mathfrak{e}, d\mathfrak{v}) \overset{def}{=} \frac{1}{Nm_\mathfrak{e}} \sum_{x \in E} \delta_{[E, \Delta_x^N, \mu]}(d\mathfrak{v}). \tag{2.32}$$

The term $\frac{1}{N m_{\mathfrak{e}}}$ corresponds to the uniform choice of the extremity and $\delta_{[E,\Delta_x^N,\mu]}$ gives a mass 1 to the new metric measure space $[E, \Delta_x^N, \mu]$ created by the elongation (and a mass 0 to all other spaces of $\mathbb{T}^N$).

We now have all the elements to write the infinitesimal generator $\mathcal{G}_{grow}^N$ for the elongation. For a suitable function f and for $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, we have:

$$\begin{aligned}\mathcal{G}_{grow}^N f(\mathfrak{e}) &\overset{def}{=} N^2 \lambda^+ m_{\mathfrak{e}} \int_{\mathbb{T}^N} (f(\mathfrak{v}) - f(\mathfrak{e}))\, Q_{grow}^N(\mathfrak{e}, d\mathfrak{v}),\\ &= N\lambda^+ \sum_{x \in E} \left(f\left([E, \Delta_x^N, \mu]\right) - f(\mathfrak{e}) \right). \end{aligned} \tag{2.33}$$

By applying this expression to a Ψ–polynomial without replacement, we have finally:

Definition 2.5.2 (Growth Generator)
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b^{1bc}(\mathbb{U}^n)$ and $\psi \in C_b(\mathbb{R}_+)$. The *growth generator* applied to the Ψ-polynomial without replacement $\Psi_{\omega,\psi}^{n,N}$ is given by:

$$\mathcal{G}_{grow}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e}) = \sum_{k=1}^{n} N\lambda^+ \psi(m_{\mathfrak{e}}) \int_{E^n} \left(\omega\left(\left(\Delta_{e_k}^N(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) - \omega\left(\left(d(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) \right) \widehat{d\mu^{\otimes n,N}}(e_1, \cdots, e_n), \tag{2.34}$$

where Δ^N is defined in (2.31).

Remark 2.5.3
The sum over $x \in E$ in expression (2.33) becomes a sum over $x \in E^n$ in expression (2.34) since if x is not in E^n, the elongation of the branch x does not change the distance matrix $\left(d(e_i, e_j) \right)_{1 \leq i < j \leq n}$.

Since $\omega \in C_b^{1bc}(\mathbb{U}^n)$, we can use a first order Taylor expansion of $\omega\left(\left(\Delta_{e_k}^N(e_i, e_j) \right)_{1 \leq i < j \leq n} \right)$ to obtain an easier expression for the growth generator defined in (2.34).

Lemma 2.5.4
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b^{1bc}(\mathbb{U}^n)$ and $\psi \in C_b(\mathbb{R}_+)$. Then

$$\mathcal{G}_{grow}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e}) = \Psi_{2\lambda^+ \overline{\nabla}\omega,\psi}^{n,N}(\mathfrak{e}) + \mathcal{O}\left(\frac{1}{N}\right), \tag{2.35}$$

where:

$$\overline{\nabla}\omega = \sum_{1 \leq i < j \leq n} \frac{\partial \omega}{\partial d(e_i, e_j)}. \tag{2.36}$$

Proof of Lemma 2.5.4. A first order Taylor expansion of $\omega\left(\left(\Delta_{e_k}^N(e_i, e_j) \right)_{1 \leq i < j \leq n} \right)$ gives us:

$$\omega\left(\left(\Delta_{e_k}^N(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) = \omega\left(\left(d(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) + \frac{1}{N} \sum_{\substack{1 \leq i < j \leq n \\ i=k \text{ or } j=k}} \frac{\partial \omega}{\partial d(e_i, e_j)} \left(\left(d(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) + \mathcal{O}\left(\frac{1}{N^2}\right).$$

Using this expression in (2.34), we obtain:

$$\begin{aligned}
\mathcal{G}^N_{grow}\Psi^{n,N}_{\omega,\psi}(\mathfrak{e}) &= \sum_{k=1}^{n} N\lambda^+\psi(m_{\mathfrak{e}}) \int_{E^n} \left(\frac{1}{N} \sum_{\substack{1\le i<j\le n \\ i=k \text{ or } j=k}} \frac{\partial\omega}{\partial d(e_i,e_j)} \left(\left(d(e_i,e_j)\right)_{1\le i<j\le n} \right) + \mathcal{O}\left(\frac{1}{N^2}\right) \right) \widehat{d\mu^{\otimes n,N}}(e_1,\cdots,e_n), \\
&= N\lambda^+\psi(m_{\mathfrak{e}}) \int_{E^n} \left(\frac{1}{N} \sum_{k=1}^{n} \sum_{i=1}^{k-1} \frac{\partial\omega}{\partial d(e_i,e_k)} \left(\left(d(e_i,e_j)\right)_{1\le i<j\le n} \right) \right. \\
&\qquad \left. + \frac{1}{N} \sum_{k=1}^{n} \sum_{j=k+1}^{n} \frac{\partial\omega}{\partial d(e_k,e_j)} \left(\left(d(e_i,e_j)\right)_{1\le i<j\le n} \right) + \mathcal{O}\left(\frac{1}{N^2}\right) \right) \widehat{d\mu^{\otimes n,N}}(e_1,\cdots,e_n), \\
&= N\lambda^+\psi(m_{\mathfrak{e}}) \int_{E^n} \left(\frac{2}{N} \sum_{1\le i<k\le n} \frac{\partial\omega}{\partial d(e_i,e_k)} \left(\left(d(e_i,e_j)\right)_{1\le i<j\le n} \right) + \mathcal{O}\left(\frac{1}{N^2}\right) \right) \widehat{d\mu^{\otimes n,N}}(e_1,\cdots,e_n), \\
&= \psi(m_{\mathfrak{e}}) \int_{E^n} 2\lambda^+\overline{\nabla}\omega \left(\left(d(e_i,e_j)\right)_{1\le i<j\le n} \right) \widehat{d\mu^{\otimes n,N}}(e_1,\cdots,e_n) + \mathcal{O}\left(\frac{1}{N}\right).
\end{aligned}$$

We conclude with expression (2.30) which defines Ψ-polynomials without replacement.

Remark 2.5.5
As ψ is bounded and $\widehat{\mu^{\otimes n,N}}$ is a probability measure, we have:

$$\left|\lambda^+\psi(m_{\mathfrak{e}})\widehat{\mu^{\otimes n,N}}(E^n)\right| \le \lambda^+||\psi||_\infty,$$

where $||\psi||_\infty = \sup_{x\in\mathbb{R}_+} |\psi(x)|$ and

$$\lambda^+\psi(m_{\mathfrak{e}})\widehat{\mu^{\otimes n,N}}(E^n)\mathcal{O}\left(\frac{1}{N}\right) = \mathcal{O}\left(\frac{1}{N}\right).$$

5.2 - Binding Generator

As explained in Section 3.1.2, we assume that the individual rate of binding is $N\beta$. So, as for the rate of elongation, the total rate of binding is $N^2\beta m_{\mathfrak{e}}$ for $\mathfrak{e} = [E,d,\mu] \in \mathbb{T}^N$.

When a binding event occurs, the branch to which the protein will bind is chosen uniformly at random from the $Nm_{\mathfrak{e}}$ extremities. Then, the type of protein is chosen using the distribution $\boldsymbol{p^N}$. It is therefore a fragmentation with probability p_0^N and a branching with probability p_2^N.

When a fragmentation occurs on the branch $x \in (E,d,\mu)$, the measure μ loses a mass $\frac{1}{N}$ in x and the mm-space $\mathfrak{e} = [E,d,\mu]$ becomes:

$$F^N_x(\mathfrak{e}) \overset{def}{=} \begin{cases} \left[E,d,\mu-\frac{1}{N}\delta_x\right] & \text{if } m_{\mathfrak{e}} > \frac{1}{N}, \\ \mathfrak{n} & \text{otherwise.} \end{cases} \tag{2.37}$$

Note that the metric space (E,d) is implicitly modified since the mass in x is now null and when dealing with representative of an mm-space, we always assume that $E = supp(\mu)$.

When a branching occurs on the extremity $x \in (E,d,\mu)$, the measure μ gains a mass $\frac{1}{N}$ in x and the mm-space $\mathfrak{e} = [E,d,\mu]$ becomes:

$$B^N_x(\mathfrak{e}) \overset{def}{=} \left[E,d,\mu+\frac{1}{N}\delta_x\right]. \tag{2.38}$$

Once again, the representative (E, d, μ) is implicitly modified and the point x is duplicated in E.

The jump kernel for the binding process, denoted by Q^N_{bind}, is defined for $\mathfrak{e} \in \mathbb{T}^N$ as follows:

$$Q^N_{bind}(\mathfrak{e}, d\mathfrak{v}) \overset{def}{=} \frac{1}{N m_{\mathfrak{e}}} \sum_{x \in E} \left(p_0^N \delta_{F_x^N(\mathfrak{e})}(d\mathfrak{v}) + p_2^N \delta_{B_x^N(\mathfrak{e})}(d\mathfrak{v}) \right), \tag{2.39}$$

where $\frac{1}{N m_{\mathfrak{e}}}$ corresponds to the uniform choice of the extremity, $\delta_{F_x^N(\mathfrak{e})}$ gives a mass 1 to the space $F_x^N(\mathfrak{e})$ created by fragmentation with probability p_0^N and $\delta_{B_x^N(\mathfrak{e})}$ gives a mass 1 to the space $B_x^N(\mathfrak{e})$ created by branching with probability p_2^N.

We can now define the infinitesimal generator $\mathcal{G}^N_{bind}$ for the binding. For a suitable function f and for $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, we have:

$$\begin{aligned} \mathcal{G}^N_{bind} f(\mathfrak{e}) &\overset{def}{=} N^2 \beta m_{\mathfrak{e}} \int_{\mathbb{T}^N} (f(\mathfrak{v}) - f(\mathfrak{e}))\, Q^N_{bind}(\mathfrak{e}, d\mathfrak{v}), \\ &= N\beta p_0^N \sum_{x \in E} \left(f(F_x^N(\mathfrak{e})) - f(\mathfrak{e}) \right) + N\beta p_2^N \sum_{x \in E} \left(f(B_x^N(\mathfrak{e})) - f(\mathfrak{e}) \right). \end{aligned}$$

By applying this expression to a Ψ–polynomial without replacement, we have finally:

Definition 2.5.6 (Binding Operator)
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_K^\infty(\mathbb{R}_+)$. The *binding generator* applied to the Ψ-polynomial without replacement $\Psi^{n,N}_{\omega,\psi}$ is given by:

$$\begin{aligned} \mathcal{G}^N_{bind} \Psi^{n,N}_{\omega,\psi}(\mathfrak{e}) = N\beta p_0^N \sum_{x \in E} &\left(\psi\left(m_{F_x^N(\mathfrak{e})} \right) \Omega^{n,N}_\omega \left(F_x^N(\mathfrak{e}) \right) - \psi(m_{\mathfrak{e}}) \Omega^{n,N}_\omega(\mathfrak{e}) \right) \\ &+ N\beta p_2^N \sum_{x \in E} \left(\psi\left(m_{B_x^N(\mathfrak{e})} \right) \Omega^{n,N}_\omega \left(B_x^N(\mathfrak{e}) \right) - \psi(m_{\mathfrak{e}}) \Omega^{n,N}_\omega(\mathfrak{e}) \right), \end{aligned} \tag{2.40}$$

where F_x^N and B_x^N, are respectively defined in (2.37) and in (2.38).

Once again, by using Taylor expansions, we can obtain an easier expression for the binding generator defined in (2.40).

Lemma 2.5.7
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then

$$\mathcal{G}^N_{bind} \Psi^{n,N}_{\omega,\psi}(\mathfrak{e}) = \left(2a\beta m_{\mathfrak{e}} \psi'(m_{\mathfrak{e}}) + \frac{\beta m_{\mathfrak{e}}}{2} \psi''(m_{\mathfrak{e}}) \right) \left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}} \right)^{\otimes n,N} \right\rangle \tag{2.41}$$

$$+ \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2} \psi'(m_{\mathfrak{e}}) \right) \times \sum_{x \in E} \left(\left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu - \frac{1}{N}\delta_x}{m_{\mathfrak{e}} - \frac{1}{N}} \right)^{\otimes n,N} \right\rangle - \left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}} \right)^{\otimes n,N} \right\rangle \right) \tag{2.42}$$

$$+ \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2} \psi'(m_{\mathfrak{e}}) \right) \times \sum_{x \in E} \left(\left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu + \frac{1}{N}\delta_x}{m_{\mathfrak{e}} + \frac{1}{N}} \right)^{\otimes n,N} \right\rangle - \left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}} \right)^{\otimes n,N} \right\rangle \right) \tag{2.43}$$

$$+ \mathcal{O}\left(\frac{1}{N} \right).$$

Proof of Lemma 2.5.7. Recall that $U^n_{(E,d)}$ denotes the distance matrix map defined in Definition 2.4.1.

By definition of F^N_x, we have:

$$\psi\left(m_{F^N_x(\mathfrak{e})}\right) = \psi\left(m_e - \frac{1}{N}\right),$$

and

$$\Omega^{n,N}_\omega\left(F^N_x(\mathfrak{e})\right) = \left\langle \omega, \widehat{\left(\mu - \tfrac{1}{N}\delta_x\right)^{\otimes n,N}} \circ \left(U^n_{(E,d)}\right)^{-1} \right\rangle = \left\langle \omega\circ U^n_{(E,d)}, \widehat{\left(\mu - \tfrac{1}{N}\delta_x\right)^{\otimes n,N}} \right\rangle.$$

Moreover, Lemma 2.5.9 gives:

$$\Omega^{n,N}_\omega\left(F^N_x(\mathfrak{e})\right) = \left\langle \omega\circ U^n_{(E,d)}, \widehat{\left(\mu - \tfrac{1}{N}\delta_x\right)}^{\otimes n,N} \right\rangle = \left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu - \frac{1}{N}\delta_x}{m_\mathfrak{e} - \frac{1}{N}}\right)^{\otimes n,N} \right\rangle.$$

By doing the same for B^N_x, expression (2.40) can be denoted by:

$$\begin{aligned}
\mathcal{G}^N_{bind}\Psi^{n,N}_{\omega,\psi}(\mathfrak{e}) = N\beta p_0^N \sum_{x\in E}\Bigg(&\psi\left(m_\mathfrak{e} - \frac{1}{N}\right)\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu - \frac{1}{N}\delta_x}{m_\mathfrak{e} - \frac{1}{N}}\right)^{\otimes n,N} \right\rangle \\
&- \psi(m_\mathfrak{e})\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_\mathfrak{e}}\right)^{\otimes n,N} \right\rangle \Bigg) \\
+ N\beta p_2^N \sum_{x\in E}\Bigg(&\psi\left(m_\mathfrak{e} + \frac{1}{N}\right)\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu + \frac{1}{N}\delta_x}{m_\mathfrak{e} + \frac{1}{N}}\right)^{\otimes n,N} \right\rangle \\
&- \psi(m_\mathfrak{e})\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_\mathfrak{e}}\right)^{\otimes n,N} \right\rangle \Bigg).
\end{aligned}$$

By rearranging terms of this expression and by noticing that $\#E = Nm_\mathfrak{e}$, we obtain:

$$\begin{aligned}
\mathcal{G}^N_{bind}\Psi^{n,N}_{\omega,\psi}(\mathfrak{e}) = N^2 m_\mathfrak{e}\beta\left(p_0^N\left(\psi\left(m_\mathfrak{e} - \frac{1}{N}\right) - \psi(m_\mathfrak{e})\right) + p_2^N\left(\psi\left(m_\mathfrak{e} + \frac{1}{N}\right) - \psi(m_\mathfrak{e})\right)\right) & \\
\times \left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_\mathfrak{e}}\right)^{\otimes n,N} \right\rangle & \\
+ N\beta p_0^N \psi\left(m_\mathfrak{e} - \frac{1}{N}\right)\sum_{x\in E}\left(\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu - \frac{1}{N}\delta_x}{m_\mathfrak{e} - \frac{1}{N}}\right)^{\otimes n,N} \right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_\mathfrak{e}}\right)^{\otimes n,N} \right\rangle\right) & \\
+ N\beta p_2^N \psi\left(m_\mathfrak{e} + \frac{1}{N}\right)\sum_{x\in E}\left(\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu + \frac{1}{N}\delta_x}{m_\mathfrak{e} + \frac{1}{N}}\right)^{\otimes n,N} \right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_\mathfrak{e}}\right)^{\otimes n,N} \right\rangle\right). &
\end{aligned}$$

Finally, by using a second order Taylor expansion of $\psi\left(m_\mathfrak{e} \pm \frac{1}{N}\right)$, we have:

$$\begin{aligned}
N^2 m_\mathfrak{e}\beta\left(p_0^N\left(\psi\left(m_\mathfrak{e} - \frac{1}{N}\right) - \psi(m_\mathfrak{e})\right) + p_2^N\left(\psi\left(m_\mathfrak{e} + \frac{1}{N}\right) - \psi(m_\mathfrak{e})\right)\right) & \\
= 2a\beta m_\mathfrak{e}\psi'(m_\mathfrak{e}) + \frac{\beta m_\mathfrak{e}}{2}\psi''(m_\mathfrak{e}) + \mathcal{O}\left(\frac{1}{N}\right), &
\end{aligned}$$

since $p_2^N - p_0^N = \frac{2a}{N}$ and $p_2^N + p_0^N = 1$. This time using a first order Taylor expansion of $\psi\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)$, we have:

$$N\beta p_0^N \psi\left(m_{\mathfrak{e}} - \frac{1}{N}\right) = N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) + \mathcal{O}\left(\frac{1}{N}\right),$$

and

$$N\beta p_2^N \psi\left(m_{\mathfrak{e}} + \frac{1}{N}\right) = N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) + \mathcal{O}\left(\frac{1}{N}\right).$$

Remark 2.5.8

1. As ω is bounded and $\left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}$ is a probability measure, we have:

$$\left|\left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle\right| \leq \|\omega\|_\infty \text{ and } \left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle \times \mathcal{O}\left(\frac{1}{N}\right) = \mathcal{O}\left(\frac{1}{N}\right).$$

2. For the other two terms, we use Lemma 5.5.23 of [Glö13] that can be immediately adapted to our generator.

Lemma 2.5.9
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, $n \in \mathbb{N}, n \geq 2$ and $f : E^n \longrightarrow \mathbb{R}$ such that $f \in C_b$. Then

$$\left\langle f, \widehat{\mu^{\otimes n,N}}\right\rangle = \left\langle f, \hat{\mu}^{\otimes n,N}\right\rangle.$$

Proof of Lemma 2.5.9. Since, $\widehat{\mu^{\otimes n,N}} = \mu^{\otimes n,N} = \mathbf{0}_{\mathcal{M}}$ when $m_{\mathfrak{e}} = 0$ or $n > N m_{\mathfrak{e}}$, we only need to prove the equality when $m_{\mathfrak{e}} > 0$ and $n \leq N m_{\mathfrak{e}}$. We also recall that:

$$(m_{\mathfrak{e}})^{n,N} = m_{\mathfrak{e}}\left(m_{\mathfrak{e}} - \frac{1}{N}\right)\left(m_{\mathfrak{e}} - \frac{2}{N}\right)\cdots\left(m_{\mathfrak{e}} - \frac{n-1}{N}\right) = \prod_{i=1}^{n}\left(m_{\mathfrak{e}} - \frac{i-1}{N}\right).$$

We have:

$$\left\langle f, \widehat{\mu^{\otimes n,N}}\right\rangle = \left\langle f, \hat{\mu}^{\otimes n,N}\right\rangle + \left\langle f, \widehat{\mu^{\otimes n,N}} - \hat{\mu}^{\otimes n,N}\right\rangle,$$

and by using expression (2.28) for $\widehat{\mu^{\otimes n,N}}$ and expression (2.26) for $\hat{\mu}^{\otimes n,N}$, we have:

$$\begin{aligned}
\widehat{\mu^{\otimes n,N}} - \hat{\mu}^{\otimes n,N} &= \frac{1}{(m_{\mathfrak{e}})^{n,N}}\bigotimes_{i=1}^{n}\left(\mu - \frac{1}{N}\sum_{j=1}^{i-1}\delta_{e_j}\right)(de_1, de_2, \cdots, de_n) - \bigotimes_{i=1}^{n}\left(\frac{\mu}{m_{\mathfrak{e}}} - \frac{1}{N}\sum_{j=1}^{i-1}\delta_{e_j}\right)(de_1, de_2, \cdots, de_n),\\
&= \bigotimes_{i=1}^{n}\left(\frac{\mu}{m_{\mathfrak{e}} - \frac{i-1}{N}} - \frac{\frac{1}{N}\sum_{j=1}^{i-1}\delta_{e_j}}{m_{\mathfrak{e}} - \frac{i-1}{N}} - \frac{\mu}{m_{\mathfrak{e}}} + \frac{1}{N}\sum_{j=1}^{i-1}\delta_{e_j}\right)(de_1, de_2, \cdots, de_n),\\
&= \mathbf{0}_{\mathcal{M}}\bigotimes_{i=2}^{n}\left(\frac{\mu}{m_{\mathfrak{e}} - \frac{i-1}{N}} - \frac{\frac{1}{N}\sum_{j=1}^{i-1}\delta_{e_j}}{m_{\mathfrak{e}} - \frac{i-1}{N}} - \frac{\mu}{m_{\mathfrak{e}}} + \frac{1}{N}\sum_{j=1}^{i-1}\delta_{e_j}\right)(de_1, de_2, \cdots, de_n).
\end{aligned}$$

Therefore, we have:

$$\left\langle f, \widehat{\mu^{\otimes n,N}} - \hat{\mu}^{\otimes n,N}\right\rangle = 0,$$

which proves this lemma.

The last lemma of this section will be useful in the proof of the tightness.

Lemma 2.5.10
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b^{1bc}(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then there exists two constants $C_{\mathcal{G},1} > 0$ and $C_{\mathcal{G},2} > 0$ such that, for every $N \in \mathbb{N}^*$,

$$\left|\mathcal{G}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e})\right| \leq m_{\mathfrak{e}} C_{\mathcal{G},1} + C_{\mathcal{G},2}.$$

Proof of Lemma 2.5.10. By definition 2.5.1, we have:

$$\left|\mathcal{G}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e})\right| \leq \left|\mathcal{G}_{grow}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e})\right| + \left|\mathcal{G}_{bind}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e})\right|,$$

so, we are going to prove that both $\left|\mathcal{G}_{grow}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e})\right|$ and $\left|\mathcal{G}_{bind}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e})\right|$ are bounded.

Lemma 2.5.4 ensures that there exists a constant $c_1 > 0$ such that:

$$\begin{aligned}\left|\mathcal{G}_{grow}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e})\right| &\leq \left|\psi(m_{\mathfrak{e}}) \int_{E^n} 2\lambda^+ \overline{\nabla}\omega\left(\left(d(e_i,e_j)\right)_{1\leq i<j\leq n}\right) \widehat{d\mu^{\otimes n,N}}(e_1,\cdots,e_n)\right| + \frac{c_1}{N},\\ &\leq 2\lambda^+ ||\psi||_\infty \left\|\overline{\nabla}\omega\right\|_\infty + c_1.\end{aligned}$$

Lemma 2.5.7 ensures that there exists a constant $c_2 > 0$ such that:

$$\begin{aligned}\left|\mathcal{G}_{bind}^N \Psi_{\omega,\psi}^{n,N}(\mathfrak{e})\right| \leq{}& \left|2a\beta m_{\mathfrak{e}} \psi'(m_{\mathfrak{e}}) + \frac{\beta m_{\mathfrak{e}}}{2}\psi''(m_{\mathfrak{e}})\right| \times \left|\left\langle \omega\circ U_{(E,d)}^n, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle\right|\\ &+ \left|N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right|\\ &\qquad \times \left|\sum_{x\in E}\left(\left\langle \omega\circ U_{(E,d)}^n, \left(\frac{\mu - \frac{1}{N}\delta_x}{m_{\mathfrak{e}} - \frac{1}{N}}\right)^{\otimes n,N}\right\rangle - \left\langle \omega\circ U_{(E,d)}^n, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle\right)\right|\\ &+ \left|N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right|\\ &\qquad \times \left|\sum_{x\in E}\left(\left\langle \omega\circ U_{(E,d)}^n, \left(\frac{\mu + \frac{1}{N}\delta_x}{m_{\mathfrak{e}} + \frac{1}{N}}\right)^{\otimes n,N}\right\rangle - \left\langle \omega\circ U_{(E,d)}^n, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle\right)\right|\\ &+ \frac{c_2}{N}.\end{aligned}$$

Moreover, expression (5.382) page 195 of [Glö13] ensures that:

$$\sum_{x\in E}\left(\left\langle \omega\circ U_{(E,d)}^n, \left(\frac{\mu - \frac{1}{N}\delta_x}{m_{\mathfrak{e}} - \frac{1}{N}}\right)^{\otimes n,N}\right\rangle - \left\langle \omega\circ U_{(E,d)}^n, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle\right) = \mathcal{O}\left(\frac{1}{N}\right),$$

and

$$\sum_{x\in E}\left(\left\langle \omega\circ U_{(E,d)}^n, \left(\frac{\mu + \frac{1}{N}\delta_x}{m_{\mathfrak{e}} + \frac{1}{N}}\right)^{\otimes n,N}\right\rangle - \left\langle \omega\circ U_{(E,d)}^n, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle\right) = \mathcal{O}\left(\frac{1}{N}\right).$$

Therefore, there exists two constants $c_3 > 0$ and $c_4 > 0$ such that:

$$\begin{aligned}\left|\mathcal{G}^N_{bind}\Psi^{n,N}_{\omega,\psi}(\mathfrak{e})\right| &\leq \left(2|a|\beta m_{\mathfrak{e}}\|\psi'\|_\infty + \frac{\beta m_{\mathfrak{e}}}{2}\|\psi''\|_\infty\right)\|\omega\|_\infty + \left|N\beta p_0^N\psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right| \times \frac{c_3}{N} \\ &\quad + \left|N\beta p_2^N\psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right| \times \frac{c_4}{N} + c_2, \\ &\leq m_{\mathfrak{e}}\left(2|a|\beta\|\psi'\|_\infty + \frac{\beta}{2}\|\psi''\|_\infty\right)\|\omega\|_\infty + \left(\beta p_0^N\|\psi\|_\infty + \frac{\beta}{2}\|\psi'\|_\infty\right)c_3 \\ &\quad + \left(\beta p_2^N\|\psi\|_\infty + \frac{\beta}{2}\|\psi'\|_\infty\right)c_4 + c_2.\end{aligned}$$

So, we have:

$$\begin{aligned}\left|\mathcal{G}^N\Psi^{n,N}_{\omega,\psi}(\mathfrak{e})\right| &\leq +m_{\mathfrak{e}}\left(2|a|\beta\|\psi'\|_\infty + \frac{\beta}{2}\|\psi''\|_\infty\right)\|\omega\|_\infty + 2\lambda^+\|\psi\|_\infty\left\|\overline{\nabla}\omega\right\|_\infty + c_1 + \left(\beta p_0^N\|\psi\|_\infty + \frac{\beta}{2}\|\psi'\|_\infty\right)c_3 \\ &\quad + \left(\beta p_2^N\|\psi\|_\infty + \frac{\beta}{2}\|\psi'\|_\infty\right)c_4 + c_2.\end{aligned}$$

Since $\omega \in \mathcal{C}_b^{1bc}$ and $\psi \in \mathcal{C}_K^\infty$, functions ω, $\overline{\nabla}\omega$, ψ, ψ' and ψ'' are bounded and the result is proved. ✠

6 - Martingale Problem

Thanks to the generator $\mathcal{G}^N$ of the discrete tree-valued process $\mathbb{T}^N$ defined in the previous section, we can define a martingale $\boldsymbol{M}^N_\Psi$ and prove that our process solves a martingale problem.

Proposition 2.6.1
Let $\Psi \in {}^{\mathcal{A}}\Pi^N_\Psi(\mathcal{C}_b^{1bc}, \mathcal{C}_K^\infty)$ and define, for $t \in \mathbb{R}_+$:

$$M^N_\Psi(t) \overset{def}{=} \Psi(\mathbb{T}^N_t) - \Psi(\mathbb{T}^N_0) - \int_0^t \mathcal{G}^N\Psi(\mathbb{T}^N_s)ds.$$

Then $\boldsymbol{M}^N_\Psi = (M^N_\Psi(t))_{t\in\mathbb{R}_+}$ is a martingale.

Proof of Proposition 2.6.1. The proof is essentially the same as that of the Proposition 3.2.17 of [Glö13]. Indeed, even if our generators $\mathcal{G}^N_{grow}$ and $\mathcal{G}^N_{bind}$ are slightly different, they have the same characteristics. ✠

The next theorem is immediate from Proposition 2.6.1 and Theorem 3.2.2 of [Glö13].

Theorem 2.6.2
The process $\mathbb{T}^N$ solves the martingale problem for $\left(\mathcal{G}^N, {}^{\mathcal{A}}\Pi^N_\Psi(\mathcal{C}_b^{1bc}, \mathcal{C}_K^\infty)\right)$ and has sample paths in $D_{\mathbb{T}^N}[0, +\infty[$.

The following results give expressions for the quadratic variation of the martingale $\boldsymbol{M}^N_\Psi$ and the semi-martingale decomposition of the process $\left(\Psi\left(\mathbb{T}^N_t\right)\right)_{t\in\mathbb{R}_+}$. This will be useful in the next section where we will prove that our stopped tree-valued process is tight.

Proposition 2.6.3
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b^{1bc}$ and $\psi \in C_K^\infty$. To simplify the notation, we denote $\Psi_{\omega,\psi}^{n,N}$ by Ψ. For each $t \in \mathbb{R}_+$, the predictable quadratic variation of $\boldsymbol{M_\Psi^N}$ is:

$$\left\langle M_\Psi^N \right\rangle (t) \overset{def}{=} \int_0^t g_\Psi^N \left(\mathfrak{T}_s^N \right) ds,$$

where, for every $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$:

$$\begin{aligned} g_\Psi^N(\mathfrak{e}) = N\lambda^+ \sum_{k=1}^n \Bigg(\psi(m_\mathfrak{e}) \int_{E^n} \left(\omega \left(\left(\Delta_{e_k}^N(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) - \omega \left(\left(d(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) \right) d\widehat{\mu^{\otimes n,N}}(e_1, \cdots, e_n) \Bigg)^2 \\ + N\beta p_0^N \sum_{x \in E} \left(\psi \left(m_{F_x^N(\mathfrak{e})} \right) \Omega_\omega^{n,N} \left(F_x^N(\mathfrak{e}) \right) - \psi(m_\mathfrak{e}) \Omega_\omega^{n,N}(\mathfrak{e}) \right)^2 \\ + N\beta p_2^N \sum_{x \in E} \left(\psi \left(m_{B_x^N(\mathfrak{e})} \right) \Omega_\omega^{n,N} \left(B_x^N(\mathfrak{e}) \right) - \psi(m_\mathfrak{e}) \Omega_\omega^{n,N}(\mathfrak{e}) \right)^2 \end{aligned}$$

with Δ^N, F_x^N and B_x^N, respectively defined in (2.31), in (2.37) and in (2.38).

Lemma 2.6.4
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b^{1bc}(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then there exists two constants $C_{g,1} > 0$ and $C_{g,2} > 0$ such that, for every $N \in \mathbb{N}^*$,

$$\left| g_\Psi^N(\mathfrak{e}) \right| \leq m_\mathfrak{e} C_{g,1} + C_{g,2}.$$

Proof of Lemma 2.6.4. The expression given in Proposition 2.6.3 yields the bound:

$$\begin{aligned} \left| g_\Psi^N(\mathfrak{e}) \right| \leq N\lambda^+ \sum_{k=1}^n \left| \psi(m_\mathfrak{e}) \int_{E^n} \left(\omega \left(\left(\Delta_{e_k}^N(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) - \omega \left(\left(d(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) \right) d\widehat{\mu^{\otimes n,N}}(e_1, \cdots, e_n) \right|^2 \\ + N\beta \left| \sum_{x \in E} \left(\psi \left(m_{F_x^N(\mathfrak{e})} \right) \Omega_\omega^{n,N} \left(F_x^N(\mathfrak{e}) \right) - \psi(m_\mathfrak{e}) \Omega_\omega^{n,N}(\mathfrak{e}) \right)^2 \right| \\ + N\beta \left| \sum_{x \in E} \left(\psi \left(m_{B_x^N(\mathfrak{e})} \right) \Omega_\omega^{n,N} \left(B_x^N(\mathfrak{e}) \right) - \psi(m_\mathfrak{e}) \Omega_\omega^{n,N}(\mathfrak{e}) \right)^2 \right|. \end{aligned}$$

We will treat the first and second terms on the right-hand-side separately. Note that we can bound the third term in exactly the same way as the second.

For the first term, we have:

$$\begin{aligned} N\lambda^+ \sum_{k=1}^n \left| \psi(m_\mathfrak{e}) \int_{E^n} \left(\omega \left(\left(\Delta_{e_k}^N(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) - \omega \left(\left(d(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) \right) d\widehat{\mu^{\otimes n,N}}(e_1, \cdots, e_n) \right|^2 \\ \leq N\lambda^+ \sum_{k=1}^n \|\psi\|_\infty \int_{E^n} \left| \omega \left(\left(\Delta_{e_k}^N(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) - \omega \left(\left(d(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) \right|^2 d\widehat{\mu^{\otimes n,N}}(e_1, \cdots, e_n), \end{aligned}$$

thanks to the Jensen's inequality. In the same way as in the proof of Lemma 2.5.4, we have:

$$\begin{aligned} \left| \omega \left(\left(\Delta_{e_k}^N(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) - \omega \left(\left(d(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) \right|^2 &= \left| \frac{1}{N} \sum_{\substack{1 \leq i < j \leq n \\ i=k \text{ or } j=k}} \frac{\partial \omega}{\partial d(e_i, e_j)} \left(\left(d(e_i, e_j) \right)_{1 \leq i < j \leq n} \right) + \mathcal{O}\left(\frac{1}{N^2} \right) \right|^2, \\ &= \mathcal{O}\left(\frac{1}{N^2} \right), \end{aligned}$$

where the bound is uniform in $\mathfrak{e}$.

Therefore, as $\omega \in \mathcal{C}_b^{1bc}$ and $\psi \in C_K^\infty$, we have:

$$N\lambda^+ \sum_{k=1}^{n} \|\psi\|_\infty \int_{E^n} \left| \omega\left(\left(\Delta_{e_k}^N(e_i,e_j)\right)_{1\le i<j\le n} \right) - \omega\left(\left(d(e_i,e_j)\right)_{1\le i<j\le n} \right) \right|^2 \widehat{d\mu^{\otimes n,N}}(e_1,\cdots,e_n) = \mathcal{O}\left(\frac{1}{N}\right).$$

For the second term, we use the same rearrangement as in Lemma 2.5.7. We have:

$$\begin{aligned}
&\left(\psi\left(m_{F_x^N(\mathfrak{e})}\right)\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \psi(m_\mathfrak{e})\Omega_\omega^{n,N}(\mathfrak{e})\right)^2 \\
&\qquad = \left(\left[\psi\left(m_{F_x^N(\mathfrak{e})}\right) - \psi(m_\mathfrak{e})\right]\Omega_\omega^{n,N}(\mathfrak{e}) + \psi\left(m_{F_x^N(\mathfrak{e})}\right)\left[\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \Omega_\omega^{n,N}(\mathfrak{e})\right]\right)^2, \\
&\qquad = \left(\left[\psi\left(m_{F_x^N(\mathfrak{e})}\right) - \psi(m_\mathfrak{e})\right]\Omega_\omega^{n,N}(\mathfrak{e})\right)^2 + \left(\psi\left(m_{F_x^N(\mathfrak{e})}\right)\left[\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \Omega_\omega^{n,N}(\mathfrak{e})\right]\right)^2 \\
&\qquad\qquad + 2\left[\psi\left(m_{F_x^N(\mathfrak{e})}\right) - \psi(m_\mathfrak{e})\right]\Omega_\omega^{n,N}(\mathfrak{e})\psi\left(m_{F_x^N(\mathfrak{e})}\right)\left[\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \Omega_\omega^{n,N}(\mathfrak{e})\right].
\end{aligned}$$

Therefore,

$$\begin{aligned}
&N\beta\left|\sum_{x\in E}\left(\psi\left(m_{F_x^N(\mathfrak{e})}\right)\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \psi(m_\mathfrak{e})\Omega_\omega^{n,N}(\mathfrak{e})\right)^2\right| \\
&\qquad \le N\beta\sum_{x\in E}\left|\left[\psi\left(m_{F_x^N(\mathfrak{e})}\right) - \psi(m_\mathfrak{e})\right]\Omega_\omega^{n,N}(\mathfrak{e})\right|^2 + N\beta\left|\sum_{x\in E}\psi\left(m_{F_x^N(\mathfrak{e})}\right)\left[\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \Omega_\omega^{n,N}(\mathfrak{e})\right]\right|^2 \\
&\qquad\qquad + N\beta\left|\sum_{x\in E} 2\left[\psi\left(m_{F_x^N(\mathfrak{e})}\right) - \psi(m_\mathfrak{e})\right]\Omega_\omega^{n,N}(\mathfrak{e})\psi\left(m_{F_x^N(\mathfrak{e})}\right)\left[\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \Omega_\omega^{n,N}(\mathfrak{e})\right]\right|, \\
&\qquad \le N^2\beta m_\mathfrak{e}\left|\psi\left(m_e - \frac{1}{N}\right) - \psi(m_\mathfrak{e})\right|^2\|\omega\|_\infty^2 + N\beta\|\psi\|_\infty^2\left|\sum_{x\in E}\left[\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \Omega_\omega^{n,N}(\mathfrak{e})\right]\right|^2 \\
&\qquad\qquad + 4N\beta\|\psi\|_\infty\|\omega\|_\infty^2\left|\sum_{x\in E}\left[\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \Omega_\omega^{n,N}(\mathfrak{e})\right]\right|.
\end{aligned}$$

Once again, we treat these three terms separately. We have:

$$\begin{aligned}
N^2\beta m_\mathfrak{e}\left|\psi\left(m_e - \frac{1}{N}\right) - \psi(m_\mathfrak{e})\right|^2\|\omega\|_\infty^2 &= N^2\beta m_\mathfrak{e}\left|-\frac{1}{N}\psi'(m_\mathfrak{e}) + \mathcal{O}\left(\frac{1}{N^2}\right)\right|^2\|\omega\|_\infty^2 \\
&= \beta m_\mathfrak{e}\|\omega\|_\infty^2\left(|\psi'(m_\mathfrak{e})|^2 + \mathcal{O}\left(\frac{1}{N}\right)\right), \\
&\le \beta m_\mathfrak{e}\|\omega\|_\infty^2\|\psi'\|_\infty^2 + \mathcal{O}\left(\frac{1}{N}\right)m_\mathfrak{e},
\end{aligned}$$

where the term $\mathcal{O}\left(\frac{1}{N}\right)$ can be chosen to be independent of $\mathfrak{e}$.

For the third term, we use expression (5.382) page 195 of [Glö13] that ensures that:

$$\sum_{x\in E}\left[\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \Omega_\omega^{n,N}(\mathfrak{e})\right] = \mathcal{O}\left(\frac{1}{N}\right).$$

So, we have:

$$N\beta\|\psi\|_\infty^2\left|\sum_{x\in E}\left[\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \Omega_\omega^{n,N}(\mathfrak{e})\right]\right|^2 = \mathcal{O}\left(\frac{1}{N}\right),$$

and

$$4N\beta\|\psi\|_\infty\|\omega\|_\infty^2\left|\sum_{x\in E}\left[\Omega_\omega^{n,N}\left(F_x^N(\mathfrak{e})\right) - \Omega_\omega^{n,N}(\mathfrak{e})\right]\right| = \mathcal{O}(1).$$

Finally, there exists two constants $c_1 > 0$ and $c_2 > 0$ such that, for N large enough,

$$N\beta \left| \sum_{x \in E} \left(\psi \left(m_{F_x^N(\mathfrak{e})} \right) \Omega_\omega^{n,N} \left(F_x^N(\mathfrak{e}) \right) - \psi(m_\mathfrak{e}) \Omega_\omega^{n,N}(\mathfrak{e}) \right)^2 \right| \leq m_\mathfrak{e} c_1 + c_2.$$

Proposition 2.6.5
The semi-martingale decomposition of $\Psi(\mathfrak{T}^N)$ is for each $t \in \mathbb{R}_+$:

$$\Psi(\mathfrak{T}_t^N) = \Psi(\mathfrak{T}_0^N) + M_\Psi^N(t) + V_\Psi^N(t),$$

with:

$$V_\Psi^N(t) = \int_0^t \mathcal{G}^N \Psi(\mathfrak{T}_s^N) ds.$$

7 - Tightness

The last step before proving that our tree-valued process converges in law is to prove that it is tight. Because of technical problems due to the many ways our process may approach the value $\mathfrak{n}$ representing an extinct population, to prove the tightness, we must restrict $\mathfrak{T}^N$ to its non-zero values. We will therefore study the process before the extinction time.

In this section, we define the stopped process $\mathfrak{T}_R^N$ and then we quote the two criteria we will use to prove that this stopped process is tight. It is interesting to note that these criteria are "classical" even though our state-space of interest is rather unusual. The second subsection focuses on the proof.

For $\delta \geq 0$, we denote by:

$$\tau_\delta^N = \inf\{ t \geq 0 \mid \mathfrak{m}_t^N = \tilde{X}_t^N \leq \delta \},$$

the first time the total mass reaches δ or drops below this level. Let $R \in \mathbb{N}^*$, we define the stopped process $\mathfrak{T}_R^N = \left(\mathfrak{T}_{t,R}^N \right)_{t \in \mathbb{R}_+}$, where for all $t \in \mathbb{R}_+$,

$$\mathfrak{T}_{t,R}^N \overset{def}{=} \mathfrak{T}^N \left(\min \left(t, \tau_{1/R}^N \right) \right).$$

We wish to show the following result.

Proposition 2.7.1 (Tightness)
Assume the following conditions :

1. $\limsup_{N \to +\infty} \mathbb{E}\left[\mathfrak{m}_0^N \right] < \infty$,
2. $\exists \nu \in \mathcal{M}_1(\mathbb{R}_+)$ such that $\mathfrak{m}_0^N \xrightarrow[N \to +\infty]{\mathcal{L}} \nu$,
3. $\nu(\{0\}) = 0$,
4. $\left\{ \mathcal{L}\text{aw}\left[\hat{\mathfrak{T}}_0^N \right] \mid N \in \mathbb{N} \right\}$ is tight in $\mathcal{M}_1(\mathbb{M}_1)$.

Then, for each $R \in \mathbb{N}^*$, the family $\left(\mathfrak{T}_{\cdot,R}^N \right)_{N \in \mathbb{N}}$ is tight.

This proposition will be proved in Section 7.2.

7.1 - Tightness Criteria

The first criterion allows to prove the tightness of a process with non-real values on the basis of three elements: a convenient class of functions, the compact containment condition and the weak tightness. For our tree-valued process, we will consider the linear span of Ψ-polynomials ${}^{S}\Pi_{\Psi}(\mathcal{C}_b^{1bc}, \mathcal{C}_K^{\infty})$, introduced in Section 4.2.2.

Theorem 2.7.2 (Jakubowski's Tightness Criterion [Jak86])
Let $(S, \mathcal{T})$ be a completely regular topological space with metrisable compacts. Let $\mathcal{F}$ be a family of real-valued continuous functions on S. Suppose that:

i - $\mathcal{F}$ separates points in S,

ii - $\mathcal{F}$ is closed under addition.

Then, the following holds:

1. A sequence of stochastic processes $(Y_n(\cdot))_{n\in\mathbb{N}}$ with sample paths in $D_S[0, +\infty[$ is tight if and only if the following two conditions hold:

   iii - For each $\varepsilon > 0$ and $T > 0$, there exists a compact $K_{\varepsilon,T} \subseteq S$ such that:

   $$\liminf_{n\to+\infty} \mathbb{P}\left(\forall t \in [0, T],\ Y_n(t) \in K_{\varepsilon,T}\right) \geq 1 - \varepsilon.$$

   iv - The sequence $(Y_n(\cdot))_{n\in\mathbb{N}}$ is $\mathcal{F}$-weakly tight, i.e. for each $f \in \mathcal{F}$, the sequence $(f(Y_n(\cdot)))_{n\in\mathbb{N}}$ is tight on $D_{\mathbb{R}}[0, +\infty[$.

2. If $(Y_n(\cdot))_{n\in\mathbb{N}}$ is tight, then it is relatively compact in the weak topology.

To prove the weak tightness (condition iv of the previous theorem), we use the following theorem due to Aldous [Ald78] and Rebolledo [Reb80]. We cite here the version of [Eth00]. This is actually the same criterion as the one used in Chapter 1.

Theorem 2.7.3 (Aldous-Rebolledo Tightness Criterion [Eth00])
Let $(Y_n(\cdot))_{n\in\mathbb{N}}$ be a sequence of real valued semi-martingale with càdlàg paths.

Let $Y_n(t) = Y_n(0) + M_n(t) + V_n(t)$ be the canonical decomposition of Y_n, where M_n is a locally square integrable local martingale with $M_n(0) = 0$ and V_n is a predictable finite variation process with $V_n(0) = 0$. Let $\langle M_n \rangle$ denote the unique predictable increasing process such that $M_n^2 - \langle M_n \rangle$ is a local martingale. Suppose that the following conditions are satisfied:

i - For each $t \geq 0$, $(Y_n(t))_{n\in\mathbb{N}}$ is tight.

ii - For each $T > 0$, for each sequence of stopping times $(\tau_n)_{n\in\mathbb{N}}$ bounded by T, and for each $\varepsilon > 0$, there exists $\delta > 0$ and n_0 such that:

   (a) $\sup_{n\geq n_0} \sup_{\theta\in[0,\delta]} \mathbb{P}\left(|V_n(\tau_n + \theta) - V_n(\tau_n)| > \varepsilon\right) \leq \varepsilon,$

   (b) $\sup_{n\geq n_0} \sup_{\theta\in[0,\delta]} \mathbb{P}\left(|\langle M_n \rangle(\tau_n + \theta) - \langle M_n \rangle(\tau_n)| > \varepsilon\right) \leq \varepsilon.$

Then, $(Y_n(\cdot))_{n\in\mathbb{N}}$ is tight.

7.2 - Proof of the Tightness of the Stopped Tree-Valued Process

Proof of Proposition 2.7.1. We use Jakubowski's Tightness Criterion.

- Proposition 2.4.5 ensures that $\mathbb{T}$ is a Polish space, so $\mathbb{T}$ is completely regular.
- The class ${}^{S}\Pi_{\Psi}(\mathcal{C}_b^{1bc}, \mathcal{C}_K^{\infty})$ is closed under addition and Proposition 2.4.8 ensures that this function class separates points on $\mathbb{T}$.
- By Proposition 2.6.2, the process $\mathfrak{T}^N$ has sample paths in $D_{\mathbb{T}}[0, +\infty[$.
- We prove in Proposition 2.7.4 that the family $\left(\mathfrak{T}_{\cdot}^N\right)_{N\in\mathbb{N}}$ satisfies the compact containment condition.
- We prove in Corollary 2.7.8 that for all $\Psi \in {}^{S}\Pi_{\Psi}(\mathcal{C}_b^{1bc}, \mathcal{C}_K^{\infty})$, the family $\left(\Psi\left(\mathfrak{T}_{\cdot,R}^N\right)\right)_{N\in\mathbb{N}}$ is tight for each $R \in \mathbb{N}^*$.

Therefore, by point 1 of Theorem 2.7.2, the family $\left(\mathfrak{T}_{\cdot,R}^N\right)_{N\in\mathbb{N}}$ is tight for each $R \in \mathbb{N}^*$. ❇

7.2.1 - Compact containment condition

Proposition 2.7.4 (Compact Containment)
Assume the four conditions of Proposition 2.7.1. Then, for every $\varepsilon > 0$ and $T > 0$, there is a compact set $K_{\varepsilon,T} \subseteq \mathbb{T}$, such that:

$$\inf_{N\in\mathbb{N}} \mathbb{P}\left(\forall t \in [0, T],\ \mathfrak{T}_t^N \in K_{\varepsilon,T}\right) \geq 1 - \varepsilon.$$

Proof of Proposition 2.7.4. We follow the proof of Proposition 4.5.1 of [Glö13]. Recall that π denotes the polar decomposition map defined in Definition 2.4.4.

As π is a continuous bijection between $(\mathbb{T}, d_{GP}^{\mathbb{T}})$ and $(\mathbb{R}_+ \times \mathbb{M}_1, d_{eucl} \otimes d_{GP})$, we can note that a compact set $K \subset \mathbb{T}$ is compact if and only if $\pi(K) \subseteq \mathbb{R}_+ \times \mathbb{M}_1$ is compact in $(\mathbb{R}_+ \times \mathbb{M}_1, d_{eucl} \otimes d_{GP})$.

Let $T > 0$ and $\varepsilon > 0$. Proposition 2.7.5 ensures that there are compact sets $K_{\varepsilon,T}^{mass} \subset \mathbb{R}_+$ and $K_{\varepsilon,T}^{gen} \subset \mathbb{M}_1$ such that, for all $N \in \mathbb{N}$,

$$\mathbb{P}\left(\forall t \in [0, T],\ \mathfrak{m}_t^N \in K_{\varepsilon,T}^{mass}\right) \geq 1 - \frac{\varepsilon}{3},$$

and

$$\mathbb{P}\left(\forall t \in [0, T],\ \hat{\mathfrak{T}}_t^N \in K_{\varepsilon,T}^{gen}\right) \geq 1 - \frac{\varepsilon}{3}.$$

Defining

$$K_{\varepsilon,T} = \pi^{-1}\left(K_{\varepsilon,T}^{mass} \times K_{\varepsilon,T}^{gen}\right),$$

$K_{\varepsilon,T}$ is compact in $\mathbb{T}$ (thanks to the previous remark) and we have, for all $N \in \mathbb{N}$,

$$\begin{aligned}\mathbb{P}\left(\forall t \in [0, T],\ \mathfrak{T}_t^N \in K_{\varepsilon,T}\right) &= \mathbb{P}\left(\forall t \in [0, T],\ \left(\mathfrak{m}_t^N, \hat{\mathfrak{T}}_t^N\right) \in K_{\varepsilon,T}^{mass} \times K_{\varepsilon,T}^{gen}\right), \\ &= \mathbb{P}\left(\left(\forall t \in [0, T],\ \mathfrak{m}_t^N \in K_{\varepsilon,T}^{mass}\right) \cap \left(\forall t \in [0, T],\ \hat{\mathfrak{T}}_t^N \in K_{\varepsilon,T}^{gen}\right)\right).\end{aligned}$$

Finally, we conclude this proof with Lemma 2.7.6. ❇

Proposition 2.7.5
Assume the four conditions of Proposition 2.7.1. Then for every $T > 0$ and $\varepsilon > 0$, there are a compact set $K^{mass}_{\varepsilon,T} \subset \mathbb{R}_+$ and a compact set $K^{gen}_{\varepsilon,T} \subset \mathbb{M}_1$ such that:

1. $\inf_{N\in\mathbb{N}} \mathbb{P}\left(\forall t \in [0,T],\ \mathfrak{m}^N_t \in K^{mass}_{\varepsilon,T}\right) \geq 1-\varepsilon.$
2. $\inf_{N\in\mathbb{N}} \mathbb{P}\left(\forall t \in [0,T],\ \hat{\mathfrak{T}}^N_t \in K^{gen}_{\varepsilon,T}\right) \geq 1-\varepsilon.$

Proof of Proposition 2.7.5.

1. To prove that the total mass satisfies the compact containment condition, we study its dynamics. The process $\left(\mathfrak{m}^N_t\right)_{t\in\mathbb{R}_+}$ is affected by two events: fragmentation and branching. When a fragmentation occurs, the total mass loses a mass $\frac{1}{N}$ whereas when a branching occurs, the total mass gains a mass $\frac{1}{N}$. As these events are the same that defined a binding event, we can proceed in the same way as in Section 5.2. So, the generator $\mathcal{G}^N_{mass}$ of the total mass process is defined for a bounded continuous function $f : \mathbb{R}_+ \longrightarrow \mathbb{R}$, by:

$$\mathcal{G}^N_{mass} f(m) = N^2 m\beta p^N_0 \left(f\left(m-\frac{1}{N}\right) - f(m)\right) + N^2 m\beta p^N_2 \left(f\left(m+\frac{1}{N}\right) - f(m)\right).$$

Assuming that $f \in \mathcal{C}^2(\mathbb{R}_+)$, a second order Taylor expansion gives:

$$\begin{aligned}
\mathcal{G}^N_{mass} f(m) &= N^2 m\beta p^N_0 \left(-\frac{1}{N}f'(m) + \frac{1}{2N^2}f''(m) + \mathcal{O}\left(\frac{1}{N^3}\right)\right) \\
&\qquad + N^2 m\beta p^N_2 \left(\frac{1}{N}f'(m) + \frac{1}{2N^2}f''(m) + \mathcal{O}\left(\frac{1}{N^3}\right)\right), \\
&= Nm\beta\left(p^N_2 - p^N_0\right)f'(m) + \frac{m\beta}{2}\left(p^N_0 + p^N_2\right)f''(m) + \mathcal{O}\left(\frac{1}{N}\right), \\
&= 2a\beta m f'(m) + \frac{\beta}{2} m f''(m) + \mathcal{O}\left(\frac{1}{N}\right).
\end{aligned}$$

We recognize a birth-death process which converges in law to a Feller's branching diffusion. So, the family $\left(\mathfrak{m}^N_\cdot\right)_{N\in\mathbb{N}}$ is tight and the compact containment condition is checked.

2. We follow the proof of Proposition 4.5.3 (ii) of [Glö13]. Recall that, for all $t \in \mathbb{R}_+$, $\#\mathfrak{A}^N_s(t)$ denotes the number of ancestors at time s of individuals from $\mathfrak{I}^N_t$ defined in (2.15). We also recall that the process $\hat{\mathfrak{T}}^N$ contains the information about the genealogical structure of the population (while the total mass encodes its size). Therefore, to prove the compact containment condition for $\hat{\mathfrak{T}}^N$, we need to control the number of ancestors and the number of descendants these ancestors will produce.

Let $T > 0$ and $\varepsilon > 0$. In order to control the number of ancestors, we need to show that for all $0 < 2\varepsilon \leq t \leq T$, the family $\left(\#\mathfrak{A}^N_{t-2\varepsilon}(t)\right)_{N\in\mathbb{N}}$ is tight. For this, we use the decomposition proposed in Proposition 4.6.6 of [Glö13]. Adapting this result to our model, we have: for all $k \in \mathbb{N}$ and $0 < 2\varepsilon \leq t \leq T$,

$$\mathbb{P}\left(\#\mathfrak{A}^N_{t-2\varepsilon}(t) = k\right) = \sum_{j=k}^{+\infty} \mathbb{P}\left(\#\mathfrak{I}^N_{t-2\varepsilon} = j\right) \binom{j}{k} \left(f_{N\beta 2\varepsilon}\right)^k \left(1 - f_{N\beta 2\varepsilon}\right)^{j-k},$$

where for all $u \geq 0$,

$$f_u = \mathbb{P}\left(Y(u) > 0\right)$$

and Y is a continuous time, rate one Galton-Watson process with $Y(0) = 1$ and offspring distribution $\boldsymbol{p^N}$.

- If $a = 0$ (critical case), the only difference between our model and the one of [Glö13] is that we do not assume that $\beta = 1$. So, Proposition 4.6.2 of [Glö13] ensures that the family $\left(\#\mathfrak{A}^N_{t-2\varepsilon}(t)\right)_{N\in\mathbb{N}}$ is tight in the critical case.
- If $a < 0$ (sub-critical case), the probability of survival at time u of the Galton-Watson process f_u in this case is smaller than in the critical case. Since the proof of Proposition 4.6.2 of [Glö13] is based on the equivalent: for all $u > 0$, $f_{Nu} \underset{N}{\sim} \frac{2}{Nu\sigma^2}$ where σ^2 is the offspring variance, this proof can be adapted and this ensures that the family $\left(\#\mathfrak{A}^N_{t-2\varepsilon}(t)\right)_{N\in\mathbb{N}}$ is tight in the sub-critical case.
- If $a > 0$ (super-critical case), we need to find the equivalent of f_{Nu} in this case.
  We see here Y, the continuous time rate one Galton-Watson process with $Y(0) = 1$ and offspring distribution $\boldsymbol{p^N}$ as a binary birth and death process, where each individuals, independently, gives birth to a offspring at rate $b = 1 \times p_2^N = \frac{1}{2} + \frac{a}{N}$ and dies naturally at rate $d = 1 \times p_0^N = \frac{1}{2} - \frac{a}{N}$.
  Let $q(t) = \mathbb{P}_1(Y(t) = 0) = 1 - f_t$. Since $q(t) = \left(P_t \mathbb{1}_{\{0\}}\right)(1)$, the Kolmogorov equation for the function $\mathbb{1}_{\{0\}}$ gives:

$$\begin{aligned}\frac{d}{dt}\left(P_t\mathbb{1}_{\{0\}}\right)(1) &= Q\left(P_t\mathbb{1}_{\{0\}}\right)(1),\\ &= b\left(P_t\mathbb{1}_{\{0\}}\right)(2) + d\left(P_t\mathbb{1}_{\{0\}}\right)(0) - (b+d)\left(P_t\mathbb{1}_{\{0\}}\right)(1),\\ &= b\mathbb{P}_2(Y(t)=0) + d\mathbb{P}_0(Y(t)=0) - (b+d)q(t),\\ &= b(q(t))^2 + d - (b+d)q(t),\end{aligned}$$

  since each individual evolves independently.
  This gives the differential equation:

$$q'(t) = b(q(t))^2 + d - (b+d)q(t),$$

  which is also written when $b \neq b$:

$$\frac{q'(t)}{1-q(t)} - \frac{bq'(t)}{d-bq(t)} = d - b.$$

  This gives us:

$$\ln\left(\frac{1-q(0)}{1-q(t)}\right) + \ln\left(\frac{d-bq(t)}{d-bq(0)}\right) = (b-d)t,$$

  and as $q(0) = 0$, we have:

$$q(t) = \frac{d\left(1-e^{(d-b)t}\right)}{b-de^{(d-b)t}} = \frac{\left(\frac{1}{2}-\frac{a}{N}\right)\left(1-e^{\frac{-2a}{N}t}\right)}{\left(\frac{1}{2}+\frac{a}{N}\right)-\left(\frac{1}{2}-\frac{a}{N}\right)e^{\frac{-2a}{N}t}}.$$

  Therefore, we have:

$$\begin{aligned}f_{Nu} = 1 - q(Nu) &= 1 - \frac{\left(\frac{1}{2}-\frac{a}{N}\right)\left(1-e^{-2au}\right)}{\left(\frac{1}{2}+\frac{a}{N}\right)-\left(\frac{1}{2}-\frac{a}{N}\right)e^{-2au}},\\ &= 1 - \left(1-\frac{2a}{N}\right)\times\frac{1}{1+\frac{2a(1+e^{-2au})}{N(1-e^{-2au})}} = 1-\left(1-\frac{2a}{N}\right)\left(1-\frac{2a(1+e^{-2au})}{N(1-e^{-2au})}+\mathcal{O}\left(\frac{1}{N}\right)\right),\\ &= \frac{2a}{N} + \frac{2a(1+e^{-2au})}{N(1-e^{-2au})} + \mathcal{O}\left(\frac{1}{N}\right) = \frac{4a}{N(1-e^{-2au})} + \mathcal{O}\left(\frac{1}{N}\right).\end{aligned}$$

This gives:

$$f_{Nu} \underset{N}{\sim} \frac{4a}{N(1-e^{-2au})}.$$

Since this equivalent still is of the order of $\frac{1}{N}$, the proof of Proposition 4.6.2 of [Glö13] can be adapted and we proved that the family $\left(\#\mathfrak{A}^N_{t-2\varepsilon}(t)\right)_{N\in\mathbb{N}}$ is tight in the super-critical case.

Define

$$S^N_{2\varepsilon}(t) = \inf\left\{ \#\left\{\mathfrak{A}^N_{t-2\varepsilon}(\mathfrak{i},t) \mid \mathfrak{i}\in\mathcal{I}\right\} \mid \mathcal{I}\subseteq\mathfrak{I}^N_t \text{ with } \#\mathcal{I}\geq(1-2\varepsilon)\cdot\#\mathfrak{I}^N_t \right\}.$$

Thus, $S^N_{2\varepsilon}(t)$ is the size of the smallest sub-population of ancestors at time $t-2\varepsilon$ whose descendants make up at least a fraction of $1-2\varepsilon$ of the population $\mathfrak{I}^N_t$ at time t. By definition, we have:

$$S^N_{2\varepsilon}(t) \leq \#\mathfrak{A}^N_{t-2\varepsilon}(t),$$

and so the family $\left(S^N_{2\varepsilon}(t)\right)_{N\in\mathbb{N}}$ is tight for all $0<2\varepsilon\leq t\leq T$.

Proposition 4.5.6 of [Glö13] ensures that the probability that a sub-population of $\mathfrak{I}^N_t$ descending from a small sub-populations $\mathcal{I}\subset\mathfrak{I}^N_s$ becomes dominant at some later time is small. So, with a high probability, the relative share of sub-populations does not increase over time.

Applying Proposition 2.22 of [GPW13], the second point is proved.

✠

Lemma 2.7.6
Let $(\Omega,\mathcal{A},\mathbb{P})$ be a probability space. Let $A,B\in\mathcal{A}$ and $\varepsilon>0$. If $\mathbb{P}(A)>1-\varepsilon$ and $\mathbb{P}(B)>1-\varepsilon$, then $\mathbb{P}(A\cap B)\geq 1-3\varepsilon$.

In particular, considering a product space $(\Omega_1\times\Omega_2,\mathcal{A}_1\otimes\mathcal{A}_2,\mathbb{P})$ and $A_1\in\mathcal{A}_1$, $A_2\in\mathcal{A}_2$,

if $\mathbb{P}(A_1\times\Omega_2)\geq 1-\varepsilon$ and $\mathbb{P}(\Omega_1\times A_2)\geq 1-\varepsilon$, then $\mathbb{P}(A_1\times A_2)\geq 1-3\varepsilon$.

Proof of Lemma 2.7.6. We follow the proof of Lemma 4.5.2 of [Glö13]. We have:

$$\begin{aligned}\mathbb{P}(A\cap B) &= \mathbb{P}(A\cup B)-\mathbb{P}(A\backslash B)-\mathbb{P}(B\backslash A),\\ &\geq \mathbb{P}(A\cup B)-\mathbb{P}(\complement B)-\mathbb{P}(\complement A),\\ &\geq (1-\varepsilon)-\varepsilon-\varepsilon,\\ &\geq 1-3\varepsilon.\end{aligned}$$

The first statement is therefore proved. For the second, it is sufficient to note that:

$$A_1\times A_2 = (A_1\times\Omega_2)\cap(\Omega_1\times A_2),$$

and to apply the first result to $(A_1\times\Omega_2)\cap(\Omega_1\times A_2)$. ✠

7.2.2 - Weak tightness

In order to apply the Jakubowski's criterion we need to prove that for all $\Psi\in {}^{S}\Pi_\Psi(C_b^{1bc},C_K^\infty)$, the family $\left(\Psi\left(\mathfrak{T}^N_{\cdot,R}\right)\right)_{N\in\mathbb{N}}$ is tight for each $R\in\mathbb{N}^*$. However, as explained at the beginning of Section 5, Ψ–polynomials without replacement are the right class of functions to study the discrete tree-valued process. So, we first prove the tightness of $\left(\Psi^N(\mathfrak{T}^N_{\cdot,R})\right)_{N\in\mathbb{N}}$ for all $\Psi^N\in {}^{S}\Pi^N_\Psi(C_b^{1bc},C_K^\infty)$ and for each $R\in\mathbb{N}^*$ and extend the result to Ψ–polynomials (in order to have a class of functions that do not depend on N).

Proposition 2.7.7 (Weak Tightness)
Assume the four conditions of Proposition 2.7.1. Let $\Psi^N \in {}^S\Pi^N_\Psi(\mathcal{C}_b^{1bc}, \mathcal{C}_K^\infty)$.
Then, the family $\left(\Psi^N(\mathbb{T}^N_{\cdot,R})\right)_{N\in\mathbb{N}}$ is tight, for each $R \in \mathbb{N}$.

Proof of Proposition 2.7.7. The weak tightness is proved thanks to Aldous-Rebolledo criterion (Theorem 2.7.3).

Let $\Psi^{n,N}_{\omega,\psi} \in \Pi^N_\Psi(\mathcal{C}_b^{1bc}, \mathcal{C}_K^\infty)$. To simplify the notation, we denote $\Psi^{n,N}_{\omega,\psi}$ by Ψ^N. We prove the result only for Ψ^N, the extension to the linear span is immediate.

By Proposition 2.6.1,

$$M^N_{\Psi^N}(t) = \Psi^N(\mathbb{T}^N_t) - \Psi^N(\mathbb{T}^N_0) - \int_0^t \mathcal{G}^N\Psi^N(\mathbb{T}^N_s)ds,$$

is a martingale. The optional stopping theorem implies that

$$M^N_{\Psi^N,R}(t) = M^N_{\Psi^N}\left(\min(t, \tau^N_{1/R})\right) = \Psi^N(\mathbb{T}^N_{t,R}) - \Psi^N(\mathbb{T}^N_0) - \int_0^{\min(t,\tau^N_{1/R})} \mathcal{G}^N\Psi^N(\mathbb{T}^N_s)ds,$$

is a martingale too. Denoting

$$V^N_{\Psi^N,R}(t) = \int_0^{\min(t,\tau^N_{1/R})} \mathcal{G}^N\Psi^N(\mathbb{T}^N_s)ds,$$

we have the semi-martingale decomposition as in Proposition 2.6.5.

i - By Proposition 2.7.4, the family $\left(\mathbb{T}^N_\cdot\right)_{N\in\mathbb{N}}$ satisfies the compact containment condition. Therefore, the stopped process $\left(\mathbb{T}^N_{\cdot,R}\right)_{N\in\mathbb{N}}$ satisfies this condition too and, as Ψ^N is continuous, this is also true for the mapped process $\left(\Psi^N(\mathbb{T}^N_{\cdot,R})\right)_{N\in\mathbb{N}}$. So, for each $t \in \mathbb{R}_+$, $\left(\Psi^N(\mathbb{T}^N_{t,R})\right)_{N\in\mathbb{N}}$ is tight.

ii - Let $T > 0$, $(\tau_N)_{N\in\mathbb{N}}$ be a sequence of stopping times bounded by T and $\varepsilon > 0$.

(a) For every $N \in \mathbb{N}^*$ and for every $\theta \geq 0$, we have:

$$\mathbb{E}\left[\left|V^N_{\Psi^N,R}(\tau_N + \theta) - V^N_{\Psi^N,R}(\tau_N)\right|\right] \leq \mathbb{E}\left[\int_{\min(\tau_N,\tau^N_{1/R})}^{\min(\tau_N+\theta,\tau^N_{1/R})} \left|\mathcal{G}^N\Psi^N(\mathbb{T}^N_s)\right| ds\right],$$

and by Lemma 2.5.10, there exists two constants $C_{\mathcal{G},1} > 0$ and $C_{\mathcal{G},2} > 0$ such that:

$$\mathbb{E}\left[\left|V^N_{\Psi^N,R}(\tau_N + \theta) - V^N_{\Psi^N,R}(\tau_N)\right|\right] \leq C_{\mathcal{G},1}\mathbb{E}\left[\int_{\min(\tau_N,\tau^N_{1/R})}^{\min(\tau_N+\theta,\tau^N_{1/R})} \mathfrak{m}^N_s ds\right] + C_{\mathcal{G},2}\theta.$$

As we have seen in the proof of Proposition 2.7.5, the process $\left(\mathfrak{m}^N_t\right)_{t\in\mathbb{R}_+}$ is a classical birth-death process for which we know that $\left(\mathfrak{m}^N_t - 2a\beta t\right)_{t\in\mathbb{R}_+}$ is a martingale. This then allows us to control the term involving the integral of $\mathfrak{m}^N_s$ in the right-hand term of the inequality just above and thus to prove that there exists $\delta_1 > 0$ such that for every $\theta \in [0, \delta_1]$,

$$\mathbb{P}\left(\left|V^N_{\Psi^N,R}(\tau_N + \theta) - V^N_{\Psi^N,R}(\tau_N)\right| > \varepsilon\right) \leq \varepsilon.$$

(b) We do exactly the same for the condition ii-(b). For every $N \in \mathbb{N}^*$ and for every $\theta \geq 0$, we have thanks to Proposition 2.6.3:

$$\mathbb{E}\left[\left|\left\langle M^N_{\Psi^N,R}\right\rangle(\tau_n+\theta)-\left\langle M^N_{\Psi^N,R}\right\rangle(\tau_n)\right|\right] \leq \mathbb{E}\left[\int_{\min(\tau_N,\tau^N_{1/R})}^{\min(\tau_N+\theta,\tau^N_{1/R})}\left|g^N_{\Psi^N}(\mathfrak{T}^N_s)\right|ds\right],$$

and by Lemma 2.6.4, there exists two constant $C_{g,1} > 0$ and $C_{g,2} > 0$ such that:

$$\begin{aligned}\mathbb{E}\left[\left|\left\langle M^N_{\Psi^N,R}\right\rangle(\tau_n+\theta)-\left\langle M^N_{\Psi^N,R}\right\rangle(\tau_n)\right|\right] &\leq C_{g,1}\mathbb{E}\left[\int_{\min(\tau_N,\tau^N_{1/R})}^{\min(\tau_N+\theta,\tau^N_{1/R})}\mathfrak{m}^N_s\,ds\right]+C_{g,2}\theta,\\ &\leq \left(C_{g,1}M+C_{g,2}\right)\theta.\end{aligned}$$

Choosing $\delta_2 = \frac{\varepsilon}{C_{g,1}M+C_{g,2}} > 0$, we have for every $\theta \in [0,\delta_2]$,

$$\mathbb{P}\left(\left|\left\langle M^N_{\Psi^N,R}\right\rangle(\tau_n+\theta)-\left\langle M^N_{\Psi^N,R}\right\rangle(\tau_n)\right| > \varepsilon\right) \leq \varepsilon.$$

Finally, choosing $\delta = \min(\delta_1,\delta_2)$, the two inequalities are satisfied for every $\theta \in [0,\delta]$ and all conditions of the criterion are verified.

✠

Corollary 2.7.8
Assume the four conditions of Proposition 2.7.1. Let $\Psi \in {}^{S}\Pi_\Psi(C_b^{1bc}, C_K^\infty)$.
Then, the family $\left(\Psi(\mathfrak{T}^N_{\cdot,R})\right)_{N\in\mathbb{N}}$ is tight, for each $R \in \mathbb{N}^*$.

Proof of Corollary 2.7.8. We only consider $\Psi \in \Pi_\Psi(C_b^{1bc}, C_K^\infty)$ since the extension to the linear span is direct. Let $R \in \mathbb{N}^*$. We assume (without loss of generality) $N \geq 2R$ and $\mathfrak{m}^N_0 > \frac{1}{R}$. Therefore, as downward jumps of the total mass process are of size $\frac{1}{N}$, we have:

$$\mathfrak{m}^N_{\tau^N_{1/R}} \in \left]\frac{1}{R}-\frac{1}{N},\frac{1}{R}\right] \subseteq \left]\frac{1}{2R},\frac{1}{R}\right] \quad \text{and} \quad \forall t \geq 0,\ \mathfrak{m}^N_{t,R} \overset{def}{=} \mathfrak{m}^N_{\min(t,\tau^N_{1/R})} > \frac{1}{2R}.$$

For every $\psi \in C_K^\infty(\mathbb{R}_+)$, we can find $\psi_R \in C_K^\infty(\mathbb{R}_+)$ such that, $supp(\psi_R) \subset \left]\frac{1}{4R^2},+\infty\right[$ and $\psi_R = \psi\Big|_{\left[\frac{1}{2R},+\infty\right[}$. We then write:

$$\Psi_R = \Psi^n_{\omega,\psi_R} \quad \text{and} \quad \Psi^N_R = \Psi^{n,N}_{\omega,\psi_R}$$

By definition of $\mathfrak{T}^N_{\cdot,R}$, we also have:

$$\Psi\circ\mathfrak{T}^N_{\cdot,R} = \Psi_R\circ\mathfrak{T}^N_{\cdot,R} \quad \text{and} \quad \Psi^N\circ\mathfrak{T}^N_{\cdot,R} = \Psi^N_R\circ\mathfrak{T}^N_{\cdot,R}.$$

Proposition 2.7.7 ensures the tightness of $\left(\Psi^N\circ\mathfrak{T}^N_{\cdot,R}\right)_{N\in\mathbb{N}}$ and so the last equality in the above implies the tightness of $\left(\Psi^N_R\circ\mathfrak{T}^N_{\cdot,R}\right)_{N\in\mathbb{N}}$.

By Proposition 2.8.1, we have the convergence:

$$\lim_{N\longrightarrow+\infty}\sup\left\{\left|\Psi^N_R(\mathfrak{e})-\Psi_R(\mathfrak{e})\right| \mid \mathfrak{e}\in\mathbb{T}^N \text{ and } m_{\mathfrak{e}} \geq \frac{1}{4R^2}\right\} = 0.$$

Moreover, by definition of ψ_R we also have, for every $\mathfrak{e} \in \mathbb{T}^N$ such that $m_{\mathfrak{e}} < \frac{1}{4R^2}$,

$$\Psi_R^N(\mathfrak{e}) = 0 \text{ and } \Psi_R(\mathfrak{e}) = 0.$$

Hence, we have:

$$\lim_{N\to+\infty} \sup\left\{ \left|\Psi_R^N(\mathfrak{e}) - \Psi_R(\mathfrak{e})\right| \mid \mathfrak{e} \in \mathbb{T}^N \right\} = 0,$$

and in particular:

$$\lim_{N\to+\infty} \sup\left\{ \left|\Psi_R^N \circ \mathfrak{T}_{t,R}^N - \Psi_R \circ \mathfrak{T}_{t,R}^N\right| \mid t \in \mathbb{R}_+ \right\} = 0 \ a.s. \tag{2.44}$$

As Proposition 2.7.4 ensures that the family $\left(\mathfrak{T}_\cdot^N\right)_{N\in\mathbb{N}}$ satisfies the compact containment condition, we also know that $\left(\Psi_R \circ \mathfrak{T}_{\cdot,R}^N\right)_{N\in\mathbb{N}}$ verifies this condition since Ψ_R is continuous. So, to prove that $\left(\Psi_R \circ \mathfrak{T}_{\cdot,R}^N\right)_{N\in\mathbb{N}}$ is tight, we have to check that for every $T > 0$ and every $\varepsilon > 0$, there exists $\delta > 0$ such that:

$$\limsup_{N\to+\infty} \mathbb{P}\left(\sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R \circ \mathfrak{T}_{t,R}^N - \Psi_R \circ \mathfrak{T}_{s,R}^N\right| > \varepsilon \right) \le \varepsilon.$$

Let thus $T > 0$, $\varepsilon > 0$ and $\delta > 0$. We have for every $N \in \mathbb{N}$,

$$\begin{aligned}
&\mathbb{P}\left(\sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R \circ \mathfrak{T}_{t,R}^N - \Psi_R \circ \mathfrak{T}_{s,R}^N\right| > \varepsilon \right) \\
&\quad \le \mathbb{P}\left(\sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R \circ \mathfrak{T}_{t,R}^N - \Psi_R^N \circ \mathfrak{T}_{t,R}^N\right| + \sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R^N \circ \mathfrak{T}_{t,R}^N - \Psi_R^N \circ \mathfrak{T}_{s,R}^N\right| + \sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R \circ \mathfrak{T}_{s,R}^N - \Psi_R^N \circ \mathfrak{T}_{s,R}^N\right| > \varepsilon \right) \\
&\quad \le \mathbb{P}\left(\sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R^N \circ \mathfrak{T}_{t,R}^N - \Psi_R^N \circ \mathfrak{T}_{s,R}^N\right| + 2 \sup_{t\in[0,T]} \left|\Psi_R \circ \mathfrak{T}_{t,R}^N - \Psi_R^N \circ \mathfrak{T}_{t,R}^N\right| > \varepsilon \right)
\end{aligned}$$

Equation (2.44) ensures that there exists $N_0 \in \mathbb{N}$ such that, for every $N \ge N_0$,

$$\sup_{t\in[0,T]} \left|\Psi_R^N \circ \mathfrak{T}_{t,R}^N - \Psi_R \circ \mathfrak{T}_{t,R}^N\right| \le \frac{\varepsilon}{4} \ a.s.$$

Hence, for every $N \ge N_0$ we have:

$$\mathbb{P}\left(\sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R \circ \mathfrak{T}_{t,R}^N - \Psi_R \circ \mathfrak{T}_{s,R}^N\right| > \varepsilon \right) \le \mathbb{P}\left(\sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R^N \circ \mathfrak{T}_{t,R}^N - \Psi_R^N \circ \mathfrak{T}_{s,R}^N\right| > \frac{\varepsilon}{2} \right)$$

Moreover, as $\left(\Psi_R^N \circ \mathfrak{T}_{\cdot,R}^N\right)_{N\in\mathbb{N}}$ is tight, we have:

$$\limsup_{N\to+\infty} \mathbb{P}\left(\sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R^N \circ \mathfrak{T}_{t,R}^N - \Psi_R^N \circ \mathfrak{T}_{s,R}^N\right| > \frac{\varepsilon}{2} \right) \le \frac{\varepsilon}{2},$$

and so:

$$\limsup_{N\to+\infty} \mathbb{P}\left(\sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R \circ \mathfrak{T}_{t,R}^N - \Psi_R \circ \mathfrak{T}_{s,R}^N\right| > \varepsilon \right) \le \limsup_{N\to+\infty} \mathbb{P}\left(\sup_{\substack{0\le s<t\le T \\ |t-s|\le\delta}} \left|\Psi_R^N \circ \mathfrak{T}_{t,R}^N - \Psi_R^N \circ \mathfrak{T}_{s,R}^N\right| > \frac{\varepsilon}{2} \right) \le \frac{\varepsilon}{2} \le \varepsilon.$$

This implies the tightness of $\left(\Psi_R \circ \mathfrak{T}_{\cdot,R}^N\right)_{N\in\mathbb{N}}$. Finally, by definition of Ψ_R, we have the tightness for $\left(\Psi \circ \mathfrak{T}_{\cdot,R}^N\right)_{N\in\mathbb{N}}$.

8 - Convergence

Now that we have proved that the stopped process $\left(\mathfrak{T}^N_{\cdot,R}\right)_{N\in\mathbb{N}}$ is tight, Prokhorov's theorem ensures that every sequence contains a convergent subsequence. In this section, we find the potential limit of $\left(\mathfrak{T}^N_{\cdot,R}\right)_{N\in\mathbb{N}}$ by finding the limit of the infinitesimal generator of $\mathfrak{T}^N$. Finally, we prove that this accumulation point is unique and so, we prove the convergence in law of $\left(\mathfrak{T}^N_{\cdot,R}\right)_{N\in\mathbb{N}}$ in $D_{\mathbb{T}}[0,+\infty[$.

In this section, we start by giving some useful definitions and results to prove the convergence of the generators. Next, as the generator of $\mathfrak{T}^N$ corresponds to the sum of the growth generator and the binding generator, we successively prove the convergence of each of these two generators. Finally, we give the limit of the infinitesimal generator of $\mathfrak{T}^N$.

8.1 - Some Useful Definitions and Results

To prove the convergence of the generators, we need the total mass to remain above a certain threshold. We therefore start by defining restricted versions of $\mathbb{T}^N$ and $\mathbb{T}$ as well as a supremum norm with a constraint on the total mass.

For $\Gamma \subseteq \mathbb{R}_+$, denote:

$$\mathbb{T}^N(\Gamma) \overset{def}{=} \left\{ \mathfrak{e} \in \mathbb{T}^N \mid m_{\mathfrak{e}} \in \Gamma \right\},$$

and

$$\mathbb{T}(\Gamma) \overset{def}{=} \left\{ \mathfrak{e} \in \mathbb{T} \mid m_{\mathfrak{e}} \in \Gamma \right\}.$$

Moreover, for $f : F \subseteq \mathbb{T} \longrightarrow \mathbb{R}$ and $\delta > 0$, define:

$$\|f\|_{N,\delta} \overset{def}{=} \sup \left\{ |f(\mathfrak{e})| \mid \mathfrak{e} \in \mathbb{T}^N([\delta,+\infty[) \cap F \right\}.$$

The next proposition, which corresponds to Proposition 5.5.1 (i) of [Glö13], is very important and useful for what follow since it proves the convergence of a Ψ–polynomial without replacement to the corresponding Ψ–polynomials.

Proposition 2.8.1 (see [Glö13])
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_b(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\left\| \Psi^{n,N}_{\omega,\psi} - \Psi^n_{\omega,\psi} \right\|_{N,\delta} \underset{N\longrightarrow+\infty}{\longrightarrow} 0.$$

As explained in Section 5.2, a binding event implies a modification of the measure μ, and since we assume $E = supp(\mu)$, this implies a modification of E. Therefore, we need an operator to modify the distance matrix correspondingly.

Definition 2.8.2 (Replacement Operator)
For $i, j, k, l, n \in \mathbb{N}$, $n \geq 2$ and $1 \leq k < l \leq n$, the *replacement operator* acting on distance matrices

$r_{k,l} : \mathbb{U}^n \longrightarrow \mathbb{U}^n$ is defined by: for every $1 \leq i < j \leq n$ and for every $(u_{p,q})_{1\leq p<q\leq n} \in \mathbb{U}^n$,

$$\left(r_{k,l}\left(\left(u_{p,q}\right)_{1\leq p<q\leq n}\right)\right)_{i,j} \stackrel{def}{=} \begin{cases} u_{\min(k,j),\max(k,j)} & \text{if } i = l \text{ and } j \neq k, \\ 0 & \text{if } i = l \text{ and } j = k, \\ u_{\min(k,i),\max(k,i)} & \text{if } i \neq k \text{ and } j = l, \\ 0 & \text{if } i = k \text{ and } j = l, \\ u_{i,j} & \text{otherwise.} \end{cases} \tag{2.45}$$

⌞ ✤

Intuitively, applying $r_{k,l}$ to a distance matrix is equivalent to copying the k-th row (resp. column) of this matrix and pasting the copy into the l-th row (resp. column). Since individuals k and l coincide after this transformation, they are at null distance from each other, so

$$\left(r_{k,l}\left(\left(u_{p,q}\right)_{1\leq p<q\leq n}\right)\right)_{k,l} = \left(r_{k,l}\left(\left(u_{p,q}\right)_{1\leq p<q\leq n}\right)\right)_{l,k} = 0.$$

This transformation corresponds to what we did in Example 2.3.6 in the case of a branching event. In the case of fragmentation, we do not need to define a specific operator. Indeed, if a fragmentation occurs on the element $x \in E$, this element will lose a mass of $\frac{1}{N}$. After this transformation, either x keeps a non-zero mass and the distance matrix does not change. Or x has a zero mass and as $E = supp(\mu)$, x automatically disappears from the set E, which implies that the row/column corresponding to x automatically disappears from the distance matrix. Moreover, according to the definition of the lexicographic process (page 77), the indices of the leaves are not modified after a fragmentation, the index of the fragmenting leaf is simply removed from the space.

We also define the replacement operator acting on E^n, $\overline{r}_{k,l} : E^n \longrightarrow E^n$ by: for every $i \in \{1, \cdots, n\}$ and for every $(e_1, \cdots, e_n) \in E^n$,

$$\left(\overline{r}_{k,l}(e_1, \cdots, e_n)\right)_i = \begin{cases} e_k & \text{if } i = l, \\ e_i & \text{if } i \neq l. \end{cases} \tag{2.46}$$

Note that:

$$U^n_{(E,d)} \circ \overline{r}_{i,j} = r_{i,j} \circ U^n_{(E,d)} \tag{2.47}$$

Finally, we introduce some notation and a lemma concerning sampling with or without replacement.

Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N$, $n \in \mathbb{N}$, $N \geq 2$, $\mu_0 \in \mathcal{M}_f(E)$, $n_0 \in \{1, \cdots, n\}$ and $1 \leq l_1 < \cdots < l_{n_0} \leq n$. For all $(e_1, \cdots, e_n) \in E^n$, we define for every $i \in \{1, \cdots, n\}$ measures μ_i and μ_i^N by:

$$\mu_i(\{e_i\}) \stackrel{def}{=} \begin{cases} \mu(\{e_i\}) & \text{if } i \in \{1, \cdots, n\} \backslash \{l_1, \cdots, l_{n_0}\}, \\ \mu_0(\{e_i\}) & \text{if } i \in \{l_1, \cdots, l_{n_0}\}. \end{cases}$$

And

$$\mu_i^N(\{e_i\}) \stackrel{def}{=} \begin{cases} \mu(\{e_i\}) - \frac{1}{N}\sum_{j=1}^{i-1} \delta_{e_j}(\{e_i\}) & \text{if } i \in \{1, \cdots, n\} \backslash \{l_1, \cdots, l_{n_0}\}, \\ \mu_0(\{e_i\}) & \text{if } i \in \{l_1, \cdots, l_{n_0}\}. \end{cases}$$

We also denote:

$$\bigotimes_{(m_1,\cdots,m_{n_0})}^{n} (\mu, \mu_0) \stackrel{def}{=} \bigotimes_{i=1}^{n} \mu_i \quad \text{and} \quad \bigotimes_{(m_1,\cdots,m_{n_0})}^{n,N} (\mu, \mu_0) \stackrel{def}{=} \bigotimes_{i=1}^{n} \mu_i^N.$$

These two measures allow to apply the measure μ_0 to the elements $(m_1, \cdots, m_{n_0})$ and the measure μ (with or without replacement) to the other elements. Moreover, if μ_0 is such that $\mu_0(\{e\}) = 1$ for all $e \in E$, we simply write:

$$\bigotimes_{(m_1,\cdots,m_{n_0})}^{n}(\mu) \quad \text{and} \quad \bigotimes_{(m_1,\cdots,m_{n_0})}^{n,N}(\mu).$$

The following lemma correspond to Lemma 5.5.4 of [Glö13] and will be very useful for us to find the limit of the binding generator.

Lemma 2.8.3 (see [Glö13])
Let $\delta > 0$, $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[)$, $n \in \mathbb{N}$, $n \geq 2$, $x \in E$. Moreover, assume that $N \in \mathbb{N}$ is large enough to ensure that $\delta - \frac{n}{N} > 0$. Then,

$$\left(\frac{\mu \pm \frac{1}{N}\delta_x}{m_{\mathfrak{e}} \pm \frac{1}{N}}\right)^{\otimes n,N} = \frac{1}{\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n,N}}\left(\mu^{\otimes n,N} + \sum_{n_0=1}^{n}\sum_{1\leq l_1<\cdots<l_{n_0}\leq n}\bigotimes_{(l_1,\cdots,l_{n_0})}^{n,N}\left(\mu, \pm\frac{1}{N}\delta_x\right)\right).$$

This lemma is quite intuitive if we make the analogy with a product. Indeed, by observing that $(a+b)^n = a^n + \sum_{n_0=1}^{n}\binom{n}{n_0}a^{n-n_0}b^{n_0}$, we can easily identify the role of each term of the right-hand side term of the equality of Lemma 2.8.3. As the term a^n is analogous to the term $\mu^{\otimes n,N}$, which means applying the measure μ without replacement to the n elements of the sample, the term $a^{n-n_0}b^{n_0}$ means applying the measure $\pm\frac{1}{N}\delta_x$ to n_0 elements of the sample and μ to the remaining $n - n_0$ elements; this is exactly what the measure $\bigotimes_{(l_1,\cdots,l_{n_0})}^{n,N}\left(\mu, \pm\frac{1}{N}\delta_x\right)$ allows to do.

We now have all the elements to find the limit and prove the convergence of the growth generator and the binding generator.

8.2 - Limit of the Growth Generator

Definition 2.8.4 (Limiting Growth Generator)
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}$, $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b^{1bc}(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_b(\mathbb{R}_+)$. The generator $\mathcal{G}^{\infty}_{grow}$ applied to the Ψ-polynomial $\Psi^n_{\omega,\psi}$ is given by:

$$\mathcal{G}^{\infty}_{grow}\Psi^n_{\omega,\psi}(\mathfrak{e}) = \Psi^n_{2\lambda^+\overline{\nabla}\omega,\psi}(\mathfrak{e}), \tag{2.48}$$

with $\overline{\nabla}\omega$ defined in (2.36).

The next proposition ensures that the generator $\mathcal{G}^{\infty}_{grow}$ is the limit of the growth generator $\mathcal{G}^{N}_{grow}$ when N goes to infinity.

Proposition 2.8.5 (Convergence of the Growth Generator)
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b^{1bc}(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_b(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\left\|\mathcal{G}^{N}_{grow}\Psi^{n,N}_{\omega,\psi} - \mathcal{G}^{\infty}_{grow}\Psi^{n}_{\omega,\psi}\right\|_{N,\delta} \underset{N\to+\infty}{\longrightarrow} 0.$$

Proof of Proposition 2.8.5. Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[)$. Using Lemma 2.5.4, we have:

$$\mathcal{G}^{N}_{grow}\Psi^{n,N}_{\omega,\psi}(\mathfrak{e}) = \Psi^{n,N}_{2\lambda^+\overline{\nabla}\omega,\psi}(\mathfrak{e}) + \mathcal{O}\left(\frac{1}{N}\right), \tag{2.35}$$

where the error term is uniform in $\mathfrak{e} \in \mathbb{T}^N$ $([\delta, +\infty[)$.

Since $\omega \in \mathcal{C}_b^{1bc}(\mathbb{U}^n)$, $2\lambda^+\overline{\nabla}\omega \in C_b(\mathbb{U}^n)$, Proposition 2.8.1 gives:

$$\left\| \Psi^{n,N}_{2\lambda^+\overline{\nabla}\omega,\psi} - \Psi^{n}_{2\lambda^+\overline{\nabla}\omega,\psi} \right\|_{N,\delta} \underset{N\to+\infty}{\longrightarrow} 0.$$

8.3 - Limit of the Binding Generator

Definition 2.8.6 (Limiting Binding Generator)
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}$, $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. The generator $\mathcal{G}^\infty_{bind}$ applied to the Ψ-polynomial $\Psi^n_{\omega,\psi}$ is given by:

$$\mathcal{G}^\infty_{bind}\Psi^n_{\omega,\psi}(\mathfrak{e}) = \left(2a\beta m_{\mathfrak{e}}\psi'(m_{\mathfrak{e}}) + \frac{\beta m_{\mathfrak{e}}}{2}\psi''(m_{\mathfrak{e}})\right)\Omega^n_\omega(\hat{\mathfrak{e}}) + \frac{\beta\psi(m_{\mathfrak{e}})}{m_{\mathfrak{e}}}\sum_{1\leq i<j\leq n}\left(\left\langle \omega\circ r_{i,j}, \nu^n_{\hat{\mathfrak{e}}}\right\rangle - \left\langle \omega, \nu^n_{\hat{\mathfrak{e}}}\right\rangle\right), \tag{2.49}$$

where $r_{i,j}$ is the replacement operator acting on distance matrices defined in (2.45), $\hat{\mathfrak{e}} = [E, d, \hat{\mu}]$ with $\hat{\mu}$ the normalised measure defined in (2.20) and where $\nu^n_{\hat{\mathfrak{e}}}$ is the distance matrix distribution for a random sample of size n of $\hat{\mathfrak{e}}$ defined in (2.18).

Remark 2.8.7
In the generator $\mathcal{G}^\infty_{bind}\Psi^n_{\omega,\psi}(\mathfrak{e})$, the term $\left(2a\beta m_{\mathfrak{e}}\psi'(m_{\mathfrak{e}}) + \frac{\beta m_{\mathfrak{e}}}{2}\psi''(m_{\mathfrak{e}})\right)\Omega^n_\omega(\hat{\mathfrak{e}})$ corresponds to the transformation of the total mass (and thus of the number of leaves in the tree) by fragmentations and branching events. The term $\frac{\beta\psi(m_{\mathfrak{e}})}{m_{\mathfrak{e}}}\sum_{1\leq i<j\leq n}\left(\left\langle \omega\circ r_{i,j}, \nu^n_{\hat{\mathfrak{e}}}\right\rangle - \left\langle \omega, \nu^n_{\hat{\mathfrak{e}}}\right\rangle\right)$ corresponds to the transformation of the tree structure (indexing of the leaves) by branching events. As explained after Definition 2.8.2, fragmentation is not involved in this term since tree transformations caused by fragmentation are automatically taken into account by the definition of the space $\mathfrak{e}$.

The next proposition ensures that the generator $\mathcal{G}^\infty_{bind}$ is the limit of the binding generator $\mathcal{G}^N_{bind}$ when N goes to infinity.

Proposition 2.8.8 (Convergence of the Binding Generator)
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\left\| \mathcal{G}^N_{bind}\Psi^{n,N}_{\omega,\psi} - \mathcal{G}^\infty_{bind}\Psi^n_{\omega,\psi} \right\|_{N,\delta} \underset{N\to+\infty}{\longrightarrow} 0.$$

Proof of Proposition 2.8.8. Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[)$. Using Lemma 2.5.7, we have:

$$\mathcal{G}^N_{bind}\Psi^{n,N}_{\omega,\psi}(\mathfrak{e}) = \left(2a\beta m_{\mathfrak{e}}\psi'(m_{\mathfrak{e}}) + \frac{\beta m_{\mathfrak{e}}}{2}\psi''(m_{\mathfrak{e}})\right)\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle \tag{2.41}$$

$$+ \left(N\beta p_0^N\psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right) \times \sum_{x\in E}\left(\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu - \frac{1}{N}\delta_x}{m_{\mathfrak{e}} - \frac{1}{N}}\right)^{\otimes n,N}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle\right) \tag{2.42}$$

$$+ \left(N\beta p_2^N\psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right) \times \sum_{x\in E}\left(\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu + \frac{1}{N}\delta_x}{m_{\mathfrak{e}} + \frac{1}{N}}\right)^{\otimes n,N}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle\right) \tag{2.43}$$

$$+ \mathcal{O}\left(\frac{1}{N}\right).$$

Lemma 2.8.9 ensures convergence of the term (2.41) to:

$$\left(2a\beta m_{\mathfrak{e}}\psi'(m_{\mathfrak{e}}) + \frac{\beta m_{\mathfrak{e}}}{2}\psi''(m_{\mathfrak{e}})\right)\left\langle \omega\circ U^n_{(E,d)}, \hat{\mu}^{\otimes n}\right\rangle = \left(2a\beta m_{\mathfrak{e}}\psi'(m_{\mathfrak{e}}) + \frac{\beta m_{\mathfrak{e}}}{2}\psi''(m_{\mathfrak{e}})\right)\Omega^n_\omega(\hat{\mathfrak{e}}).$$

Lemma 2.8.10 ensures that the sum of the terms (2.43) and (2.42) converges to:

$$\frac{\beta\psi(m_{\mathfrak{e}})}{m_{\mathfrak{e}}}\sum_{1\le l_1<l_2\le n}\left(\left\langle \omega\circ r_{l_1,l_2}\circ U^n_{(E,d)}, \hat{\mu}^{\otimes n}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \hat{\mu}^{\otimes n}\right\rangle\right),$$

which is also written (using (2.18)).

$$\frac{\beta\psi(m_{\mathfrak{e}})}{m_{\mathfrak{e}}}\sum_{1\le i<j\le n}\left(\left\langle \omega\circ r_{i,j}, \nu^n_{\hat{\mathfrak{e}}}\right\rangle - \left\langle \omega, \nu^n_{\hat{\mathfrak{e}}}\right\rangle\right).$$

Lemma 2.8.9

Let $n \in \mathbb{N}$, $n \ge 2$, $\omega \in C_b(\mathbb{U}^n)$ and $\psi \in C^\infty_K(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[)}\left|\left(2a\beta m_{\mathfrak{e}}\psi'(m_{\mathfrak{e}}) + \frac{\beta m_{\mathfrak{e}}}{2}\psi''(m_{\mathfrak{e}})\right)\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle - \left(2a\beta m_{\mathfrak{e}}\psi'(m_{\mathfrak{e}}) + \frac{\beta m_{\mathfrak{e}}}{2}\psi''(m_{\mathfrak{e}})\right)\left\langle \omega\circ U^n_{(E,d)}, \hat{\mu}^{\otimes n}\right\rangle\right| \underset{N\to+\infty}{\longrightarrow} 0.$$

Proof of Lemma 2.8.9. We follow the proof of Lemma 5.5.7 of [Glö13].

Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[)$. Using Lemme 2.5.9, we have:

$$\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle = \left\langle \omega\circ U^n_{(E,d)}, \hat{\mu}^{\otimes n,N}\right\rangle = \left\langle \omega\circ U^n_{(E,d)}, \widehat{\mu^{\otimes n,N}}\right\rangle = \Omega^{n,N}_\omega(\mathfrak{e}).$$

Moreover, the function ψ_1 defined on $\mathbb{R}_+$ by:

$$\psi_1(x) = 2a\beta x\psi'(x) + \frac{\beta x}{2}\psi''(x)$$

is a bounded continuous function since $\psi \in C_K^\infty$. So, by definition of Ψ–polynomials without replacement, we have:

$$\left(2a\beta m_{\mathfrak{e}}\psi'(m_{\mathfrak{e}}) + \frac{\beta m_{\mathfrak{e}}}{2}\psi''(m_{\mathfrak{e}})\right)\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle = \psi_1(m_{\mathfrak{e}})\Omega^{n,N}_{\omega}(\mathfrak{e}) = \Psi^{n,N}_{\omega,\psi_1}(\mathfrak{e}).$$

Finally, Proposition 2.8.1 gives:

$$\left\|\Psi^{n,N}_{\omega,\psi_1} - \Psi^{n}_{\omega,\psi_1}\right\|_{N,\delta} \underset{N\to+\infty}{\longrightarrow} 0.$$

Lemma 2.8.10
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\begin{aligned}
\sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[)} \Bigg| & \left(N\beta p_0^N\psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right) \\
& \times \sum_{x\in E}\left(\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu - \frac{1}{N}\delta_x}{m_{\mathfrak{e}} - \frac{1}{N}}\right)^{\otimes n,N}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle\right) \\
& + \left(N\beta p_2^N\psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right) \\
& \times \sum_{x\in E}\left(\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu + \frac{1}{N}\delta_x}{m_{\mathfrak{e}} + \frac{1}{N}}\right)^{\otimes n,N}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle\right) \\
& - \frac{\beta\psi(m_{\mathfrak{e}})}{m_{\mathfrak{e}}}\sum_{1\leq l_1<l_2\leq n}\left(\left\langle \omega\circ r_{l_1,l_2}\circ U^n_{(E,d)}, \hat{\mu}^{\otimes n}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \hat{\mu}^{\otimes n}\right\rangle\right)\Bigg| \underset{N\to+\infty}{\longrightarrow} 0.
\end{aligned}$$

Note that because of the presence of factor $\psi(m_{\mathfrak{e}})$ (or $\psi'(m_{\mathfrak{e}})$) in each of the three terms, we can replace the supremum over $\mathfrak{e} \in \mathbb{T}^N([\delta,+\infty[)$ by the supremum over $\mathfrak{e} \in \mathbb{T}^N([\delta,+\infty[\cap supp(\psi))$. This is important since $supp(\psi)$ is assumed to be compact.

Proof of Lemma 2.8.10. We follow the proof of Lemma 5.5.8 of [Glö13].

Assume $\mathfrak{e} = [E,d,\mu] \in \mathbb{T}^N([\delta,+\infty[)$. Using Lemme 2.5.9, we have:

$$\left\langle \omega\circ U^n_{(E,d)}, \left(\frac{\mu}{m_{\mathfrak{e}}}\right)^{\otimes n,N}\right\rangle = \left\langle \omega\circ U^n_{(E,d)}, \hat{\mu}^{\otimes n,N}\right\rangle = \left\langle \omega\circ U^n_{(E,d)}, \widehat{\mu^{\otimes n,N}}\right\rangle = \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{(m_{\mathfrak{e}})^{n,N}}\mu^{\otimes n,N}\right\rangle.$$

Moreover, using Lemma 2.8.3, we have:

$$\begin{aligned}
\left(\frac{\mu \pm \frac{1}{N}\delta_x}{m_{\mathfrak{e}} \pm \frac{1}{N}}\right)^{\otimes n,N} = & \frac{1}{\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n,N}}\mu^{\otimes n,N} + \frac{1}{\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n,N}}\sum_{l_1=1}^{n}\bigotimes_{(l_1)}^{n,N}\left(\mu, \pm\frac{1}{N}\delta_x\right) \\
& + \frac{1}{\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n,N}}\sum_{1\leq l_1<l_2\leq n}\bigotimes_{(l_1,l_2)}^{n,N}\left(\mu, \pm\frac{1}{N}\delta_x\right) + \epsilon_{n,N},
\end{aligned}$$

where $\epsilon_{n,N} = \frac{1}{\left(m_\mathfrak{e} \pm \frac{1}{N}\right)^{n,N}} \sum_{n_0=3}^{n} \sum_{1 \le l_1 < \cdots < l_{n_0} \le n} \bigotimes_{(l_1,\cdots,l_{n_0})}^{n,N} \left(\mu, \pm \frac{1}{N}\delta_x\right)$. So, we have:

$$
\begin{aligned}
&\left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left(\left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu - \frac{1}{N}\delta_x}{m_\mathfrak{e} - \frac{1}{N}}\right)^{\otimes n,N} \right\rangle - \left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu}{m_\mathfrak{e}}\right)^{\otimes n,N} \right\rangle \right) \\
&\quad + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left(\left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu + \frac{1}{N}\delta_x}{m_\mathfrak{e} + \frac{1}{N}}\right)^{\otimes n,N} \right\rangle - \left\langle \omega \circ U^n_{(E,d)}, \left(\frac{\mu}{m_\mathfrak{e}}\right)^{\otimes n,N} \right\rangle \right) \\
&= \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left(\left\langle \omega \circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} \mu^{\otimes n,N} \right\rangle - \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{(m_\mathfrak{e})^{n,N}} \mu^{\otimes n,N} \right\rangle \right) \\
&\quad + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left(\left\langle \omega \circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} \mu^{\otimes n,N} \right\rangle - \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{(m_\mathfrak{e})^{n,N}} \mu^{\otimes n,N} \right\rangle \right) \\
&+ \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} \left(\mu, -\frac{1}{N}\delta_x\right) \right\rangle \\
&\quad + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} \left(\mu, \frac{1}{N}\delta_x\right) \right\rangle \\
&+ \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} \sum_{1 \le l_1 < l_2 \le n} \bigotimes_{(l_1,l_2)}^{n,N} \left(\mu, -\frac{1}{N}\delta_x\right) \right\rangle \\
&\quad + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} \sum_{1 \le l_1 < l_2 \le n} \bigotimes_{(l_1,l_2)}^{n,N} \left(\mu, \frac{1}{N}\delta_x\right) \right\rangle + \epsilon'_{n,N},
\end{aligned}
$$

where $\epsilon'_{n,N}$ corresponds to terms that result from the term $\epsilon_{n,N}$ of the measure (terms of order greater than or equal to 3).

Lemma 2.8.11 ensures convergence of the term of order 0,

$$
\begin{aligned}
&\left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left(\left\langle \omega \circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} \mu^{\otimes n,N} \right\rangle - \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{(m_\mathfrak{e})^{n,N}} \mu^{\otimes n,N} \right\rangle \right) \\
&\quad + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left(\left\langle \omega \circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} \mu^{\otimes n,N} \right\rangle - \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{(m_\mathfrak{e})^{n,N}} \mu^{\otimes n,N} \right\rangle \right)
\end{aligned}
$$

to $\left(-\frac{n}{m_\mathfrak{e}^n}\left(2a\beta\psi(m_\mathfrak{e}) + \beta\psi'(m_\mathfrak{e})\right) + \frac{n(n+1)}{2m_\mathfrak{e}^{n+1}}\beta\psi(m_\mathfrak{e})\right) \left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n} \right\rangle$.

Lemma 2.8.15 ensures convergence of the term of order 1,

$$
\begin{aligned}
&\left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} \left(\mu, -\frac{1}{N}\delta_x\right) \right\rangle \\
&\qquad + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} \left(\mu, \frac{1}{N}\delta_x\right) \right\rangle
\end{aligned}
$$

to $\left(\frac{n}{m_\mathfrak{e}^n}\left(2a\beta\psi(m_\mathfrak{e}) + \beta\psi'(m_\mathfrak{e})\right) - \frac{n^2}{m_\mathfrak{e}^{n+1}}\beta\psi(m_\mathfrak{e})\right) \left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n} \right\rangle$.

Lemma 2.8.20 ensures convergence of the term of order 2,

$$\left(N\beta p_0^N\psi(m_\epsilon)-\frac{\beta}{2}\psi'(m_\epsilon)\right)\sum_{x\in E}\left\langle\omega\circ U^n_{(E,d)},\frac{1}{\left(m_\epsilon-\frac{1}{N}\right)^{n,N}}\sum_{1\le l_1<l_2\le n}\bigotimes_{(l_1,l_2)}^{n,N}\left(\mu,-\frac{1}{N}\delta_x\right)\right\rangle$$
$$+\left(N\beta p_2^N\psi(m_\epsilon)+\frac{\beta}{2}\psi'(m_\epsilon)\right)\sum_{x\in E}\left\langle\omega\circ U^n_{(E,d)},\frac{1}{\left(m_\epsilon+\frac{1}{N}\right)^{n,N}}\sum_{1\le l_1<l_2\le n}\bigotimes_{(l_1,l_2)}^{n,N}\left(\mu,\frac{1}{N}\delta_x\right)\right\rangle$$

to $\frac{1}{m_\epsilon^{n+1}}\beta\psi(m_\epsilon)\sum_{1\le l_1<l_2\le n}\left\langle\omega\circ r_{l_1,l_2}\circ U^n_{(E,d)},\mu^{\otimes n}\right\rangle$.

Finally, Lemma 5.5.21 of [Glö13] ensures convergence of terms of order greater than or equal to 3 to 0. Indeed, even if our generators are not identical, all the arguments of the proof remain valid since the terms introduced by the term $\pm\frac{a}{N}$ in the probability distribution $\boldsymbol{p}^{\boldsymbol{N}}$ vanish in the limit due to the integration with respect to the term $\epsilon_{n,N}$ of the measure.

Therefore, the limit is written:

$$\begin{aligned}
&\Bigg(-\frac{n}{m_\epsilon^n}\left(2a\beta\psi(m_\epsilon)+\beta\psi'(m_\epsilon)\right)+\frac{n(n+1)}{2m_\epsilon^{n+1}}\beta\psi(m_\epsilon)\\
&\qquad+\frac{n}{m_\epsilon^n}\left(2a\beta\psi(m_\epsilon)+\beta\psi'(m_\epsilon)\right)-\frac{n^2}{m_\epsilon^{n+1}}\beta\psi(m_\epsilon)\Bigg)\left\langle\omega\circ U^n_{(E,d)},\mu^{\otimes n}\right\rangle\\
&\qquad+\frac{1}{m_\epsilon^{n+1}}\beta\psi(m_\epsilon)\sum_{1\le l_1<l_2\le n}\left\langle\omega\circ r_{l_1,l_2}\circ U^n_{(E,d)},\mu^{\otimes n}\right\rangle\\
&=\left(\frac{n^2}{2m_\epsilon}\beta\psi(m_\epsilon)+\frac{n}{2m_\epsilon}\beta\psi(m_\epsilon)-\frac{n^2}{m_\epsilon}\beta\psi(m_\epsilon)\right)\left\langle\omega\circ U^n_{(E,d)},\hat\mu^{\otimes n}\right\rangle\\
&\qquad+\frac{1}{m_\epsilon}\beta\psi(m_\epsilon)\sum_{1\le l_1<l_2\le n}\left\langle\omega\circ r_{l_1,l_2}\circ U^n_{(E,d)},\hat\mu^{\otimes n}\right\rangle,\\
&=-\frac{1}{m_\epsilon}\beta\psi(m_\epsilon)\left(\frac{n^2}{2}-\frac{n}{2}\right)\left\langle\omega\circ U^n_{(E,d)},\hat\mu^{\otimes n}\right\rangle+\frac{1}{m_\epsilon}\beta\psi(m_\epsilon)\sum_{1\le l_1<l_2\le n}\left\langle\omega\circ r_{l_1,l_2}\circ U^n_{(E,d)},\hat\mu^{\otimes n}\right\rangle,\\
&=-\frac{1}{m_\epsilon}\beta\psi(m_\epsilon)\binom{n}{2}\left\langle\omega\circ U^n_{(E,d)},\hat\mu^{\otimes n}\right\rangle+\frac{1}{m_\epsilon}\beta\psi(m_\epsilon)\sum_{1\le l_1<l_2\le n}\left\langle\omega\circ r_{l_1,l_2}\circ U^n_{(E,d)},\hat\mu^{\otimes n}\right\rangle,\\
&=\frac{\beta\psi(m_\epsilon)}{m_\epsilon}\sum_{1\le l_1<l_2\le n}\left(\left\langle\omega\circ r_{l_1,l_2}\circ U^n_{(E,d)},\hat\mu^{\otimes n}\right\rangle-\left\langle\omega\circ U^n_{(E,d)},\hat\mu^{\otimes n}\right\rangle\right).
\end{aligned}$$

Lemma 2.8.11
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\begin{aligned}\sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \Bigg| & \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \\ & \times \sum_{x\in E}\left(\left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}}\mu^{\otimes n,N}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{(m_\mathfrak{e})^{n,N}}\mu^{\otimes n,N}\right\rangle\right) \\ & + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \\ & \times \sum_{x\in E}\left(\left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}}\mu^{\otimes n,N}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{(m_\mathfrak{e})^{n,N}}\mu^{\otimes n,N}\right\rangle\right) \\ & - \left(-\frac{n}{m_\mathfrak{e}^n}\left(2a\beta\psi(m_\mathfrak{e}) + \beta\psi'(m_\mathfrak{e})\right) + \frac{n(n+1)}{2m_\mathfrak{e}^{n+1}}\beta\psi(m_\mathfrak{e})\right)\left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle \Bigg| \underset{N\to+\infty}{\longrightarrow} 0.\end{aligned}$$

Proof of Lemma 2.8.11. We follow the proof of Lemma 5.5.9 of [Glö13].

Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))$, since $\#E = N m_\mathfrak{e}$, we have:

$$\begin{aligned}& \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\sum_{x\in E}\left(\left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}}\mu^{\otimes n,N}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{(m_\mathfrak{e})^{n,N}}\mu^{\otimes n,N}\right\rangle\right) \\ & + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\sum_{x\in E}\left(\left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}}\mu^{\otimes n,N}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{(m_\mathfrak{e})^{n,N}}\mu^{\otimes n,N}\right\rangle\right) \\ & = \left[N m_\mathfrak{e}\left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} - \frac{1}{(m_\mathfrak{e})^{n,N}}\right)\right. \\ & \left. + N m_\mathfrak{e}\left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(\frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} - \frac{1}{(m_\mathfrak{e})^{n,N}}\right)\right]\left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n,N}\right\rangle.\end{aligned}$$

Moreover, we have:

$$\left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n,N}\right\rangle = \left(\left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n,N}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle\right) + \left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle,$$

and

$$\begin{aligned}& \frac{1}{\left(m_\mathfrak{e} \pm \frac{1}{N}\right)^{n,N}} - \frac{1}{(m_\mathfrak{e})^{n,N}} \\ & \qquad = \left(\frac{1}{\left(m_\mathfrak{e} \pm \frac{1}{N}\right)^{n}} - \frac{1}{m_\mathfrak{e}^n}\right) + \left(\left(\frac{1}{\left(m_\mathfrak{e} \pm \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_\mathfrak{e} \pm \frac{1}{N}\right)^{n}}\right) - \left(\frac{1}{(m_\mathfrak{e})^{n,N}} - \frac{1}{m_\mathfrak{e}^n}\right)\right).\end{aligned}$$

Lemma 2.8.12 (which corresponds to Lemma 5.5.10 of [Glö13]) ensures the convergence of the term

$$\begin{aligned}& \left[N m_\mathfrak{e}\left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} - \frac{1}{(m_\mathfrak{e})^{n,N}}\right)\right. \\ & \left. + N m_\mathfrak{e}\left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(\frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} - \frac{1}{(m_\mathfrak{e})^{n,N}}\right)\right]\left(\left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n,N}\right\rangle - \left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle\right)\end{aligned}$$

to 0.

Lemma 2.8.13 ensures the convergence of the term

$$\left[N m_{\mathfrak{e}} \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^n} - \frac{1}{m_{\mathfrak{e}}^n} \right) + N m_{\mathfrak{e}} \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^n} - \frac{1}{m_{\mathfrak{e}}^n} \right) \right] \left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n} \right\rangle$$

to $\left(-\frac{n}{m_{\mathfrak{e}}^n} \left(2a\beta\psi(m_{\mathfrak{e}}) + \beta\psi'(m_{\mathfrak{e}}) \right) + \frac{n(n+1)}{2m_{\mathfrak{e}}^{n+1}} \beta\psi(m_{\mathfrak{e}}) \right) \left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n} \right\rangle$.

Lemma 2.8.14 ensures the convergence of the term

$$\left[N m_{\mathfrak{e}} \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \times \left(\left(\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^n} \right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^n} \right) \right) + N m_{\mathfrak{e}} \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \times \left(\left(\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^n} \right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^n} \right) \right) \right] \left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n} \right\rangle$$

to 0.

Therefore, the next three lemmas allow us to finish this proof. ✻

Lemma 2.8.12 (see [Glö13])
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\sup_{\mathfrak{e} \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))} \left| \left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n,N} \right\rangle - \left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n} \right\rangle \right| \underset{N \to +\infty}{\longrightarrow} 0.$$

Lemma 2.8.13
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\sup_{\mathfrak{e} \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))} \left| N m_{\mathfrak{e}} \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^n} - \frac{1}{m_{\mathfrak{e}}^n} \right) + N m_{\mathfrak{e}} \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^n} - \frac{1}{m_{\mathfrak{e}}^n} \right) - \left(-\frac{n}{m_{\mathfrak{e}}^n} \left(2a\beta\psi(m_{\mathfrak{e}}) + \beta\psi'(m_{\mathfrak{e}}) \right) + \frac{n(n+1)}{2m_{\mathfrak{e}}^{n+1}} \beta\psi(m_{\mathfrak{e}}) \right) \right| \underset{N \to +\infty}{\longrightarrow} 0.$$

Proof of Lemma 2.8.13. We follow the proof of Lemma 5.5.11 of [Glö13].

Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))$. A second order Taylor expansion of $\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^n}$ and of $\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^n}$ gives:

$$
\begin{aligned}
& N m_{\mathfrak{e}} \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^n} - \frac{1}{m_{\mathfrak{e}}^n} \right) \\
& \qquad + N m_{\mathfrak{e}} \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^n} - \frac{1}{m_{\mathfrak{e}}^n} \right) \\
& = N m_{\mathfrak{e}} \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\frac{n}{N m_{\mathfrak{e}}^{n+1}} + \frac{n(n+1)}{2N^2 m_{\mathfrak{e}}^{n+2}} + \mathcal{O}\left(\frac{1}{N^3}\right) \right) \\
& \qquad + N m_{\mathfrak{e}} \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(-\frac{n}{N m_{\mathfrak{e}}^{n+1}} + \frac{n(n+1)}{2N^2 m_{\mathfrak{e}}^{n+2}} + \mathcal{O}\left(\frac{1}{N^3}\right) \right), \\
& = \frac{n}{m_{\mathfrak{e}}^n} \left(N\beta(p_0^N - p_2^N)\psi(m_{\mathfrak{e}}) - \beta\psi'(m_{\mathfrak{e}}) \right) + \frac{n(n+1)}{2m_{\mathfrak{e}}^{n+1}} \beta(p_0^N + p_2^N)\psi(m_{\mathfrak{e}}) + \mathcal{O}\left(\frac{1}{N}\right), \\
& = \frac{n}{m_{\mathfrak{e}}^n} \left(-2a\beta\psi(m_{\mathfrak{e}}) - \beta\psi'(m_{\mathfrak{e}}) \right) + \frac{n(n+1)}{2m_{\mathfrak{e}}^{n+1}} \beta\psi(m_{\mathfrak{e}}) + \mathcal{O}\left(\frac{1}{N}\right).
\end{aligned}
$$

Lemma 2.8.14
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$
\begin{aligned}
\sup_{\mathfrak{e} \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))} \Bigg| & N m_{\mathfrak{e}} \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \\
& \times \left(\left(\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^n} \right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^n} \right) \right) \\
& + N m_{\mathfrak{e}} \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \\
& \times \left(\left(\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^n} \right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^n} \right) \right) \Bigg| \underset{N \to +\infty}{\longrightarrow} 0.
\end{aligned}
$$

Proof of Lemma 2.8.14. We follow the proof of Lemma 5.5.12 of [Glö13].

Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))$. For $x \geq \delta$, we define:

$$
f_1(x) = \frac{1}{x^n}, \quad f_2(x) = \frac{1}{(x)^{n,N}} \quad \text{and} \quad f_0(x) = f_2(x) - f_1(x).
$$

Then, we have:

$$
\begin{aligned}
&N m_{\mathfrak{e}} \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\left(\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n}} \right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^n} \right) \right) \\
&\qquad + N m_{\mathfrak{e}} \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\left(\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n}} \right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^n} \right) \right) \\
&= N m_{\mathfrak{e}} \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(f_0\left(m_{\mathfrak{e}} - \frac{1}{N}\right) - f_0(m_{\mathfrak{e}}) \right) \\
&\qquad + N m_{\mathfrak{e}} \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(f_0\left(m_{\mathfrak{e}} + \frac{1}{N}\right) - f_0(m_{\mathfrak{e}}) \right).
\end{aligned}
$$

Hence, a second order Taylor expansion of $f_0\left(m_{\mathfrak{e}} - \frac{1}{N}\right)$ and of $f_0\left(m_{\mathfrak{e}} + \frac{1}{N}\right)$ gives:

$$
\begin{aligned}
&N m_{\mathfrak{e}} \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\left(\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n}} \right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^n} \right) \right) \\
&\qquad + N m_{\mathfrak{e}} \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\left(\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n}} \right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^n} \right) \right) \\
&= N m_{\mathfrak{e}} \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(-\frac{1}{N} f_0'(m_{\mathfrak{e}}) + \frac{1}{2N^2} f_0''(m_{\mathfrak{e}}) + \mathcal{O}\left(\frac{1}{N^3}\right) \right) \\
&\qquad + N m_{\mathfrak{e}} \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}}) \right) \left(\frac{1}{N} f_0'(m_{\mathfrak{e}}) + \frac{1}{2N^2} f_0''(m_{\mathfrak{e}}) + \mathcal{O}\left(\frac{1}{N^3}\right) \right), \\
&= f_0'(m_{\mathfrak{e}}) m_{\mathfrak{e}} \left(N\beta(p_2^N - p_0^N)\psi(m_{\mathfrak{e}}) + \beta\psi'(m_{\mathfrak{e}}) \right) + f_0''(m_{\mathfrak{e}}) \frac{m_{\mathfrak{e}}}{2} \beta(p_0^N + p_2^N)\psi(m_{\mathfrak{e}}) + \mathcal{O}\left(\frac{1}{N}\right), \\
&= f_0'(m_{\mathfrak{e}}) m_{\mathfrak{e}} \left(2a\beta\psi(m_{\mathfrak{e}}) + \beta\psi'(m_{\mathfrak{e}}) \right) + f_0''(m_{\mathfrak{e}}) \frac{m_{\mathfrak{e}}}{2} \beta\psi(m_{\mathfrak{e}}) + \mathcal{O}\left(\frac{1}{N}\right).
\end{aligned} \tag{2.50}
$$

Moreover, expression (5.235) page 183 of [Glö13] ensures that:

$$
\sup_{\mathfrak{e} \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))} f_0''(m_{\mathfrak{e}}) \underset{N \to +\infty}{\longrightarrow} 0. \tag{2.51}
$$

And with expression (5.228) page 182 of [Glö13], we can show that, uniformly in $x \geq \delta$:

$$
f_2'(x) = -\frac{\sum_{i=0}^{n-1} (n-i) c_i \left(\frac{1}{N}\right)^i x^{n-i-1}}{\left(\sum_{i=0}^{n-1} c_i \left(\frac{1}{N}\right)^i x^{n-i}\right)^2} \underset{N \to +\infty}{\longrightarrow} -\frac{n c_0 x^{n-1}}{(c_0 x^n)^2} = -\frac{n}{c_0 x^{n+1}} = f_1'(x),
$$

since $c_0 = 1$ (see (5.128) page 172 of [Glö13]). So, this ensures:

$$
\sup_{\mathfrak{e} \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))} f_0'(m_{\mathfrak{e}}) \underset{N \to +\infty}{\longrightarrow} 0. \tag{2.52}
$$

Finally, expressions (2.50), (2.51) and (2.52) allow to conclude the proof. ✻

Lemma 2.8.15
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} \left(\mu, -\frac{1}{N}\delta_x\right) \right\rangle \right.$$
$$+ \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} \left(\mu, \frac{1}{N}\delta_x\right) \right\rangle$$
$$\left. - \left(\frac{n}{m_\mathfrak{e}^n}\left(2a\beta\psi(m_\mathfrak{e}) + \beta\psi'(m_\mathfrak{e})\right) - \frac{n^2}{m_\mathfrak{e}^{n+1}}\beta\psi(m_\mathfrak{e})\right) \left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle \right| \underset{N\longrightarrow+\infty}{\longrightarrow} 0.$$

Proof of Lemma 2.8.15. We follow the proof of Lemma 5.5.13 of [Glö13].

Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))$. We have:

$$\sum_{x\in E} \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} \pm \frac{1}{N}\right)^{n,N}} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} \left(\mu, \pm\frac{1}{N}\delta_x\right) \right\rangle = \frac{1}{\left(m_\mathfrak{e} \pm \frac{1}{N}\right)^{n,N}} \left\langle \omega\circ U^n_{(E,d)}, \pm\frac{1}{N} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} \sum_{x\in E} (\mu, \delta_x) \right\rangle,$$
$$= \frac{\pm 1}{\left(m_\mathfrak{e} \pm \frac{1}{N}\right)^{n,N}} \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{N} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} (\mu) \right\rangle.$$

Hence,

$$\left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} \left(\mu, -\frac{1}{N}\delta_x\right) \right\rangle$$
$$+ \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \sum_{x\in E} \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} \left(\mu, \frac{1}{N}\delta_x\right) \right\rangle$$
$$= \left[- \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} \right.$$
$$\left. + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right) \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} \right] \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{N} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} (\mu) \right\rangle.$$

Moreover, we have:

$$\left\langle \omega\circ U^n_{(E,d)}, \frac{1}{N} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} (\mu) \right\rangle = \left(\left\langle \omega\circ U^n_{(E,d)}, \frac{1}{N} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} (\mu) \right\rangle - n \left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle \right) + n \left\langle \omega\circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle,$$

and,

$$\frac{1}{\left(m_\mathfrak{e} \pm \frac{1}{N}\right)^{n,N}} = \left(\frac{1}{\left(m_\mathfrak{e} \pm \frac{1}{N}\right)^{n,N}} - \frac{1}{(m_\mathfrak{e})^{n,N}} \right) + \frac{1}{(m_\mathfrak{e})^{n,N}}.$$

So, using the same decomposition as in the proof of Lemma 2.8.11, we have:

$$\frac{1}{\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n,N}} = \left(\frac{1}{\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n}} - \frac{1}{m_{\mathfrak{e}}^{n}}\right) + \left(\left(\frac{1}{\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n}}\right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^{n}}\right)\right) + \frac{1}{(m_{\mathfrak{e}})^{n,N}}.$$

Lemma 2.8.16 (which corresponds to Lemma 5.5.14 of [Glö13]) ensures the convergence of the term

$$\left[-\left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right)\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n,N}} + \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right)\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n,N}}\right]\left(\left\langle \omega \circ U^n_{(E,d)}, \frac{1}{N}\sum_{l_1=1}^{n}\bigotimes_{(l_1)}^{n,N}(\mu)\right\rangle - n\left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle\right)$$

to 0.

Lemma 2.8.17 ensures the convergence of the term

$$\left[-\left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right)\left(\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n}} - \frac{1}{m_{\mathfrak{e}}^{n}}\right) + \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right)\left(\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n}} - \frac{1}{m_{\mathfrak{e}}^{n}}\right)\right] n\left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle$$

to $-\frac{n^2}{m_{\mathfrak{e}}^{n+1}}\beta\psi(m_{\mathfrak{e}})\left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle$.

Lemma 2.8.18 ensures the convergence of the term

$$\left[-\left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right) \times \left(\left(\frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n}}\right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^{n}}\right)\right) + \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right) \times \left(\left(\frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n}}\right) - \left(\frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^{n}}\right)\right)\right] n\left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle$$

to 0.

Finally, Lemma 2.8.19 ensures the convergence of the term

$$\left[-\left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right)\frac{1}{(m_{\mathfrak{e}})^{n,N}} + \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right)\frac{1}{(m_{\mathfrak{e}})^{n,N}}\right] n\left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle$$

to $\frac{n}{m_{\mathfrak{e}}^n}\left(2a\beta\psi(m_{\mathfrak{e}}) + \beta\psi'(m_{\mathfrak{e}})\right)\left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n}\right\rangle$.

Lemma 2.8.16 (see [Glö13])
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\sup_{\mathfrak{e} \in \mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| \left\langle \omega \circ U^n_{(E,d)}, \frac{1}{N} \sum_{l_1=1}^{n} \bigotimes_{(l_1)}^{n,N} (\mu) \right\rangle - n \left\langle \omega \circ U^n_{(E,d)}, \mu^{\otimes n} \right\rangle \right| \underset{N \to +\infty}{\longrightarrow} 0.$$

Lemma 2.8.17
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\sup_{\mathfrak{e} \in \mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| - \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \left(\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^n} - \frac{1}{m_\mathfrak{e}^n} \right) + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \left(\frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^n} - \frac{1}{m_\mathfrak{e}^n} \right) + \frac{n}{m_\mathfrak{e}^{n+1}} \beta \psi(m_\mathfrak{e}) \right| \underset{N \to +\infty}{\longrightarrow} 0.$$

Proof of Lemma 2.8.17. We follow the proof of Lemma 5.5.15 of [Glö13].

Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))$. As in the proof of Lemma 2.8.13, a first order Taylor expansion of $\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^n}$ and of $\frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^n}$ makes it possible to conclude. We have:

$$\begin{aligned}
&\left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \left(\frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^n} - \frac{1}{m_\mathfrak{e}^n} \right) - \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \left(\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^n} - \frac{1}{m_\mathfrak{e}^n} \right) \\
&= \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \left(-\frac{n}{N m_\mathfrak{e}^{n+1}} + \mathcal{O}\left(\frac{1}{N^2}\right) \right) \\
&\qquad - \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \left(\frac{n}{N m_\mathfrak{e}^{n+1}} + \mathcal{O}\left(\frac{1}{N^2}\right) \right) \\
&= -\frac{n}{m_\mathfrak{e}^{n+1}} \beta (p_2^N + p_0^N) \psi(m_\mathfrak{e}) + \mathcal{O}\left(\frac{1}{N}\right) \\
&= -\frac{n}{m_\mathfrak{e}^{n+1}} \beta \psi(m_\mathfrak{e}) + \mathcal{O}\left(\frac{1}{N}\right)
\end{aligned}$$

Lemma 2.8.18
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\begin{aligned}
\sup_{\mathfrak{e} \in \mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \Bigg| &- \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \\
&\times \left(\left(\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^n} \right) - \left(\frac{1}{(m_\mathfrak{e})^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \right) \right) \\
&\left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \\
&\times \left(\left(\frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^n} \right) - \left(\frac{1}{(m_\mathfrak{e})^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \right) \right) \Bigg| \underset{N \to +\infty}{\longrightarrow} 0.
\end{aligned}$$

Proof of Lemma 2.8.18. We follow the proof of Lemma 5.5.16 of [Glö13].

Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))$. The same function f_0 is used as in the proof of Lemma 2.8.14.

We have:

$$\begin{aligned}
&\left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(\left(\frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n}}\right) - \left(\frac{1}{(m_\mathfrak{e})^{n,N}} - \frac{1}{m_\mathfrak{e}^n}\right)\right)\\
&\qquad - \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(\left(\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n}}\right) - \left(\frac{1}{(m_\mathfrak{e})^{n,N}} - \frac{1}{m_\mathfrak{e}^n}\right)\right)\\
&= \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(f_0\left(m_\mathfrak{e} + \frac{1}{N}\right) - f_0(m_\mathfrak{e})\right)\\
&\qquad - \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(f_0\left(m_\mathfrak{e} - \frac{1}{N}\right) - f_0(m_\mathfrak{e})\right).
\end{aligned}$$

Hence, a first order Taylor expansion of $f_0\left(m_\mathfrak{e} - \frac{1}{N}\right)$ and of $f_0\left(m_\mathfrak{e} + \frac{1}{N}\right)$ gives:

$$\begin{aligned}
&\left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(\left(\frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n}}\right) - \left(\frac{1}{(m_\mathfrak{e})^{n,N}} - \frac{1}{m_\mathfrak{e}^n}\right)\right)\\
&\qquad - \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(\left(\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n}}\right) - \left(\frac{1}{(m_\mathfrak{e})^{n,N}} - \frac{1}{m_\mathfrak{e}^n}\right)\right)\\
&= \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(\frac{1}{N}f_0'(m_\mathfrak{e}) + \mathcal{O}\left(\frac{1}{N^2}\right)\right)\\
&\qquad - \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\left(-\frac{1}{N}f_0'(m_\mathfrak{e}) + \mathcal{O}\left(\frac{1}{N^2}\right)\right),\\
&= f_0'(m_\mathfrak{e})\beta(p_0^N + p_2^N)\psi(m_\mathfrak{e}) + \mathcal{O}\left(\frac{1}{N}\right),\\
&= f_0'(m_\mathfrak{e})\beta\psi(m_\mathfrak{e}) + \mathcal{O}\left(\frac{1}{N}\right).
\end{aligned}$$

Finally, expression (2.52) allows to conclude. ❋

Lemma 2.8.19
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\begin{aligned}
\sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))}\Bigg| &- \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{(m_\mathfrak{e})^{n,N}} + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{(m_\mathfrak{e})^{n,N}}\\
&- \left(2a\beta\psi(m_\mathfrak{e}) + \beta\psi'(m_\mathfrak{e})\right)\frac{1}{m_\mathfrak{e}^n}\Bigg| \underset{N\longrightarrow+\infty}{\longrightarrow} 0.
\end{aligned}$$

Proof of Lemma 2.8.19. Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))$.

We have:

$$
\begin{aligned}
-\left(N\beta p_0^N\psi(m_\epsilon)-\frac{\beta}{2}\psi'(m_\epsilon)\right)\frac{1}{(m_\epsilon)^{n,N}}+\left(N\beta p_2^N\psi(m_\epsilon)+\frac{\beta}{2}\psi'(m_\epsilon)\right)\frac{1}{(m_\epsilon)^{n,N}}& \\
=\frac{1}{(m_\epsilon)^{n,N}}\left(N\beta\left(p_2^N-p_0^N\right)\psi(m_\epsilon)+\beta\psi'(m_\epsilon)\right),& \\
=\frac{1}{(m_\epsilon)^{n,N}}\left(2a\beta\psi(m_\epsilon)+\beta\psi'(m_\epsilon)\right).&
\end{aligned}
$$

So, to prove this lemma, it suffices to prove:

$$
\sup_{\epsilon\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))}\left|\frac{1}{(m_\epsilon)^{n,N}}-\frac{1}{m_\epsilon^n}\right|\underset{N\longrightarrow+\infty}{\longrightarrow}0.
$$

In order to do this, we first prove that the function:

$$
f_{0,n}(x)=f_{2,n}(x)-f_{1,n}(x)=\frac{1}{(x)^{n,N}}-\frac{1}{x^n} \tag{2.53}
$$

is decreasing for each $n\in\mathbb{N}$, $n\geq 2$ and for all $x\in[\delta,+\infty[\cap supp(\psi)$ when $N\in\mathbb{N}$ is large enough to ensure that $\delta-\frac{n}{N}>0$.

We can note that for each $n\in\mathbb{N}$, $n\geq 2$ and for all $x\in[\delta,+\infty[\cap supp(\psi)$,

$$
f_{0,n}(x)>0.
$$

- For $n=2$,

$$
f_{0,2}(x)=f_{2,2}(x)-f_{1,2}(x)=\frac{1}{(x)^{2,N}}-\frac{1}{x^2}=\frac{1}{x\left(x-\frac{1}{N}\right)}-\frac{1}{x^2}>0.
$$

Hence, we have:

$$
\begin{aligned}
f'_{0,2}(x)&=-\frac{1}{x^2\left(x-\frac{1}{N}\right)}-\frac{1}{x\left(x-\frac{1}{N}\right)^2}+\frac{2}{x^3},\\
&=-\left(\frac{1}{x}+\frac{1}{\left(x-\frac{1}{N}\right)}\right)f_{2,2}(x)+\frac{2}{x}f_{1,2}(x),\\
&=\left(\frac{2}{x}-\frac{1}{x}-\frac{1}{\left(x-\frac{1}{N}\right)}\right)f_{2,2}(x)+\frac{2}{x}f_{1,2}(x)-\frac{2}{x}f_{2,2}(x),\\
&=\left(\frac{1}{x}-\frac{1}{\left(x-\frac{1}{N}\right)}\right)f_{2,2}(x)-\frac{2}{x}f_{0,2}(x)<0.
\end{aligned}
$$

So, the function $f_{0,2}$ is decreasing.

- We assume that:

$$
f'_{2,n-1}(x)=-\left(\frac{1}{x}+\frac{1}{\left(x-\frac{1}{N}\right)}+\cdots+\frac{1}{\left(x-\frac{n-2}{N}\right)}\right)f_{2,n-1}(x).
$$

We can note that:

$$
f_{0,n}(x)=\frac{1}{\left(x-\frac{n-1}{N}\right)}f_{2,n-1}(x)-\frac{1}{x^n}.
$$

Hence, we have:

$$\begin{aligned}
f'_{0,n}(x) &= -\frac{1}{\left(x-\frac{n-1}{N}\right)^2} f_{2,n-1}(x) + \frac{1}{\left(x-\frac{n-1}{N}\right)} f'_{2,n-1}(x) + \frac{n}{x^{n+1}} \\
&= -\frac{1}{\left(x-\frac{n-1}{N}\right)} f_{2,n}(x) - \left(\frac{1}{x} + \frac{1}{\left(x-\frac{1}{N}\right)} + \cdots + \frac{1}{\left(x-\frac{n-2}{N}\right)}\right) \frac{1}{\left(x-\frac{n-1}{N}\right)} f_{2,n-1}(x) + \frac{n}{x^{n+1}} \\
&= -\left(\frac{1}{x} + \frac{1}{\left(x-\frac{1}{N}\right)} + \cdots + \frac{1}{\left(x-\frac{n-2}{N}\right)} + \frac{1}{\left(x-\frac{n-1}{N}\right)}\right) f_{2,n}(x) + \frac{n}{x} f_{1,n}(x) \\
&= \left(\frac{n}{x} - \frac{1}{x} - \frac{1}{\left(x-\frac{1}{N}\right)} - \cdots - \frac{1}{\left(x-\frac{n-1}{N}\right)}\right) f_{2,n}(x) - \frac{n}{x} f_{0,n}(x) \\
&= \left(\left(\frac{1}{x} - \frac{1}{\left(x-\frac{1}{N}\right)}\right) + \cdots + \left(\frac{1}{x} - \frac{1}{\left(x-\frac{n-1}{N}\right)}\right)\right) f_{2,n}(x) - \frac{n}{x} f_{0,n}(x) < 0.
\end{aligned}$$

By induction, the function $f_{0,n}$ is decreasing for each $n \in \mathbb{N}$, $n \geq 2$ and for all $x \in [\delta, +\infty[\cap supp(\psi)$ when $N \in \mathbb{N}$ is large enough. In particular, as $m_{\mathfrak{e}} \geq \delta$, we have:

$$\lim_{N\to+\infty} \sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| \frac{1}{(m_{\mathfrak{e}})^{n,N}} - \frac{1}{m_{\mathfrak{e}}^n} \right| \leq \lim_{N\to+\infty} \left| \frac{1}{(\delta)^{n,N}} - \frac{1}{\delta^n} \right| = 0.$$

Lemma 2.8.20
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\begin{aligned}
\sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \Bigg| & \left(N\beta p_0^N \psi(m_{\mathfrak{e}}) - \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right) \sum_{x\in E} \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_{\mathfrak{e}} - \frac{1}{N}\right)^{n,N}} \sum_{1\leq l_1<l_2\leq n} \bigotimes_{(l_1,l_2)}^{n,N} \left(\mu, -\frac{1}{N}\delta_x\right) \right\rangle \\
& + \left(N\beta p_2^N \psi(m_{\mathfrak{e}}) + \frac{\beta}{2}\psi'(m_{\mathfrak{e}})\right) \sum_{x\in E} \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n,N}} \sum_{1\leq l_1<l_2\leq n} \bigotimes_{(l_1,l_2)}^{n,N} \left(\mu, \frac{1}{N}\delta_x\right) \right\rangle \\
& - \frac{1}{m_{\mathfrak{e}}^{n+1}} \beta\psi(m_{\mathfrak{e}}) \sum_{1\leq l_1<l_2\leq n} \left\langle \omega\circ r_{l_1,l_2}\circ U^n_{(E,d)}, \mu^{\otimes n} \right\rangle \Bigg| \underset{N\to+\infty}{\longrightarrow} 0.
\end{aligned}$$

where $r_{i,j}$ is defined in (2.45).

Proof of Lemma 2.8.20. We follow the proof of Lemma 5.5.17 of [Glö13].

Assume $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}^N([\delta, +\infty[\cap supp(\psi))$. We have:

$$\begin{aligned}
\sum_{x\in E} & \left\langle \omega\circ U^n_{(E,d)}, \frac{1}{\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n,N}} \sum_{1\leq l_1<l_2\leq n} \bigotimes_{(l_1,l_2)}^{n,N} \left(\mu, \pm\frac{1}{N}\delta_x\right) \right\rangle \\
&= \frac{1}{\left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n,N}} \left\langle \omega\circ U^n_{(E,d)}, \left(\pm\frac{1}{N}\right)^2 \sum_{1\leq l_1<l_2\leq n} \sum_{x\in E} \bigotimes_{(l_1,l_2)}^{n,N} (\mu, \delta_x) \right\rangle, \\
&= \frac{1}{N m_{\mathfrak{e}} \left(m_{\mathfrak{e}} \pm \frac{1}{N}\right)^{n,N}} \left\langle \omega\circ U^n_{(E,d)}, \sum_{1\leq l_1<l_2\leq n} \frac{m_{\mathfrak{e}}}{N} \sum_{x\in E} \bigotimes_{(l_1,l_2)}^{n,N} (\mu, \delta_x) \right\rangle.
\end{aligned}$$

Hence,

$$\begin{aligned}
&\left(N\beta p_0^N\psi(m_\mathfrak{e})-\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\sum_{x\in E}\left\langle \omega\circ U^n_{(E,d)},\frac{1}{\left(m_\mathfrak{e}-\frac{1}{N}\right)^{n,N}}\sum_{1\le l_1<l_2\le n}\bigotimes_{(l_1,l_2)}^{n,N}\left(\mu,-\frac{1}{N}\delta_x\right)\right\rangle\\
&\qquad+\left(N\beta p_2^N\psi(m_\mathfrak{e})+\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\sum_{x\in E}\left\langle \omega\circ U^n_{(E,d)},\frac{1}{\left(m_\mathfrak{e}+\frac{1}{N}\right)^{n,N}}\sum_{1\le l_1<l_2\le n}\bigotimes_{(l_1,l_2)}^{n,N}\left(\mu,\frac{1}{N}\delta_x\right)\right\rangle\\
&=\left[\left(N\beta p_0^N\psi(m_\mathfrak{e})-\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}\left(m_\mathfrak{e}-\frac{1}{N}\right)^{n,N}}\right.\\
&\qquad\left.+\left(N\beta p_2^N\psi(m_\mathfrak{e})+\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}\left(m_\mathfrak{e}+\frac{1}{N}\right)^{n,N}}\right]\left\langle \omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n,N}(\mu,\delta_x)\right\rangle.
\end{aligned}$$

Moreover, we have:

$$\begin{aligned}
&\left\langle \omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n,N}(\mu,\delta_x)\right\rangle\\
&\qquad=\left(\left\langle \omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n,N}(\mu,\delta_x)\right\rangle-\left\langle \omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n}(\mu,\delta_x)\right\rangle\right)\\
&\qquad\qquad+\left\langle \omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n}(\mu,\delta_x)\right\rangle.
\end{aligned}$$

And thanks to Lemma 2.8.21 (which corresponds to Lemma 5.5.18 of [Glö13]), we have:

$$\begin{aligned}
\left\langle \omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n}(\mu,\delta_x)\right\rangle&=\left\langle \omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\mu^{\otimes n}\circ\left(\overline{r}_{l_1,l_2}\right)^{-1}\right\rangle,\\
&=\sum_{1\le l_1<l_2\le n}\left\langle \omega\circ U^n_{(E,d)}\circ\overline{r}_{l_1,l_2},\mu^{\otimes n}\right\rangle,\\
&=\sum_{1\le l_1<l_2\le n}\left\langle \omega\circ r_{l_1,l_2}\circ U^n_{(E,d)},\mu^{\otimes n}\right\rangle.
\end{aligned}$$

The last equality being obtained thanks to expression (2.47). We also have:

$$\frac{1}{\left(m_\mathfrak{e}\pm\frac{1}{N}\right)^{n,N}}=\left(\frac{1}{\left(m_\mathfrak{e}\pm\frac{1}{N}\right)^{n,N}}-\frac{1}{m_\mathfrak{e}^n}\right)+\frac{1}{m_\mathfrak{e}^n}.$$

Hence,

$$\begin{aligned}
&\left(N\beta p_0^N\psi(m_\mathfrak{e})-\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}\left(m_\mathfrak{e}-\frac{1}{N}\right)^{n,N}}+\left(N\beta p_2^N\psi(m_\mathfrak{e})+\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}\left(m_\mathfrak{e}+\frac{1}{N}\right)^{n,N}}\\
&\quad=\left(N\beta p_0^N\psi(m_\mathfrak{e})-\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}}\left(\frac{1}{\left(m_\mathfrak{e}-\frac{1}{N}\right)^{n,N}}-\frac{1}{m_\mathfrak{e}^n}\right)\\
&\qquad\qquad+\left(N\beta p_2^N\psi(m_\mathfrak{e})+\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}}\left(\frac{1}{\left(m_\mathfrak{e}+\frac{1}{N}\right)^{n,N}}-\frac{1}{m_\mathfrak{e}^n}\right)\\
&\qquad\qquad+\left(N\beta p_0^N\psi(m_\mathfrak{e})-\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}^{n+1}}+\left(N\beta p_2^N\psi(m_\mathfrak{e})+\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}^{n+1}}.
\end{aligned}$$

And,

$$\left(N\beta p_0^N\psi(m_\mathfrak{e})-\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}^{n+1}}+\left(N\beta p_2^N\psi(m_\mathfrak{e})+\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}^{n+1}}=\frac{1}{m_\mathfrak{e}^{n+1}}\beta\psi(m_\mathfrak{e}).$$

Lemma 2.8.22 (which corresponds to Lemma 5.5.19 of [Glö13]) ensures convergence of term

$$\left[\left(N\beta p_0^N\psi(m_\mathfrak{e})-\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}\left(m_\mathfrak{e}-\frac{1}{N}\right)^{n,N}}+\left(N\beta p_2^N\psi(m_\mathfrak{e})+\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}\left(m_\mathfrak{e}+\frac{1}{N}\right)^{n,N}}\right]$$
$$\times\left(\left\langle\omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n,N}(\mu,\delta_x)\right\rangle-\left\langle\omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n}(\mu,\delta_x)\right\rangle\right)$$

to 0.

Lemma 2.8.23 ensures convergence of term

$$\left[\left(N\beta p_0^N\psi(m_\mathfrak{e})-\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}}\left(\frac{1}{\left(m_\mathfrak{e}-\frac{1}{N}\right)^{n,N}}-\frac{1}{m_\mathfrak{e}^n}\right)\right.$$
$$\left.+\left(N\beta p_2^N\psi(m_\mathfrak{e})+\frac{\beta}{2}\psi'(m_\mathfrak{e})\right)\frac{1}{Nm_\mathfrak{e}}\left(\frac{1}{\left(m_\mathfrak{e}+\frac{1}{N}\right)^{n,N}}-\frac{1}{m_\mathfrak{e}^n}\right)\right]\sum_{1\le l_1<l_2\le n}\left\langle\omega\circ r_{l_1,l_2}\circ U^n_{(E,d)},\mu^{\otimes n}\right\rangle$$

to 0.

Therefore, all that remains is the term

$$\frac{1}{m_\mathfrak{e}^{n+1}}\beta\psi(m_\mathfrak{e})\sum_{1\le l_1<l_2\le n}\left\langle\omega\circ r_{l_1,l_2}\circ U^n_{(E,d)},\mu^{\otimes n}\right\rangle.$$

✻

Lemma 2.8.21 (see [Glö13])
Let $\mathfrak{e}=[E,d,\mu]\in\mathbb{M}^N_{>0}$, $n\in\mathbb{N}$, $n\ge 2$ and $l_1,l_2\in\mathbb{N}$ such that $1\le l_1<l_2\le n$. Then,

$$\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n}(\mu,\delta_x)=\mu^{\otimes n}\circ\left(\overline{r}_{l_1,l_2}\right)^{-1},$$

where $\overline{r}_{i,j}$ is defined in (2.46).

Lemma 2.8.22 (see [Glö13])
Let $n\in\mathbb{N}$, $n\ge 2$, $\omega\in C_b(\mathbb{U}^n)$ and $\psi\in C^\infty_K(\mathbb{R}_+)$. Then, for each $\delta>0$,

$$\sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))}\left|\left\langle\omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n,N}(\mu,\delta_x)\right\rangle\right.$$
$$\left.-\left\langle\omega\circ U^n_{(E,d)},\sum_{1\le l_1<l_2\le n}\frac{m_\mathfrak{e}}{N}\sum_{x\in E}\bigotimes_{(l_1,l_2)}^{n}(\mu,\delta_x)\right\rangle\right|\underset{N\to+\infty}{\longrightarrow}0.$$

Lemma 2.8.23
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in \mathcal{C}_b(\mathbb{U}^n)$ and $\psi \in \mathcal{C}_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \frac{1}{Nm_\mathfrak{e}} \left(\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \right) \right.$$
$$\left. + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \frac{1}{Nm_\mathfrak{e}} \left(\frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \right) \right| \underset{N\longrightarrow+\infty}{\longrightarrow} 0.$$

Proof of Lemma 2.8.23. We have:

$$\lim_{N\longrightarrow+\infty} \sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| \left(N\beta p_0^N \psi(m_\mathfrak{e}) - \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \frac{1}{Nm_\mathfrak{e}} \left(\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \right) \right.$$
$$\left. + \left(N\beta p_2^N \psi(m_\mathfrak{e}) + \frac{\beta}{2}\psi'(m_\mathfrak{e}) \right) \frac{1}{Nm_\mathfrak{e}} \left(\frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \right) \right|$$
$$\leq \beta\|\psi\|_\infty \frac{1}{\delta} \lim_{N\longrightarrow+\infty} \sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \right| + \frac{\beta}{2}\|\psi'\|_\infty \frac{1}{\delta} \lim_{N\longrightarrow+\infty} \frac{1}{N} \left(\frac{1}{\left(\delta - \frac{1}{N}\right)^{n,N}} + \frac{1}{\delta^n} \right)$$
$$+ \beta\|\psi\|_\infty \frac{1}{\delta} \lim_{N\longrightarrow+\infty} \sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \right| + \frac{\beta}{2}\|\psi'\|_\infty \frac{1}{\delta} \lim_{N\longrightarrow+\infty} \frac{1}{N} \left(\frac{1}{\left(\delta + \frac{1}{N}\right)^{n,N}} + \frac{1}{\delta^n} \right)$$

Clearly,

$$\lim_{N\longrightarrow+\infty} \frac{1}{N} \left(\frac{1}{\left(\delta - \frac{1}{N}\right)^{n,N}} + \frac{1}{\delta^n} \right) = 0 \text{ and } \lim_{N\longrightarrow+\infty} \frac{1}{N} \left(\frac{1}{\left(\delta + \frac{1}{N}\right)^{n,N}} + \frac{1}{\delta^n} \right) = 0.$$

Moreover, we have:

$$\frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \geq \frac{1}{(m_\mathfrak{e})^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \geq 0,$$

and we can show that the function $x \mapsto \left(\frac{1}{\left(x - \frac{1}{N}\right)^{n,N}} - \frac{1}{x^n} \right)$ is decreasing as we have done for the function $x \mapsto \left(\frac{1}{(x)^{n,N}} - \frac{1}{x^n} \right)$ in the proof of Lemma 2.8.19. So, since $m_\mathfrak{e} \geq \delta$, we have:

$$\beta\|\psi\|_\infty \frac{1}{\delta} \lim_{N\longrightarrow+\infty} \sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| \frac{1}{\left(m_\mathfrak{e} - \frac{1}{N}\right)^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \right| \leq \beta\|\psi\|_\infty \frac{1}{\delta} \lim_{N\longrightarrow+\infty} \left(\frac{1}{\left(\delta - \frac{1}{N}\right)^{n,N}} - \frac{1}{\delta^n} \right) = 0.$$

Finally, we have:

$$\left| \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} - \frac{1}{m_\mathfrak{e}^n} \right| \leq \left| \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n}} \right| + \left| \frac{1}{\left(m_\mathfrak{e} + \frac{1}{N}\right)^{n}} - \frac{1}{m_\mathfrak{e}^n} \right|$$

As in the proof of Lemma 2.8.19, we have:

$$\beta\|\psi\|_\infty \frac{1}{\delta} \lim_{N\to+\infty} \sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| \frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n}} \right|$$
$$\leq \beta\|\psi\|_\infty \frac{1}{\delta} \lim_{N\to+\infty} \left(\frac{1}{\left(\delta + \frac{1}{N}\right)^{n,N}} - \frac{1}{\left(\delta + \frac{1}{N}\right)^{n}} \right) = 0.$$

And for every $x \in [\delta, +\infty[\cap supp(\psi)$, we have:

$$f_{3,n}(x) = \frac{1}{x^n} - \frac{1}{\left(x + \frac{1}{N}\right)^n} > 0,$$

and

$$f'_{3,n}(x) = -n\left(\frac{1}{x^{n+1}} - \frac{1}{\left(x + \frac{1}{N}\right)^{n+1}} \right) < 0.$$

Therefore, the function $x \mapsto \frac{1}{x^n} - \frac{1}{\left(x+\frac{1}{N}\right)^n}$ is decreasing and we have:

$$\beta\|\psi\|_\infty \frac{1}{\delta} \lim_{N\to+\infty} \sup_{\mathfrak{e}\in\mathbb{T}^N([\delta,+\infty[\cap supp(\psi))} \left| \frac{1}{\left(m_{\mathfrak{e}} + \frac{1}{N}\right)^{n}} - \frac{1}{m_{\mathfrak{e}}^n} \right| \leq \beta\|\psi\|_\infty \frac{1}{\delta} \lim_{N\to+\infty} \left(\frac{1}{\delta^n} - \frac{1}{\left(\delta + \frac{1}{N}\right)^{n}} \right) = 0.$$

8.4 - Convergence of the Tree-Valued Process

Now that we have found the limits for the growth generator and for the binding generator, we can define the limit of the infinitesimal generator of the process $\mathfrak{T}^N$.

Definition 2.8.24 (Limit of Generator of $\mathfrak{T}^N$)
Let $\mathfrak{e} = [E, d, \mu] \in \mathbb{T}$, $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b^{1bc}(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. The *limit of the generator of the tree-valued process* applied to the Ψ-polynomial $\Psi^n_{\omega,\psi}$ is given by:

$$\mathcal{G}^\infty \Psi^n_{\omega,\psi}(\mathfrak{e}) = \mathcal{G}^\infty_{grow} \Psi^n_{\omega,\psi}(\mathfrak{e}) + \mathcal{G}^\infty_{bind} \Psi^n_{\omega,\psi}(\mathfrak{e}),$$

where $\mathcal{G}^\infty_{grow}$ and $\mathcal{G}^\infty_{bind}$ are defined respectively in (2.48) and (2.49).

Theorem 2.8.25 (Convergence of Generator of $\mathfrak{T}^N$)
Let $n \in \mathbb{N}$, $n \geq 2$, $\omega \in C_b^{1bc}(\mathbb{U}^n)$ and $\psi \in C_K^\infty(\mathbb{R}_+)$. Then, for each $\delta > 0$,

$$\left\| \mathcal{G}^N \Psi^{n,N}_{\omega,\psi} - \mathcal{G}^\infty \Psi^n_{\omega,\psi} \right\|_{N,\delta} \underset{N\to+\infty}{\longrightarrow} 0.$$

Proof of Theorem 2.8.25. This proof is direct by using Proposition 2.8.5 and Proposition 2.8.8.

In Section 7, we have proved that the family of stopped process $\left(\mathfrak{T}^N_{\cdot,R}\right)_{N\in\mathbb{N}}$ is tight. Hence, by Prokhorov's theorem, every sequence contains a convergent subsequence. The uniform convergence stated in Theorem 2.8.25 proves that every such convergent subsequence solves a certain martingale problem. To summarize what we have obtained in this chapter, we have the following result.

Theorem 2.8.26
Let $R \in \mathbb{N}^*$. Assume $\limsup_{N\to+\infty} \mathbb{E}\left[\mathfrak{m}_0^N\right] < \infty$ and $\mathfrak{T}_{0,R}^N \xrightarrow[N\to+\infty]{\mathcal{L}} \nu \in \mathcal{M}_1(\mathbb{T})$ where $\nu(\mathbb{M}_{>0}) = 1$. Then, $\left(\mathfrak{T}_{\cdot,R}^N\right)_{N\in\mathbb{N}}$ is relatively compact on $D_{\mathbb{T}}[0,+\infty[$ and the limit of any convergent subsequence solves the martingale problem for:

$$\left(\mathcal{G}^\infty,\ {}^S H_R,\ \nu\right),$$

where ${}^S H_R$ is the linear span of:

$$H_R = \left\{\Psi \in \Pi_\Psi\left(C_b^{1bc}, C_K^\infty\right) \mid \mathcal{G}^\infty\Psi \equiv 0 \text{ on } \mathbb{T}\left([0,1/R]\right)\right\}.$$

Moreover, Theorem 1 of [DG19] (and remarks 3.7, 5.8 and 5.13 suggesting that this theorem and its proof can be adapted to the non-critical case) ensures that the above martingale problem is well-posed (i.e. the solution of the martingale problem is unique). Therefore, we can prove that the accumulation point of $\left(\mathfrak{T}_{\cdot,R}^N\right)_{N\in\mathbb{N}}$ is unique and consequently prove that our stopped tree-valued branching process converges in distribution. This theorem also provides properties of the limit, detailed in the following theorem.

Theorem 2.8.27
Let $R \in \mathbb{N}^*$. Assume $\limsup_{N\to+\infty} \mathbb{E}\left[\mathfrak{m}_0^N\right] < \infty$ and $\mathfrak{T}_{0,R}^N \xrightarrow[N\to+\infty]{\mathcal{L}} \nu \in \mathcal{M}_1(\mathbb{T})$ where $\nu(\mathbb{M}_{>0}) = 1$.

Then the stopped process $\mathfrak{T}_R^N$ converges in distribution in $D_{\mathbb{T}}[0,+\infty[$ as $N \to +\infty$ to $\mathfrak{T}^\infty = (\mathfrak{T}_t^\infty)_{t\in\mathbb{R}_+}$. Moreover, the limit has the Feller property (which means that for every continuous and bounded function Φ on $\mathbb{T}$, the function $\nu \mapsto \mathbb{E}_\nu\left[\Phi\left(\mathfrak{T}_t^\infty\right)\right]$ is continuous in the weak topology on $\mathcal{M}_1(\mathbb{T})$), the strong Markov property and almost surely there exists a version with continuous paths.

We can also note that Definition 7.3.1 (and the following comment at the top of page 219) and Proposition 7.4.1 of [Glö13] ensures the convergence in law of the scaled number of extremities $(\tilde{X}_t^N)_{t\in\mathbb{R}_+}$ to a Feller branching diffusion. That is,

Proposition 2.8.28 (see [Glö13])
Assume that $\limsup_{N\to+\infty} \mathbb{E}\left[\tilde{X}_0^N\right] < M < +\infty$. Assume moreover that $\nu^N = \mathcal{L}\text{aw}\left[\tilde{X}_0^N\right] \xrightarrow[N\to+\infty]{\mathcal{L}} \nu \in \mathcal{M}_1(\mathbb{R}_+)$. Then,

$$\tilde{X}^N \xrightarrow[N\to+\infty]{\mathcal{L}} \tilde{X}^\infty$$

on $D_{\mathbb{R}_+}[0,+\infty[$, where $\tilde{X}^\infty$ is a Feller branching diffusion solution to:

$$dZ_t = 2a\beta Z_t dt + \sqrt{\beta Z_t} dB_t,$$

and such that, $\mathcal{L}\text{aw}\left[\tilde{X}_0^\infty\right] = \nu$.

9 - Perspectives

This model may seem naive in view of the complexity of actin dynamics, but it is in fact relevant at least in particular situations. The first thing we can notice is that in this model, the number of available monomers is not assumed to be bounded (unlike previous models). Therefore, the elongation rate does not depend on the concentration of free monomers and is assumed to be constant. We made this choice

because we further assume that our polymer has a large number of extremities (of the order of N) and short branches (of the order of 1). This type of actin polymers is mainly found in protrusions, which allow cells to move, and in which the concentration of free monomers is very high.

We can also note that the choice of the offspring distribution was motivated by mathematical considerations. Indeed, by choosing $p_N = \frac{1}{2} \pm \frac{a}{N}$, we chose a slightly sub-/super-critical case which is quite simple to study but had not yet been considered. Moreover, this assumption is consistent with biological considerations since it allows to give an advantage to one of the accessory proteins over the other.

This model takes into account the following phenomena:

- spontaneous elongation of the extremities,
- branching due to the Arp2/3 complexes,
- fragmentation by cofilins.

To this we could add:

- depolymerisation of the main branch, which then only takes place at the root of the tree since the pointed ends of the secondary branches are attached to an Arp2/3 complex,
- detachment of an Arp2/3 complex, resulting in the loss of a branch,
- effect of capping proteins, which inhibits the elongation of the capped ends.

We have chosen not to take depolymerisation into account because its effect seems negligible compared to fragmentation. Indeed, depolymerisation releases monomers one at a time from a single extremity whereas fragmentation can take place on all extremities and can result in the release of several monomers at once. In addition, we are considering here a polymer with a number of extremities of the order of N, therefore the phenomenon described above is exacerbated.

Detachment of an Arp2/3 complex and fragmentation having the same result — the loss of one extremity and its branch — we can consider that our current rate of fragmentation is in fact an average rate of "loss of a branch" taking into account the effect of cofilin as well as the detachment of Arp2/3.

The effect of capping proteins has not been taken into account in this model in order to keep the dynamics relatively simple. This allowed us to focused on the introduction of a framework, already known in population genetics but not in the field of molecular biology (to our knowledge) that will later allow us to model a wide range of interactions between branched polymers and other proteins present in the cellular cortex. As the effect of capping proteins seems to be very important in the dynamics of branched actin polymers, this is the first improvement to be made to this model. To take capping proteins into account, we would have to define a new process $(\tilde{Y}_t^N)_{t\in\mathbb{R}_+}$, defined in the same way as process $(\tilde{X}_t^N)_{t\in\mathbb{R}_+}$, but which would represent the evolution of the (scaled) number of free extremities (i.e. uncapped) instead of the total (scaled) number of extremities. As capping proteins inhibit elongation but not the binding of an Arp2/3 complex or a cofilin, the operators defined for the binding event will not change. The operator encoding the elongation will also remain the same, however it will no longer apply to the $N\tilde{X}_t^N$ extremities, but only to the $N\tilde{Y}_t^N$ free extremities.

A second relatively simple improvement would be not to choose the branch on which a binding will take place according to a uniform law. Indeed, Arp2/3 complexes or cofilin can bind to any monomer in a branch. Thus, the longer a branch is, the greater the probability that a binding event will take place

on that branch. For example, we could choose the following probability:

$$\mathbb{P}\,(\text{binding on the } i\text{-th branch}) = \frac{\ell_i}{\sum_{j=1}^{N\tilde{X}_t^N} \ell_j},$$

where ℓ_i is the length in number of monomer of the i-th branch. The difficulty would then be to determine the length of each branch from the distance matrix. As the lexicographical process $(\tilde{\mathcal{I}}_t^N)_{t\in\mathbb{R}_+}$ keeps the structure of the tree over time, this is still possible by adding the distance between the root and each extremity to the distance matrix. Indeed, when there are $N\tilde{X}_t^N$ extremities, there are $N\tilde{X}_t^N - 1$ branching (internal node) and therefore there are $2(N\tilde{X}_t^N - 1)$ branches (since our tree is binary). So, we have $2(N\tilde{X}_t^N - 1)$ unknowns and the modified distance matrix gives $\frac{N\tilde{X}_t^N(N\tilde{X}_t^N-1)}{2} + N\tilde{X}_t^N > 2(N\tilde{X}_t^N - 1)$ equations. We can note that we need to modify the distance matrix (by adding the distance between the root and each extremity) because we have $\frac{N\tilde{X}_t^N(N\tilde{X}_t^N-1)}{2} > 2(N\tilde{X}_t^N - 1)$ only when $N\tilde{X}_t^N > 4$. If $N\tilde{X}_t^N \leq 4$, then the distance matrix (without modification) does not provide enough equations.

A final improvement, difficult this time, would be to take into account the fact that binding can occur on all branches of the tree and not only on the leaves. Thus, fragmentation (or detachment of an Arp2/3 complex) could make an entire sub-tree disappear, not just a branch. This could therefore lead to a severe decrease in the number of extremities. Similarly, a branching event could lead to the modification of an entire sub-tree. In the regime of many extremities/short branches that we considered, "internal" binding seems less relevant as the proteins could have to reach the central part of a dense polymer for such an event to occur. However, it may be an important phenomenon to take into account in other biological situations where we expect long but less bushy branched polymers.

We now present some ideas to add this last improvement to the model. As we want to keep the same modelling, we want to continue working with leaves only. So we have to find a way to reach the internal branches when we only have information about the leaves.

When a binding event occurs, we start by choosing the leaf i that will be impacted. This can be done by using:

- the uniform distribution (as we did in Section 3.2.2),
- a probability distribution that depends on the heigh of the leaf (for example, we may choose the probability:

$$\mathbb{P}\,(\text{binding on the } i\text{-th branch}) = \frac{|\iota^{(i)}|}{\sum_{j=1}^{N\tilde{X}_t^N} |\iota^{(j)}|},$$

  where $|\iota^{(i)}|$ is the heigh of the leaf i),
- a probability distribution that depends on the distance (in number of monomers) between the leaf and the root (in that case, the distance between the root to each extremity should be added to the distance matrix as proposed previously).

We then choose on which branch under the leaf i the binding will take place. As the heigh of leaf i, noted $|\iota^{(i)}|$ corresponds to the number of internal nodes under the leaf i, we choose the branch $b \in \{0, 1, 2, \cdots, |\iota^{(i)}|\}$ using either:

- the uniform distribution,
- a probability distribution that depends on the length of the branches (in the same way as we proposed in the previous improvement).

To define the new state of the lexicographic process, we will use a truncation of the element $\iota^{(i)}$. Since the heigh of leaf $\iota^{(i)}$ is $|\iota^{(i)}|$, for any $k \in \{0, 1, 2, \cdots, |\iota^{(i)}|\}$, we can find two elements $\iota_k^{(i)}$ and $\iota_{k,prefix}^{(i)}$ with $|\iota_k^{(i)}| = k$, and two elements $\iota_k'^{(i)}$ and $\iota_{k,suffix}^{(i)}$ with $|\iota_k'^{(i)}| = k$ such that:

$$\iota^{(i)} = c\left(\iota_{k,prefix}^{(i)}, \iota_k^{(i)}\right) = c\left(\iota_k'^{(i)}, \iota_{k,suffix}^{(i)}\right),$$

where c is the concatenation defined in Section 3.2.1. We can then define a prefix truncation function t_{prefix} such that, for each $k \in \{0, 1, 2, \cdots, |\iota^{(i)}|\}$,

$$t_{prefix}\left(\iota^{(i)}, k\right) = \iota_{k,prefix}^{(i)},$$

and a suffix truncation function t_{suffix} such that, for each $k \in \{0, 1, 2, \cdots, |\iota^{(i)}|\}$,

$$t_{suffix}\left(\iota^{(i)}, k\right) = \iota_{k,suffix}^{(i)}.$$

If a binding event occurs on the branch b, under the leaf i, the element $t_{prefix}\left(\iota^{(i)}, b\right)$ and all its offspring will be modified. We group the elements that will be impacted in a space $\mathcal{O}_{i,b}$ defined by:

$$\mathcal{O}_{i,b} = \left\{ \text{leaf } \iota \mid \iota \cap t_{prefix}\left(\iota^{(i)}, b\right) = t_{prefix}\left(\iota^{(i)}, b\right) \right\}.$$

To describe the evolution of the different processes, we will use the discrete time construction of the processes presented in Section 3.

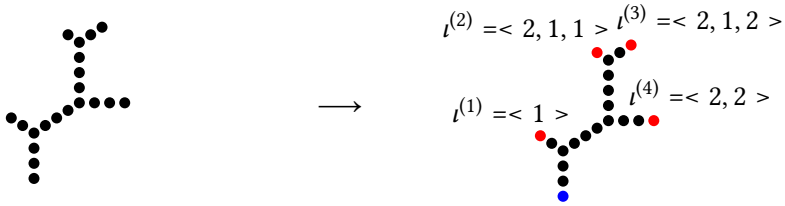
For $n \in \mathbb{N}$, at the n–th binding of a protein on branch b under leaf i,

- if it is a fragmentation, we assume here that the whole branch (and its offspring) disappears, therefore we have:
$$\begin{cases} X_n^N = X_{n-1}^N - \#\mathcal{O}_{i,b}, \\ \mathcal{I}_n^N = \mathcal{I}_{n-1}^N \backslash \mathcal{O}_{i,b}, \end{cases}$$
and columns and rows of the distance matrix that correspond to elements of $\mathcal{I}_{n-1}^N \cap \mathcal{O}_{i,b}$ are removed.

- If it is a branching, we assume here that the binding occurs on the first monomer of the branch and create a new branch on the right. Therefore, we have:
$$\begin{cases} X_n^N = X_{n-1}^N + 1, \\ \mathcal{I}_n^N = \mathcal{I}_{n-1}^N \backslash \mathcal{O}_{i,b} \cup \left\{ c\left(t_{prefix}\left(\iota^{(i)}, b\right), 2\right) \right\} \cup \mathcal{O}'_{i,b}, \end{cases}$$
where:
$$\mathcal{O}'_{i,b} = \left\{ \iota \mid \exists \iota' \in \mathcal{I}_{n-1}^N \cap \mathcal{O}_{i,b} \text{ such that } \iota = c\left(c\left(t_{prefix}\left(\iota^{(i)}, b\right), 1\right), t_{suffix}\left(\iota', |\iota^{(i)}| - b\right)\right) \right\}.$$
The pseudo-distance matrix can be updated since we can deduce the structure of the tree from the leaf indices (or at least an equivalent class) and we can find the length of each branch of the tree.

Example 2.9.1
Let's see an example. We use Examples 2.3.5 and 2.3.6.

If the branched actin polymer is as follows:



there are $X = 4$ extremities, the set of extremities is:

$$\mathcal{I} = \{< 1 >, < 2,1,1 >, < 2,1,2 >, < 2,2 >\} = \{\iota^{(1)}, \iota^{(2)}, \iota^{(3)}, \iota^{(4)}\},$$

and the pseudo-distance matrix with the distance between the root and each extremity is given by:

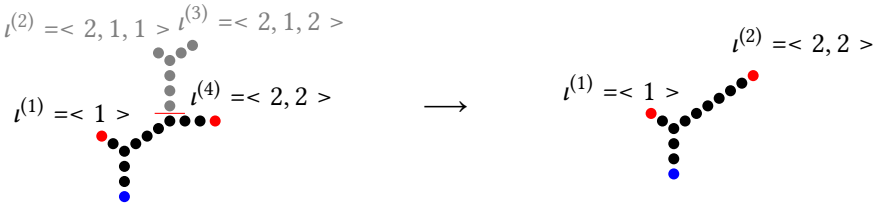
$$\mathcal{D} = \begin{pmatrix} 0 & 6 & 13 & 14 & 11 \\ 6 & 0 & 12 & 13 & 10 \\ 13 & 12 & 0 & 4 & 9 \\ 14 & 13 & 4 & 0 & 10 \\ 11 & 10 & 9 & 10 & 0 \end{pmatrix}$$

If we choose $i = 3$, then the binding event will occurs on leaf $\iota^{(3)} =< 2,1,2 >$.

$|\iota^{(3)}| = 3$, so there are 3 internal nodes (and so 4 branches) under $\iota^{(3)}$. If we choose $b = 1$, the binding will occurs on the first branch under $\iota^{(3)}$, in other words, just after the node $t_{prefix}\left(\iota^{(3)}, b+1\right) =< 2 >$. The elements that will be impacted are therefore:

$$\mathcal{O}_{3,1} = \left\{ \text{ leaf } \iota \mid \iota \cap t_{prefix}(< 2,1,2 >, 1) = t_{prefix}(< 2,1,2 >, 1) =< 2,1 > \right\} = \{< 2,1,1 >, < 2,1,2 >\}.$$

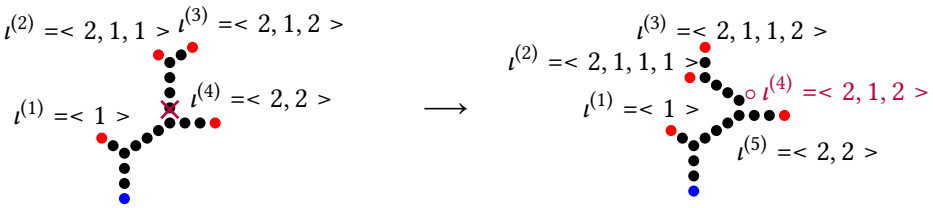
- If a fragmentation occurs, we have:



After this fragmentation, there are $X' = X - \#\mathcal{I} \cap \mathcal{O}_{3,1} = 4 - 2 = 2$ extremities, indexed by $\mathcal{I}' = \mathcal{I} \backslash \mathcal{O}_{3,1} = \{< 1 >, < 2,2 >\}$ and the pseudo-distance matrix becomes:

$$\mathcal{D} = \begin{pmatrix} 0 & 6 & 13 & 14 & 11 \\ 6 & 0 & 12 & 13 & 10 \\ 13 & 12 & 0 & 4 & 9 \\ 14 & 13 & 4 & 0 & 10 \\ 11 & 10 & 9 & 10 & 0 \end{pmatrix} \longrightarrow \mathcal{D}' = \begin{pmatrix} 0 & 6 & 11 \\ 6 & 0 & 10 \\ 11 & 10 & 0 \end{pmatrix}$$

- If a branching occurs, we have:



After this event, there are $X' = X + 1 = 5$ extremities, indexed by:

$$\mathcal{I}' = \mathcal{I} \backslash \mathcal{O}_{3,1} \cup \left\{ c\left(t_{prefix}(< 2,1,2 >, 1), 2\right)\right\} \cup \mathcal{O}'_{3,1}$$

with:

- $\mathcal{I} \backslash \mathcal{O}_{3,1} = \{< 1 >, < 2,2 >\}$,
- $c\left(t_{prefix}(< 2,1,2 >, 1), 2\right) = c(< 2,1 >, 2) =< 2,1,2 >$,
- $$\begin{aligned}\mathcal{O}'_{3,1} &= \left\{ \iota \mid \exists \iota' \in \mathcal{I} \cap \mathcal{O}_{3,1} \text{ such that } \iota = c\left(c\left(t_{prefix}(< 2,1,2 >, 1), 1\right), t_{suffix}\left(\iota', 2\right)\right)\right\} \\ &= \{< 2,1,1,1 >, < 2,1,1,2 >\}.\end{aligned}$$

The pseudo-distance matrix becomes:

$$\mathcal{D} = \begin{pmatrix} 0 & 6 & 13 & 14 & 9 & 11 \\ 6 & 0 & 12 & 13 & 8 & 10 \\ 13 & 12 & 0 & 4 & 5 & 9 \\ 14 & 13 & 4 & 0 & 6 & 10 \\ 9 & 8 & 5 & 6 & 0 & 5 \\ 11 & 10 & 9 & 10 & 5 & 0 \end{pmatrix}.$$

10 - Simulations

In this section, we present some simulations in order to visualise the evolution of a tree defined by the model described in this chapter. We also consider the modifications presented in Section 9 (on perspectives), one by one, in order to determine which ones have a significant impact and should be considered first. These few simulations also allow us to check that our assumptions are appropriate.

For each simulation, we start with a simple initial tree chosen arbitrarily and represented in Figure 2.2. In addition to the tree representation (graph on the left), we plot the height function associated with the initial tree (graph in the middle) and the length (in number of monomers) of each branch (graph on the right).

As the number of events that occur in a time interval is random and can vary significantly, we will observe each simulation after 1000 events. For each set of parameter values, we ran the simulation about ten times and chose the one that seemed to corresponds to an average behaviour. In the following, only one behaviour is presented, but it should be noted that, depending on the parameter values, the results are more or less variable.

10.1 - Influence of the Parameters

Firstly, we want to observe the influence of the parameters on the polymer and more precisely on its topology, on the number of monomers included in the polymer and on the number of Arp2/3 complex attached to the polymer. We recall that there are three parameters in the model developed in this chapter:

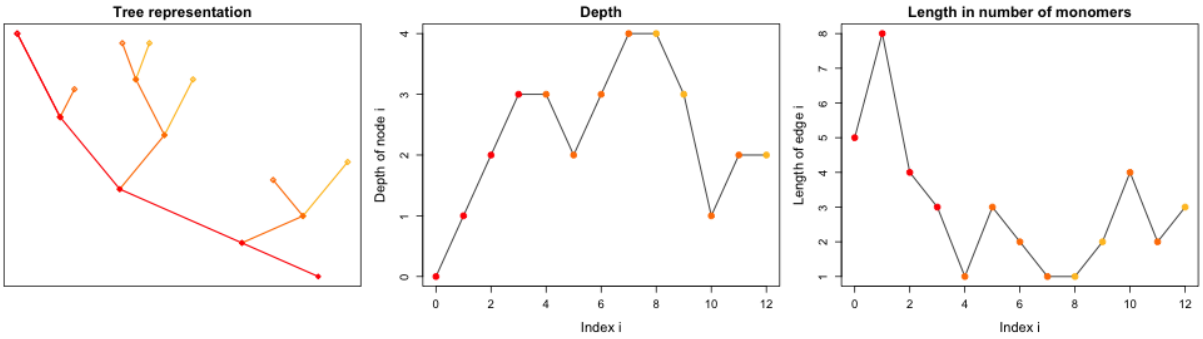


Figure 2.2 Initial tree, chosen arbitrarily. Left: representation of the tree as a graph. Middle: height function associated to the tree. Right: length in number of monomers of each branch (the branch i corresponds to the branch that connects node i to its parent).

- λ^+: rate of spontaneous elongation
- β: binding rate of accessory proteins (which leads to branching or fragmentation)
- a: advantage for one of the accessory proteins, if $a > 0$ (resp. $a < 0$) branching (resp. fragmentations) have an advantage.

Figure 2.3 presents a case where elongation and binding of proteins take place at the same rate. Since $a = 0$, when a protein binding occurs, there is as much chance for it to be a branching as to be a fragmentation.

This figure (and the following ones) is divided into six graphs. On the top three graphs we plot the representation of the tree as a graph, the height function associated with the tree and the length in number of monomers of each branch. These three graphs provide an idea of the topology of the tree. On the bottom three graphs, we plot the evolution of the number of monomers included in the polymer, the evolution of the number of internal nodes, which corresponds to the number of Arp2/3 complexes attached to the polymer and the evolution of the rescaled number of extremities, which corresponds to the evolution of the rescaled process $\tilde{X}^N$. These three graphs provide information on the dynamics of actin and accessory proteins.

After 1000 events, we observe that the tree has about 350 monomers and 250 nodes. The majority of the branches have a length between 0 and 1; the rest are between 2 and 4. The majority of the extremities appear to be at depth 10. This case is defined as the base case for this section. In the following, we will note within parentheses the variations between this base case and the observed case.

In Figure 2.4, we choose $a = 100$, which means that the branching has an advantage (the probability of branching is then 0.7). We choose a large enough value for a to observe a significant change after a time of the order of 10^{-4}. If we choose $a = 1$, we will have to wait a time of the order of 10^{-2} to observe a similar variation. After 1000 events the tree has about 400 (+50 compared to base case) monomers and 300 (+50) nodes. Elongation also has an advantage in this case as there are more extremities. The distribution of branch lengths is similar to the previous case. However, the average depth of the extremities is slightly higher.

In Figure 2.5, we now choose $a = -100$, which means that the fragmentation has an advantage (the probability of fragmentation is then 0.7). After 1000 events the tree has about 300 (-50) monomers and 200 (-50) nodes. Thus, we find the same variation as in the previous case where branching had an advantage. It can also be noted that it took longer to achieve this result. As fragmentation limits the number of extremities, events become less frequent and it takes longer for the advantage given to

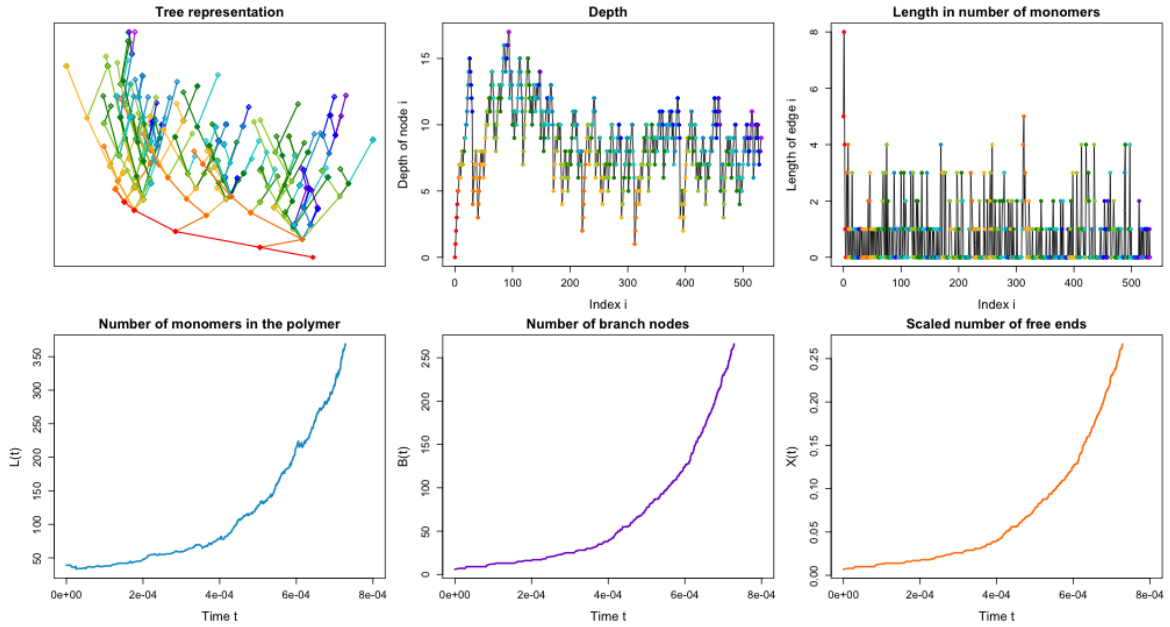


Figure 2.3 Evolution of the polymer with $N = 1000$, $\lambda^+ = 10$, $\beta = 10$ and $a = 0$. Observation after 1000 events, corresponding to $t = 7.28 \times 10^{-4}$. Top left: representation of the tree as a graph. Top middle: height function associated with the tree. Top right: length in number of monomers of each branch. Bottom left: evolution of the number of monomers included in the polymer. Bottom middle: evolution of the number of internal nodes. Bottom right: evolution of $\tilde{X}^N$ the rescaled number of leaves.

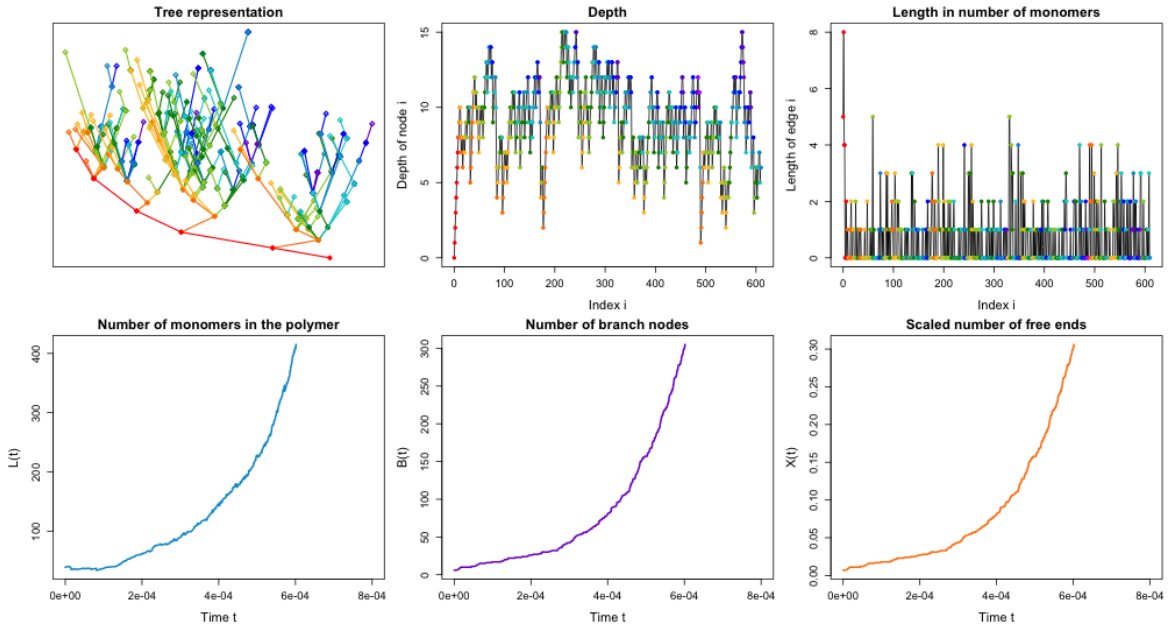


Figure 2.4 Evolution of the polymer with $N = 1000$, $\lambda^+ = 10$, $\beta = 10$ and $a = 100$. Observation after 1000 events, corresponding to $t = 6.02 \times 10^{-4}$. Top left: representation of the tree as a graph. Top middle: height function associated with the tree. Top right: length in number of monomers of each branch. Bottom left: evolution of the number of monomers included in the polymer. Bottom middle: evolution of the number of internal nodes. Bottom right: evolution of $\tilde{X}^N$ the rescaled number of leaves.

fragmentation to become significant. The majority of branches still have a length between 0 and 1 but longer branches (with a length between 3 and 7) appear. Fragmentation therefore limits the amount of actin polymerised but allows the appearance of longer branches by limiting branching.

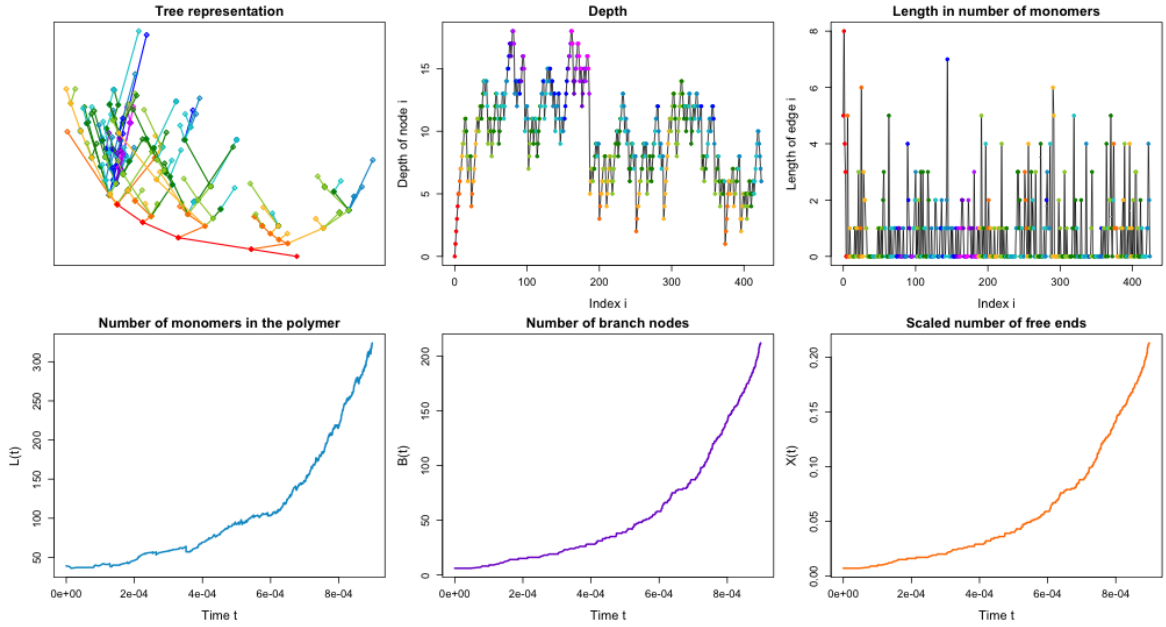


Figure 2.5 Evolution of the polymer with $N = 1000$, $\lambda^+ = 10$, $\beta = 10$ and $a = -100$. Observation after 1000 events, corresponding to $t = 8.98 \times 10^{-4}$. Top left: representation of the tree as a graph. Top middle: height function associated with the tree. Top right: length in number of monomers of each branch. Bottom left: evolution of the number of monomers included in the polymer. Bottom middle: evolution of the number of internal nodes. Bottom right: evolution of $\tilde{X}^N$ the rescaled number of leaves.

In Figure 2.6, elongation is favoured (×2). After 1000 events, the tree has about 500 (×1.4) monomers and 150 (×0.6) nodes. The fact that there are fewer nodes here is related to the fact that we are looking at a fixed number of events. If we look at this case at time $t \approx 7.28 \times 10^{-4}$ as in Figure 2.3, we observe that there is twice as much polymerised actin and the same number of branches. The branches are therefore on average twice as long. Elongation does not influence the topology of the tree.

10.2 - Effect of Capping Proteins

We now consider the first improvement proposed in Section 9, which is the inclusion of capping proteins. A capping protein binds to a free extremity of the tree at a constant rate Φ_{Cap} and blocks elongation (branching and fragmentation remain possible).

Figure 2.7 presents a case where elongation and capping take place at the same rate. After 1000 events, the tree has about 200 (-150) monomers and 250 (=) nodes. Although the tree appears less bushy, the height function (and average depth) remains similar. This is due to the fact that many of the capped branches have a zero length. Regarding the distribution of branch lengths, it can be seen that the large majority of branches have a zero length. We can also note that when capping takes place at the same rate as elongation, about half of the extremities are capped.

Our first idea to limit the number of zero-length branches was to forbid the binding of a capping protein to a zero-length branch. This seems biologically relevant since these proteins bind to actin and not to Arp2/3 complexes. We can see in Figure 2.8 that the tree obtained with this constraint appears a

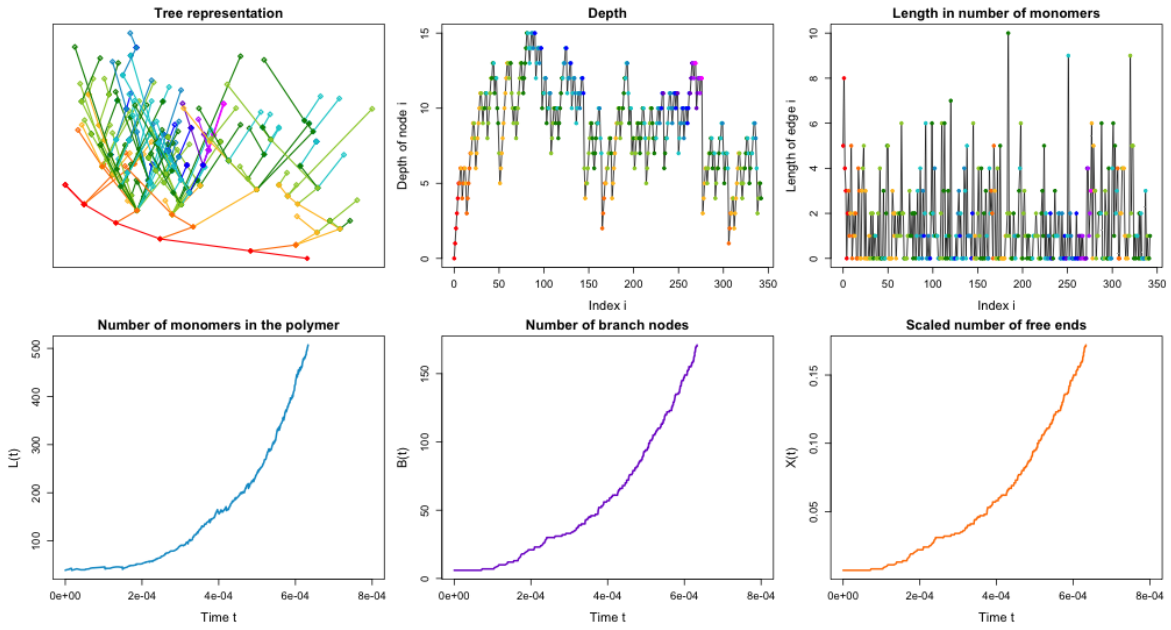


Figure 2.6 Evolution of the polymer with $N = 1000$, $\lambda^+ = 20$, $\beta = 10$ and $a = 0$. Observation after 1000 events, corresponding to $t = 6.33 \times 10^{-4}$. Top left: representation of the tree as a graph. Top middle: height function associated with the tree. Top right: length in number of monomers of each branch. Bottom left: evolution of the number of monomers included in the polymer. Bottom middle: evolution of the number of internal nodes. Bottom right: evolution of $\tilde{X}^N$ the rescaled number of leaves.

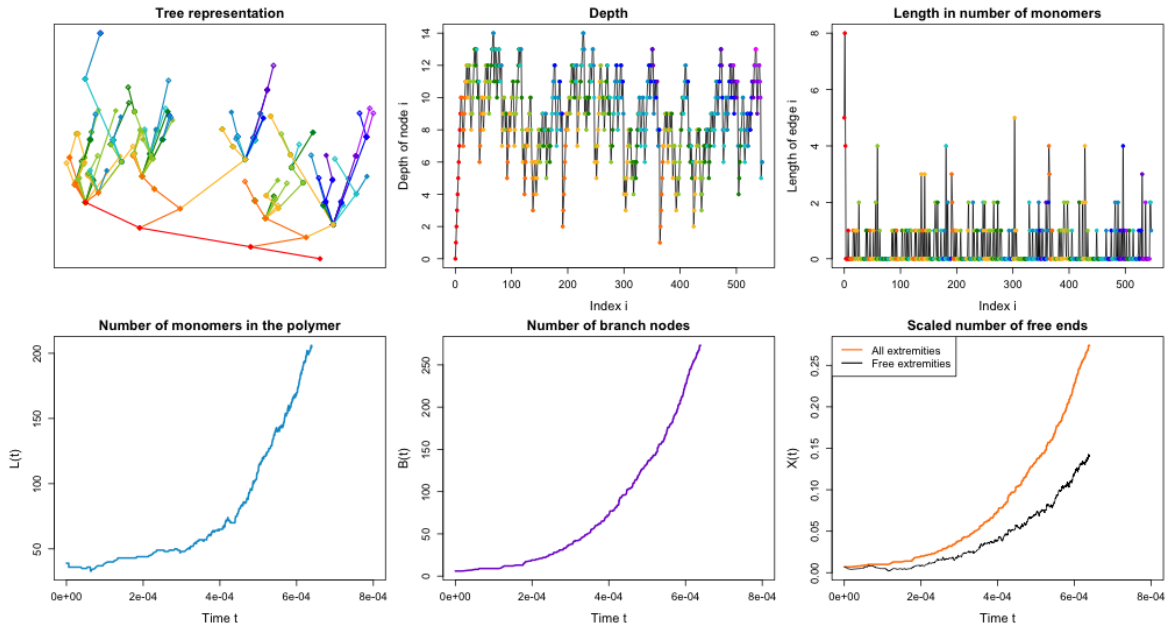


Figure 2.7 Evolution of the polymer with $N = 1000$, $\lambda^+ = 10$, $\beta = 10$, $a = 0$ and $\Phi_{Cap} = 10$. Observation after 1000 events, corresponding to $t = 6.39 \times 10^{-4}$. Top left: representation of the tree as a graph. Top middle: height function associated with the tree. Top right: length in number of monomers of each branch. Bottom left: evolution of the number of monomers included in the polymer. Bottom middle: evolution of the number of internal nodes. Bottom right: evolution of $\tilde{X}^N$ the rescaled number of leaves and evolution of free ends (not capped).

little more bushy than in Figure 2.7 and that there are fewer internal branches of zero length, however the tree is still not homogeneous. This must be due to the fact that branching can occur on a zero length extremity. In this case, about one third of the extremities are capped.

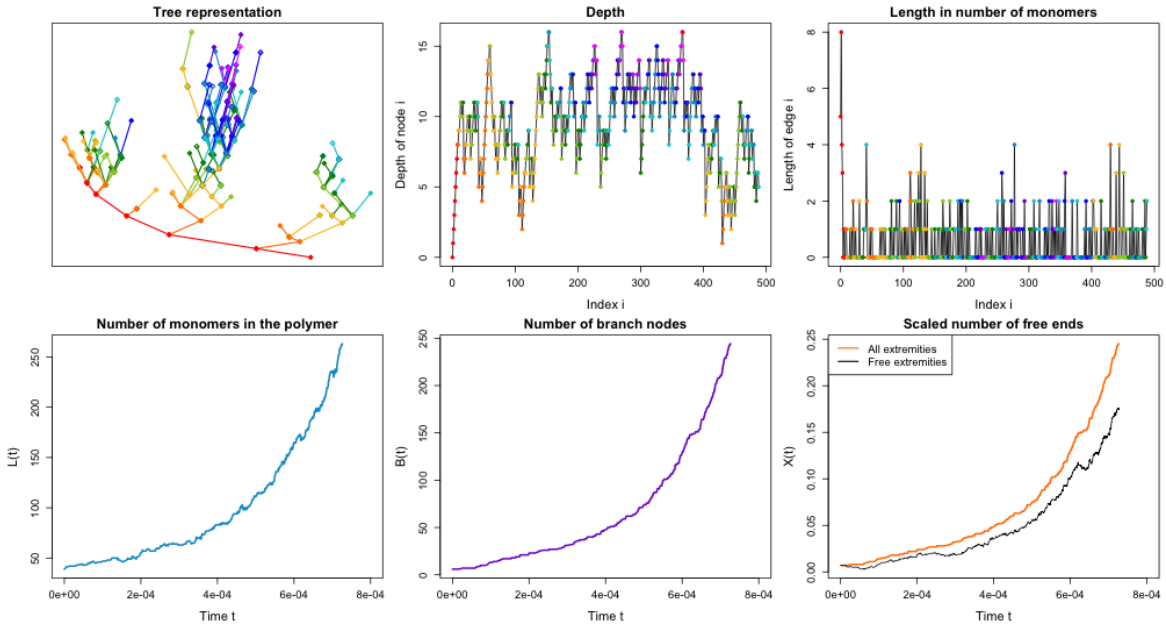


Figure 2.8 Evolution of the polymer with $N = 1000$, $\lambda^+ = 10$, $\beta = 10$, $a = 0$ and $\Phi_{Cap} = 10$ forbidding the capping of branches of zero length. Observation after 1000 events, corresponding to $t = 7.27 \times 10^{-4}$. Top left: representation of the tree as a graph. Top middle: height function associated with the tree. Top right: length in number of monomers of each branch. Bottom left: evolution of the number of monomers included in the polymer. Bottom middle: evolution of the number of internal nodes. Bottom right: evolution of $\tilde{X}^N$ the rescaled number of leaves and evolution of free ends (not capped).

As this first idea did not allow us to obtain a dense tree, we tried to limit the capping by choosing a rate lower than the elongation rate (while keeping the rate large enough for the capping to have a significant effect). We can see in Figure 2.9 that by dividing the capping rate by half, we obtain a tree that seems more bushy than in Figure 2.7, more homogeneous than in Figure 2.8, and whose distribution of branch lengths is similar to the base case. In this case, about one third of the extremities are capped.

10.3 - Non-uniform Choice of Branches

This time, the capping proteins are no longer taken into account, but the model is slightly modified so that the choice of branch during branching or fragmentation is no longer uniform but depends on the branch length. The longer the branch of an extremity, the more likely it is that a protein will bind to it. It can also be noted that this non-uniform choice makes branching or fragmenting of a zero length branch impossible unless all branches are of length zero.

In Figure 2.10, using the same parameter values as in the base case, we obtain after 1000 events a tree that has about 300 (-50) monomers and 250 (=) nodes. It can be seen that the non-uniform choice limits the amount of polymerised actin by favouring the fragmentation of long branches (which implies a greater loss of monomers). It also results in a fairly dense and uniform tree: almost all the non-zero length branches are of length 1. The average depth seems similar.

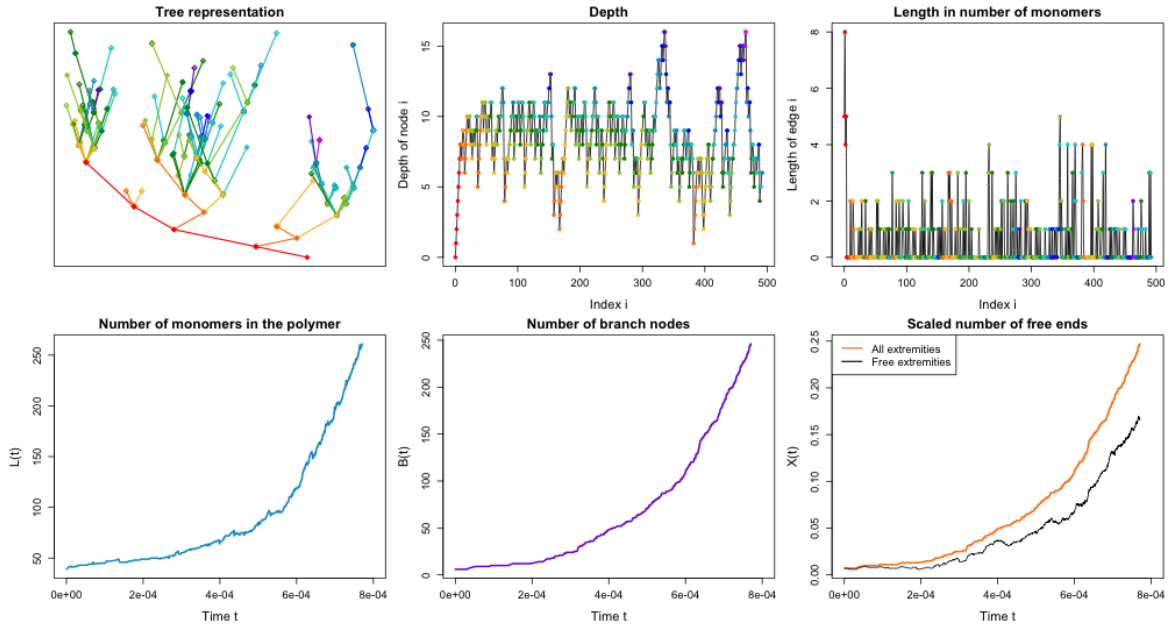


Figure 2.9 Evolution of the polymer with $N = 1000$, $\lambda^+ = 10$, $\beta = 10$, $a = 0$ and $\Phi_{Cap} = 5$. Observation after 1000 events, corresponding to $t = 7.71 \times 10^{-4}$. Top left: representation of the tree as a graph. Top middle: height function associated with the tree. Top right: length in number of monomers of each branch. Bottom left: evolution of the number of monomers included in the polymer. Bottom middle: evolution of the number of internal nodes. Bottom right: evolution of $\tilde{X}^N$ the rescaled number of leaves and evolution of free ends (not capped).

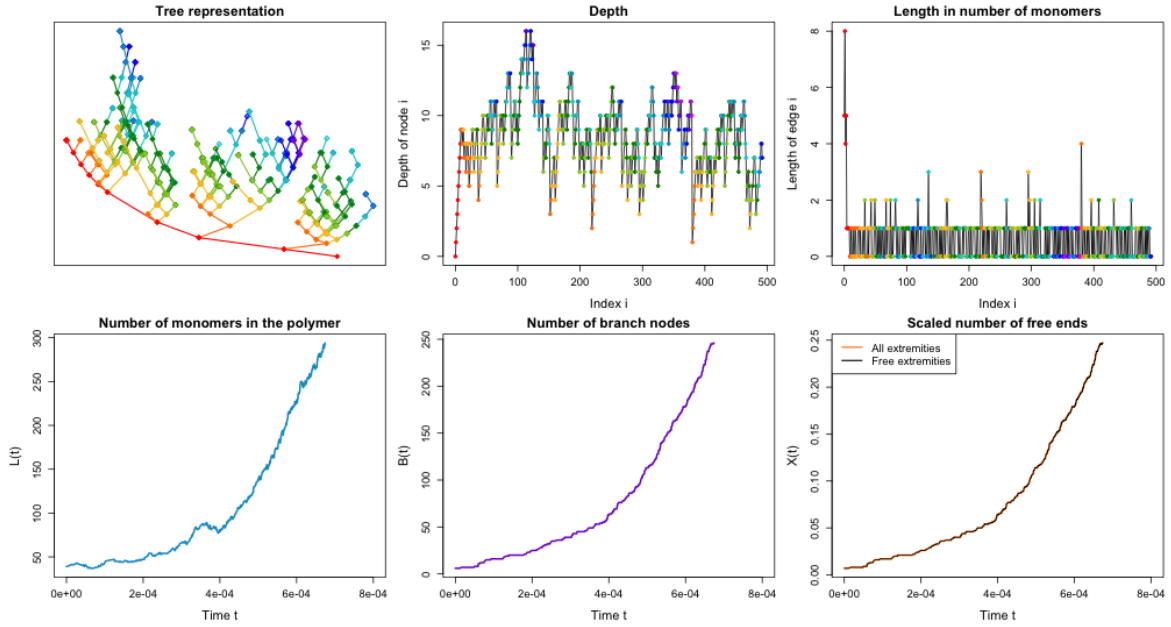


Figure 2.10 Evolution of the polymer with $N = 1000$, $\lambda^+ = 10$, $\beta = 10$, $a = 0$ and $\Phi_{Cap} = 0$ with non-uniform choice of branches. Observation after 1000 events, corresponding to $t = 6.74 \times 10^{-4}$. Top left: representation of the tree as a graph. Top middle: height function associated with the tree. Top right: length in number of monomers of each branch. Bottom left: evolution of the number of monomers included in the polymer. Bottom middle: evolution of the number of internal nodes. Bottom right: evolution of $\tilde{X}^N$ the rescaled number of leaves and evolution of free ends (not capped).

10.4 - Branching and Fragmentation Allowed on Internal Branches

Finally, we consider the last case proposed in Section 9 where branching and fragmentation do not only take place at the extremities but are possible on all branches of the tree. We keep the choice of branch that depends on the length of the branches, we then choose the monomer on which the binding will take place uniformly at random among the monomers of the chosen branch. Capping proteins are not taken into account in this case.

We can see in Figure 2.11 that using again the same parameter values as in the base case, we obtain after 100 events a tree with only about 10 monomers and about 5 nodes. The branches are mainly of length 1 and the tree is straggly. As fragmentation can occur anywhere in the tree, the number of actin monomers and the number of branches fluctuate a lot. We have seen with the previous case that the non-uniform choice of branch leads to a uniformity of branch length. After some time, all the branches of non-zero length are then of size 1 and the choice of branch is still uniform: there is therefore as much chance to choose a branch close to an extremity (and therefore of losing few monomers) as to choose a branch close to the root (and therefore lose almost the entire tree). In order for this improvement to obtain a tree of reasonable length and to be biologically relevant, branching must therefore be favoured. The values in the literature (see Table 3.2) suggest that the branching rate is about 10^4 times greater than the fragmentation rate, so we choose a value for a that is large enough to reach this order of magnitude.

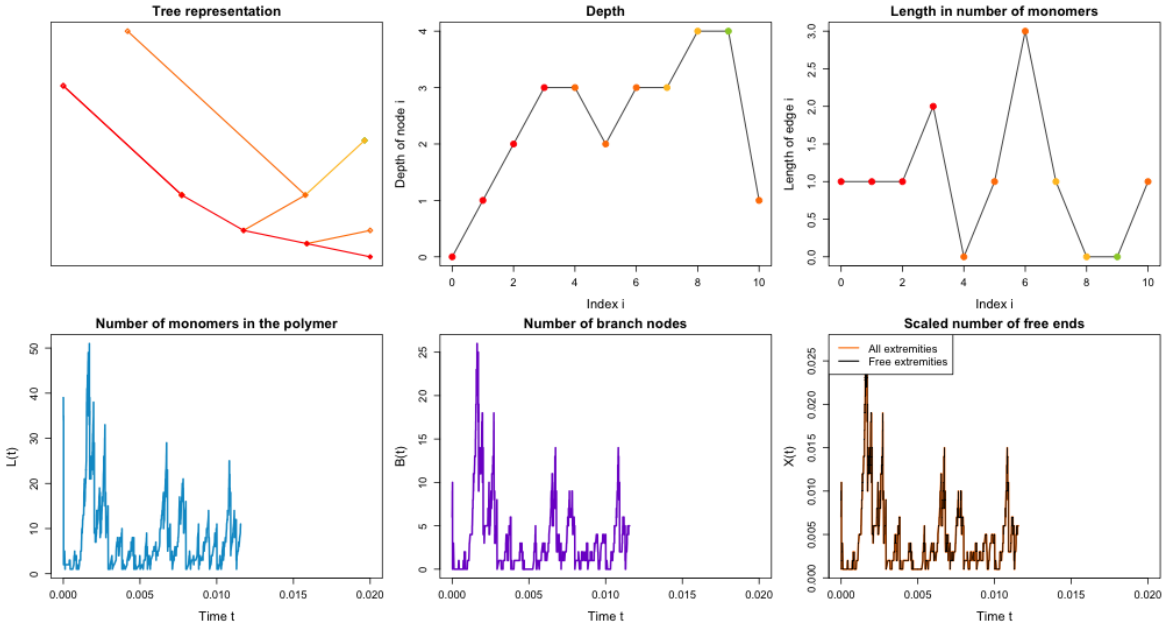


Figure 2.11 Evolution of the polymer with $N = 1000$, $\lambda^+ = 10$, $\beta = 10$, $a = 0$ and $\Phi_{Cap} = 0$ with branching and fragmentations allowed on internal branches. Observation after 1000 events, corresponding to $t = 1.15 \times 10^{-2}$. Top left: representation of the tree as a graph. Top middle: height function associated with the tree. Top right: length in number of monomers of each branch. Bottom left: evolution of the number of monomers included in the polymer. Bottom middle: evolution of the number of internal nodes. Bottom right: evolution of $\tilde{X}^N$ the rescaled number of leaves and evolution of free ends (not capped).

By choosing $a = 499$, which corresponds to a probability of branching equal to 0.999, we obtain in Figure 2.12 a very dense polymer. After 1000 events, the tree has 600 (+250) monomers and 400 (+150) nodes. The large majority of the branches are of length 1.

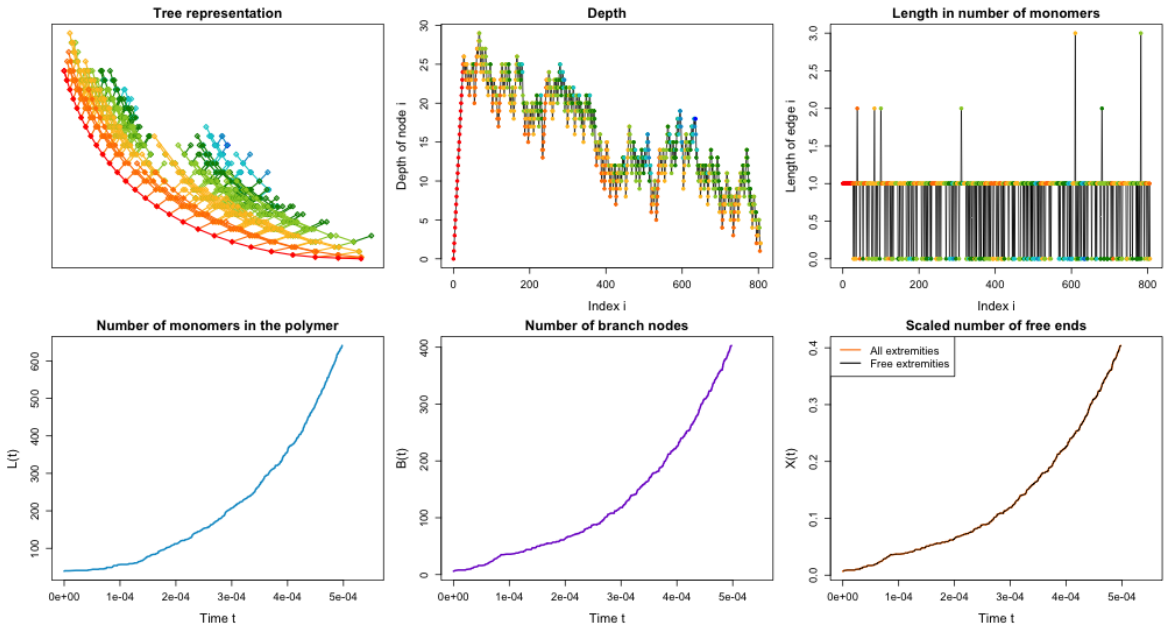


Figure 2.12 Evolution of the polymer with $N = 1000$, $\lambda^+ = 10$, $\beta = 10$, $a = 499$ and $\Phi_{Cap} = 0$ with branching and fragmentations allowed on internal branches. Observation after 1000 events, corresponding to $t = 4.98 \times 10^{-4}$. Top left: representation of the tree as a graph. Top middle: height function associated with the tree. Top right: length in number of monomers of each branch. Bottom left: evolution of the number of monomers included in the polymer. Bottom middle: evolution of the number of internal nodes. Bottom right: evolution of $\tilde{X}^N$ the rescaled number of leaves and evolution of free ends (not capped).

10.5 - Conclusion

The model proposed in this chapter seems to produce satisfactory trees: dense with relatively short branches. The amount of polymerised actin can be regulated by the rate of elongation, but this also affects the length of the branches. In order to better limit polymerisation, the capping proteins must be taken into account. The non-uniform choice of branches allows for a more uniform branch length. To obtain a truly homogeneous tree, with as many branches at the root as close to the leaves, it is necessary to take the branching on the whole tree into account. In addition, the inclusion of capping proteins and non-uniform branching throughout the tree significantly modifies the amount of polymerised actin and the number of branches; so it is important to take these into account too.

Chapter III

Numerical Exploration of Models

Contents

1 - Overview

This chapter proposes a numerical exploration of the models of Chapter I. In Section 2 we present a qualitative study of the different behaviours that can be encountered when observing a single actin polymer. This allows us to learn a lot about the role of the different proteins involved. In Section 3, we take up the cases encountered in the previous section, but this time considering a population of polymers, and we observe the differences. This allows us to identify the effects of competition between polymers for free monomers.

Section 4 is devoted to a new model, corresponding to an adaptation of the model for a population of polymers of Chapter I and taking into account all the accessory proteins introduced in this work. This later model, which is more complex, is studied only through simulations and aims to understand

the interactions between linear and branched polymers. A qualitative study, left for future work, would also allow us to determine which parameters are robust and what is the influence of each parameter on our variables of interest such as: the average length of polymers, the distribution of polymer lengths, the amount of polymerised actin, the amount of branched polymers, the average number of branches per polymer or the average number of free extremities per polymer.

2 - Model for a Single Linear Actin Polymer

In this section, we present the numerical exploration that led to the results of Section 5.1 of Chapter I. These additional simulations allow us to present the different behaviours of the process but also to better understand the role of formins and profilin.

2.1 - Behaviour without Accessory Proteins

When there is no accessory protein (i.e. $\Phi_P = 0$, $\Phi_F^+ = 0$ and $\Phi_F^- = 0$) the system is very simple since the only possible events are spontaneous elongation and depolymerisation. Using Theorem 1.2.1, we find the fluid limit and the equilibrium state given by:

$$\begin{cases} \left(\frac{\lambda^-}{\lambda^+}, 1 - \frac{\lambda^-}{\lambda^+}, 0\right), & \text{if } t_0 = +\infty, \\ (1,0,0), & \text{otherwise.} \end{cases}$$

In this case, we can note that if $\lambda^- \geq \lambda^+$, then $1 - \frac{\lambda^-}{\lambda^+} \leq 0$ and therefore $t_0 < +\infty$. As we want to obtain a non-zero polymer length at equilibrium, we only consider the supercritical case (i.e. $\lambda^+ > \lambda^-$).

Figure 3.1 displays the evolution of the length of a single actin polymer in two supercritical cases. In addition to the convergence of the polymer length to $1 - \frac{\lambda^-}{\lambda^+}$, we can observe that the closer $\lambda^+ - \lambda^-$ is to 0, the greater the oscillations around the fluid limit, and therefore the more the system is subject to stochasticity.

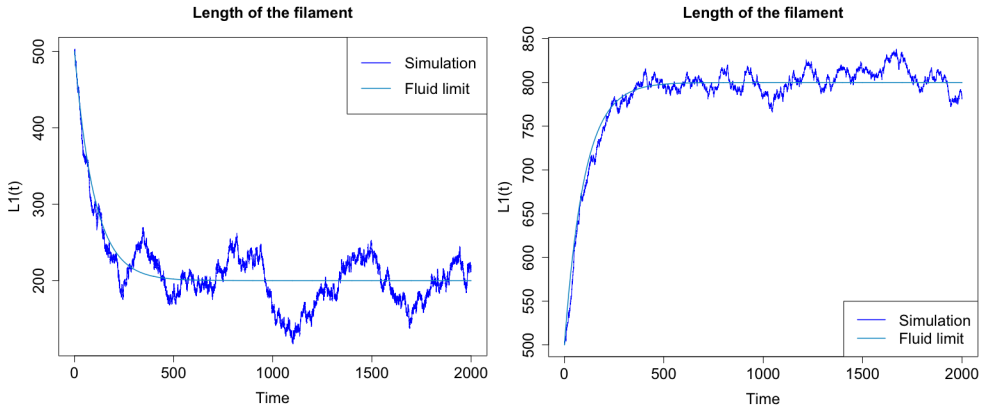
In the following, we have chosen $\lambda^+ = 10$ and $\lambda^- = 2$ in order to get the simulations close to the fluid limit. However, it can be highlighted that if we want a result closer to biological results, about 30% of actin monomers should be in a polymerised form (filaments or complexes) in the limit and for that, we should choose $\lambda^+ = 10$ and $\lambda^- = 7$.

2.2 - Effect of Profilin

In order to observe the effect of the profilin, we now choose $\Phi_P > 0$. Two cases can be distinguished here:

1. The polymer is simple (i.e. without formin), the system therefore remains in normal mode.

2. The polymer is associated with a formin, the system therefore remains in fast mode.



(A) $\lambda^- = 8$ (B) $\lambda^- = 2$

Figure 3.1 Evolution of the length of the polymer with $N = 1000$, $\lambda^+ = 10$, $\lambda_F^+ = 0$, $\Phi_P = 0$, $\Phi_F^+ = 0$ and $\Phi_F^- = 0$ for two values of λ^-. Initially, there are a (unique) polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. On each graph, we plot a single realisation of the stochastic system for $N = 1000$ and the fluid limit given in Theorem 1.2.1 (i.e. N times $l_1(t)$).

2.2.1 - Normal mode

In presence of profilin, if the system remains in the normal mode, there is only one possible end: the disappearance of the polymer. Indeed, when the polymer is simple, it does not consume G-actin/profilin complex. The profilin therefore sequesters the free monomers and the polymer does not have enough resources to survive. This behaviour is depicted in Figure 3.2. This result is proved in Section 5.1 in Chapter I. It can be noted that the larger Φ_P is, the faster the disappearance. In the left-hand graph of Figure 3.2, the rupture corresponds to the time t_0 at which the polymer "size" cancels. It can also be noted that despite the depolymerisation (decreasing polymer length in the middle graph), the number of free monomers does not increase since the creation of G-actin/profilin complexes is too fast.

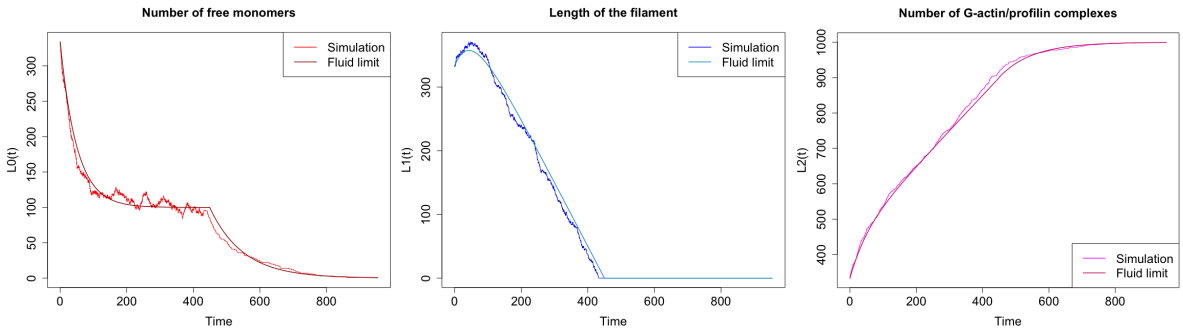
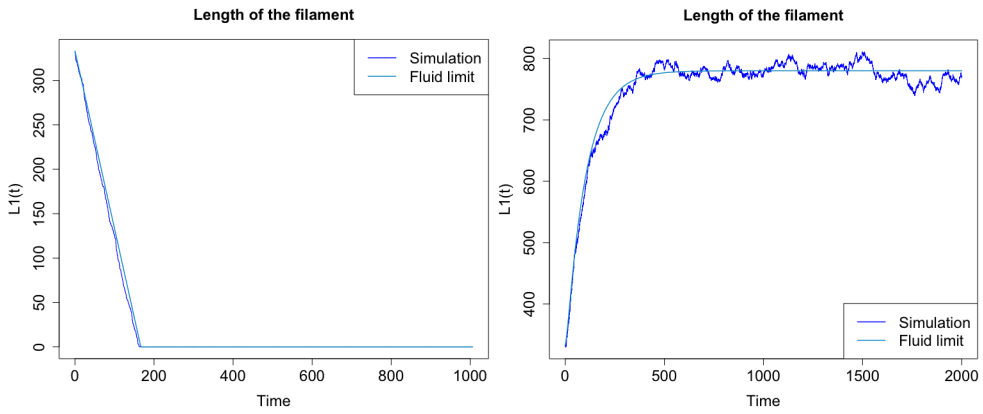


Figure 3.2 Evolution of the system described by the model for a single actin polymer with $N = 1000$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 0$, $\Phi_P = 10$, $\Phi_F^+ = 0$ and $\Phi_F^- = 0$. Initially, there are a (unique) polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. Left: evolution of the number of free monomers. Middle: evolution of the length of the polymer. Right: evolution of the number of G-actin/profilin complexes. On each graph, we plot a single realisation of the stochastic system for $N = 1000$ and the fluid limit given in Theorem 1.2.1 (i.e. N times $l_1(t)$).

2.2.2 - Fast mode

If the system remains in fast mode, there are this time two possible outcomes:

- If the creation of complex is too slow, there is not enough resource and the polymer disappears in the same way as the previous case. We have found in Chapter I a sufficient condition for this case (given by $\Phi_P \leq \lambda^-$). This case is shown in Figure 3.3a.
- On the other hand, as soon as Φ_P is large enough, the polymer length reaches a non-zero equilibrium similar to the case without protein. This case is shown in Figure 3.3b.



(A) $\Phi_P = 2$ (B) $\Phi_P = 100$

Figure 3.3 Evolution of the length of the polymer with $N = 1000$, $\lambda^+ = 0$, $\lambda^- = 2$, $\lambda_F^+ = 10$, $\Phi_F^+ = 0$ and $\Phi_F^- = 0$ for two values of Φ_P. Initially, there are a (unique) polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. On each graph, we plot a single realisation of the stochastic system for $N = 1000$ and the fluid limit given in Theorem 1.2.1 (i.e. N times $l_1(t)$).

2.3 - Effect of the Switching Mode

When the system switches modes, five behaviours can be observed.

- If Φ_P is small enough (i.e. satisfies condition (1.56)), an equilibrium is obtained where the length of the polymer at equilibrium is greater than if the system remains in the fast mode. This corresponds to the case presented in Chapter I. However, we can distinguish two behaviours.
  - If $\Phi_P > \lambda^-$. We have just seen in Section 2.2.2 that, in this case, if the system remains in the fast mode then the polymer length reaches a non-zero equilibrium. As condition (1.56) ensures that the length at equilibrium with mode switching is greater than the length at equilibrium if the system remains in fast mode, the length of polymer necessarily reaches a non-zero equilibrium. This behaviour is depicted in Figure 1.3.
  - If $\Phi_P \leq \lambda^-$. We have seen in Section 2.2.2 that if the system remains in the fast mode then the length of the polymer goes to 0. When the system can switch between the two modes, we can distinguish two cases.
    - If $\lambda^+ - \lambda^-$ is large enough, the length of the polymer with mode switching reaches a non-zero equilibrium. The effect of the switching mode described in Chapter I is then

more apparent since the length of the polymer at equilibrium is non-zero (and quite large) while the length at equilibrium is zero if the system remains in fast mode. This case is illustrated in Figure 3.4.

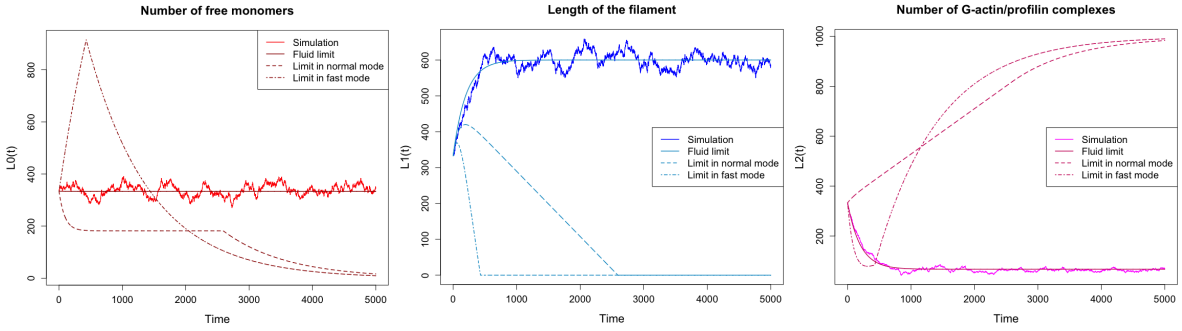
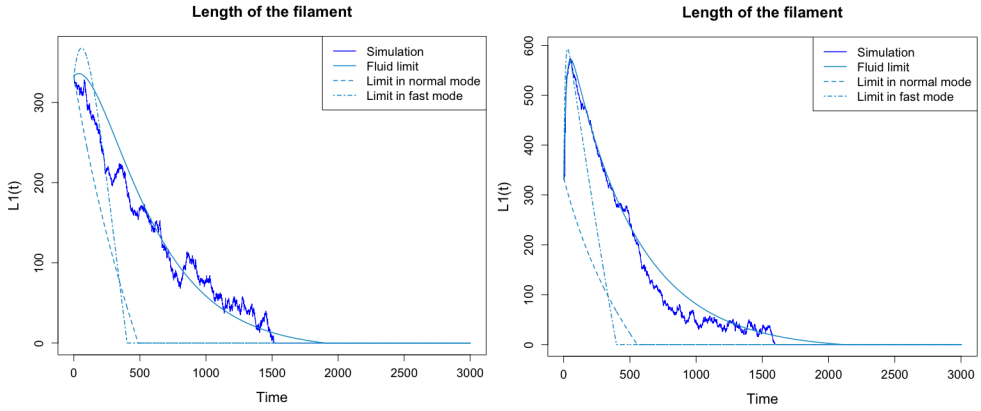


Figure 3.4 Evolution of the system described by the model for a single actin polymer with $N = 1000$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 10$, $\Phi_P = 1$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$. Initially, there are a (unique) polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. Left: evolution of the number of free monomers. Middle: evolution of the length of the polymer. Right: evolution of the number of G-actin/profilin complexes. On each graph, we plot a single realisation of the stochastic system for $N = 1000$, the fluid limit given in Theorem 1.2.1 (i.e. N times $l_i(t)$), the fluid limit one would obtain by assuming that the system stays in normal mode and the fluid limit one would obtain by assuming that the system remains in fast mode all the time.

⋆ If λ^+ is small, we can observe in Figure 3.5 that the mode switching does not allow to compensate the depolymerisation and the polymer length goes to 0. By increasing λ_F^+ significantly, Figure 3.5b shows that this behaviour occurs for a slightly lower Φ_P. Therefore, we can conclude that λ_F^+ has a small impact in this case.



(A) $\lambda_F^+ = 10$, $\Phi_P = 0.8$ (B) $\lambda_F^+ = 100$, $\Phi_P = 0.5$

Figure 3.5 Evolution of the length of the polymer with $N = 1000$, $\lambda^+ = 3$, $\lambda^- = 2$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$ for two values of (λ_F^+, Φ_P). Initially, there are a (unique) polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. On each graph, we plot a single realisation of the stochastic system for $N = 1000$, the fluid limit given in Theorem 1.2.1 (i.e. N times $l_i(t)$), the fluid limit one would obtain by assuming that the system stays in normal mode and the fluid limit one would obtain by assuming that the system remains in fast mode all the time.

⁕ If Φ_P is large, the profilin sequesters free monomers and we can observe two behaviours.

- If $\lambda_F^+ - \lambda^-$ is large enough, elongation induced by formins is fast enough to use G-actin/profilin complexes produced and an intermediate state is obtained. Indeed, the equilibrium length with mode switching is between the equilibrium length when the system stays in fast mode and the equilibrium length when the system stays in normal mode (which is always zero). This case is illustrated in Figure 3.6.

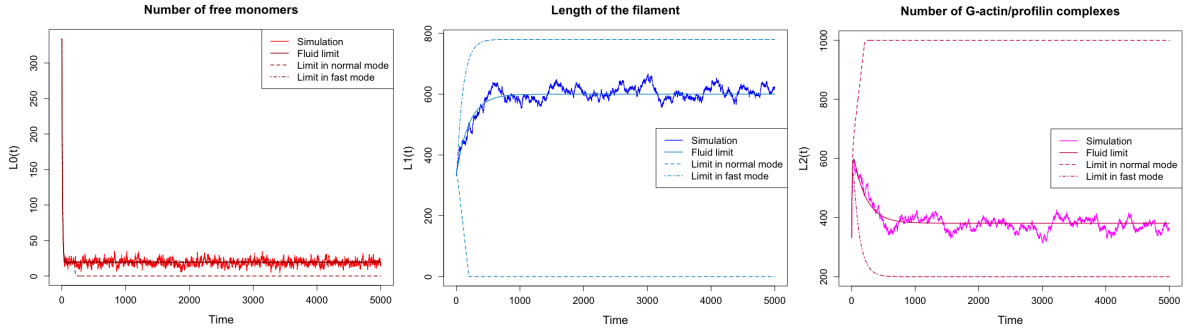


Figure 3.6 Evolution of the system described by the model for a single actin polymer with $N = 1000$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 10$, $\Phi_P = 100$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$. Initially, there are a (unique) polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. Left: evolution of the number of free monomers. Middle: evolution of the length of the polymer. Right: evolution of the number of G-actin/profilin complexes. On each graph, we plot a single realisation of the stochastic system for $N = 1000$, the fluid limit given in Theorem 1.2.1 (i.e. N times $l_i(t)$), the fluid limit one would obtain by assuming that the system stays in normal mode and the fluid limit one would obtain by assuming that the system remains in fast mode all the time.

- If λ_F^+ is small, G-actin/profilin complexes are not used fast enough and profilin sequesters too many free monomers, therefore the polymer length at equilibrium is zero. This behaviour is depicted in Figure 3.7. As when Φ_P is large there are very few free monomers, increasing spontaneous elongation (i.e. λ^+) does not allow a better use of the different pools of monomers.

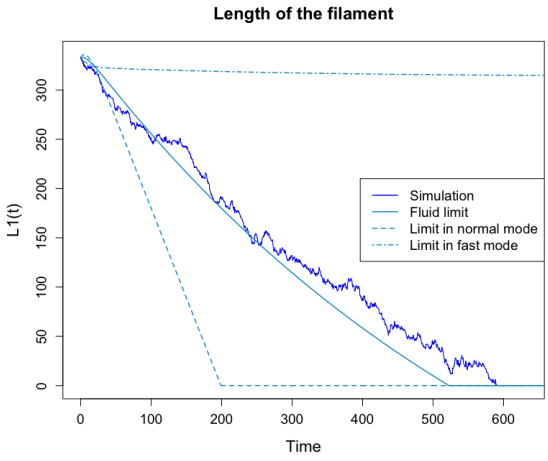
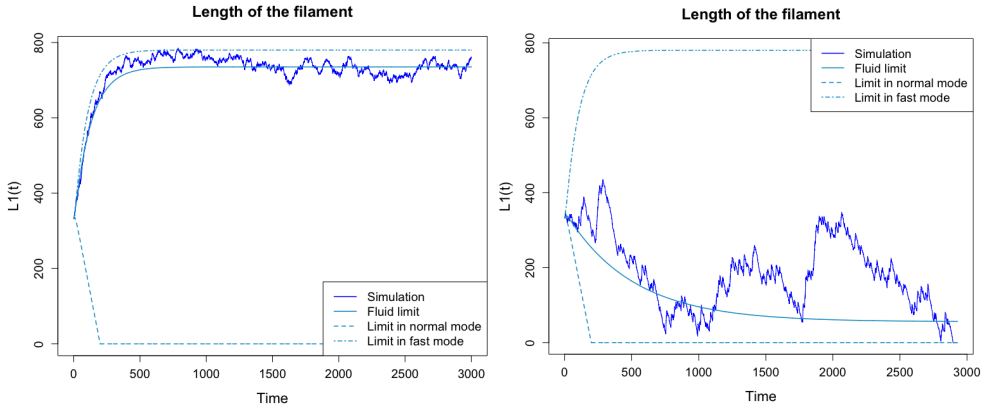


Figure 3.7 Evolution of the length of the polymer with $N = 1000$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 3$, $\Phi_P = 100$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$. Initially, there are a (unique) polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. On this graph, we plot a single realisation of the stochastic system for $N = 1000$, the fluid limit given in Theorem 1.2.1 (i.e. N times $l_i(t)$), the fluid limit one would obtain by assuming that the system stays in normal mode and the fluid limit one would obtain by assuming that the system remains in fast mode all the time.

2.4 - Conclusion

This section has allowed us to determine the different effects of mode switching and to identify the role of each parameter. In particular, it can be noted that in order to obtain a non-zero polymer length at equilibrium, the elongation rates λ^+ and λ_F^+ must be sufficiently large compared to the depolymerisation rate λ^-. The rate of creation of complexes Φ_P determines whether the mode switching will lead to a better use of the different pools of monomers. This parameter can therefore be used to adjust the amount of actin polymerised at equilibrium. Finally, parameters Φ_F^+ and Φ_F^- influence the behaviour in an intuitive way. As shown in Figure 3.8, when $\Phi_F^+ > \Phi_F^-$ (resp. $\Phi_F^+ < \Phi_F^-$), the behaviour with mode switching is more similar to the behaviour when the system remains in the fast (resp. normal) mode. It can also be noted that the slower the formin exchange, the more stochastic the system is.



(A) $\Phi_F^+ = 0.4$, $\Phi_F^- = 0.1$ (B) $\Phi_F^+ = 0.1$, $\Phi_F^- = 0.4$

Figure 3.8 Evolution of the length of the polymer with $N = 1000$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 10$, $\Phi_P = 100$ for two values of (Φ_F^+,Φ_F^-). Initially, there are a (unique) polymer of length 334, 333 free monomers and 333 G-actin/profilin complexes. On each graph, we plot a single realisation of the stochastic system for $N = 1000$, the fluid limit given in Theorem 1.2.1 (i.e. N times $l_i(t)$), the fluid limit one would obtain by assuming that the system stays in normal mode and the fluid limit one would obtain by assuming that the system remains in fast mode all the time.

3 - Model for a Population of Linear Actin Polymers

In this section, we aim to understand the impact of the competition between polymers for monomers on actin dynamics. For this purpose, we have taken up and compared all the cases from the previous section (Section 2). However, we will only present the differences with the case of a single polymer. We can already note that when there is no accessory protein, the large population limit has the same behaviour and the same limit as the fluid limit because the competition between polymers impacts elongation and depolymerisation in the same way.

3.1 - Effect of Profilin

As in the previous section, two cases can be distinguished:

1. All polymers in the population are simple (i.e. without formin), which corresponds for the pre-

vious model to remain in the normal mode.

2. All polymers in the population are associated with a formin, which corresponds for the previous model to remain in the fast mode.

When all the polymers are simple, we find the same behaviour as before: all the polymers in the population finally disappear because the profilin sequesters the monomers.

When all polymers are associated to a formin, there are always two possible issues, but the case analogous to that presented in Figure 3.3a seems, in the case of a population, biologically not very relevant. Indeed, to ensure that the amount of polymerised actin tends to 0, Φ_P must be very small. The sufficient condition $\Phi_P \leq \lambda^-$ is written in the case of a population: $\Phi_P \leq \lambda^- f_\infty$ where f_∞ represents the limiting amount of "real" polymers (defined in Chapter I page 66). As f_∞ is very small, this condition is only respected for Φ_P which is very small compared to λ^-, and this does not correspond to biological values (see Table 3.2).

More intuitively, competition between polymers for monomers (represented here by f_∞) primarily impacts elongation. Competition for monomers makes elongation difficult and thus leads to the disappearance of polymers. There is therefore a reduction in the number of ends that can depolymerise. Depolymerisation is therefore also affected by this competition. As for profilin, it is not slowed down by the competition since it only interacts with the free monomers, so we are still in a case similar to the one presented in Figure 3.9.

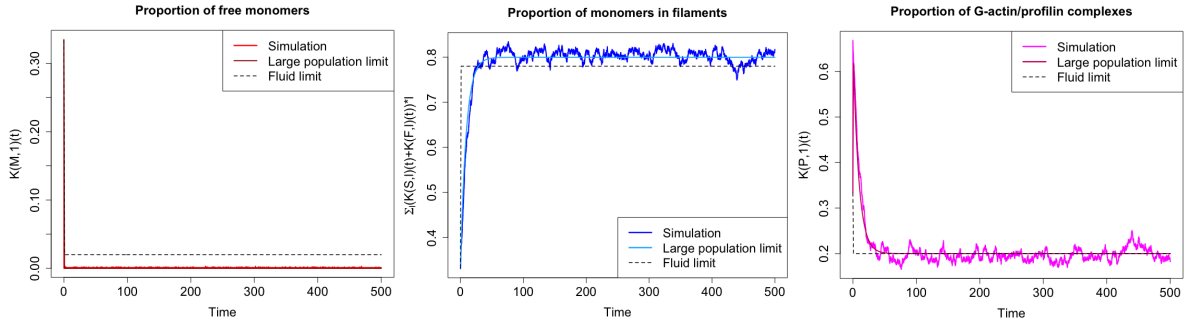


Figure 3.9 Evolution of the system described by the limiting model for a population of actin polymers with $dt = 0.001$, $\lambda^+ = 0$, $\lambda^- = 2$, $\lambda_F^+ = 10$, $\Phi_P = 100$, $\Phi_F^+ = 0$ and $\Phi_F^- = 0$, starting with 33, 4% of free monomers, 33, 3% of G-actin/profilin complexes and 33, 3% of monomers included within polymers (associated with a formin) of length 30. Left: evolution of the proportion of free monomers. Middle: evolution of the proportion of monomers included in a polymer. Right: evolution of the proportion of G-actin/profilin complexes. On each graph, we plot a single rescaled realisation of the stochastic system for $N = 1000$, the large population limit given in Theorem 1.2.2 and the fluid limit given in Theorem 1.2.1.

Figure 3.9 therefore presents the only relevant case when all polymers are associated with a formin. This case is analogous to the one presented in Figure 3.3b. It would therefore seem that, in a population of actin polymers associated to a formin, the amount of polymerised actin always tends to a non-zero equilibrium. Moreover, we can also note that the amount of actin polymerised at equilibrium in the fast mode is greater in the population than for a single polymer.

This can be easily proven by using results from Chapter I. By fitting equation (1.7) to the case where the system remains in the fast mode, we know that the length of the polymer at equilibrium is:

$$\ell^\infty_{single,\, fast} = 1 - \frac{\lambda^-}{\Phi_P} - \frac{\lambda^-}{\lambda_F^+}.$$

Using the assumption $\overline{F^{\infty}}(t) \underset{t\to+\infty}{\longrightarrow} f_{\infty}$ introduced on page 66 and system (1.58), we obtain the amount of actin polymerised at equilibrium given by:

$$\ell^{\infty}_{pop,\,fast} = 1 - \frac{\lambda^{-} f_{\infty}}{\Phi_P} - \frac{\lambda^{-}}{\lambda^{+}_{F}}.$$

We therefore want to show that:

$$\ell^{\infty}_{single,\,fast} \leq \ell^{\infty}_{pop,\,fast} \iff 1 - \frac{\lambda^{-}}{\Phi_P} - \frac{\lambda^{-}}{\lambda^{+}_{F}} \leq 1 - \frac{\lambda^{-} f_{\infty}}{\Phi_P} - \frac{\lambda^{-}}{\lambda^{+}_{F}},$$

which is obvious since $f_{\infty} \in [0, 1]$.

3.2 - Effect of the Switching Mode

As explain in Chapter I, the condition (1.59) ensures that for small values of Φ_P, we can find cases where the mode switching improves the use of the different pools of monomers. However, such small values for Φ_P seem to have limited biological relevance. We therefore consider that, in the case of a population of linear polymers, there is always more actin polymerised if all the polymers are associated with a formin and so switching modes does not improve the use of the different pools of monomers.

We consider the five behaviours depicted in Section 2.3.

- Figure 1.5 showing the analogous case to Figure 1.3 is discussed in Chapter I. We can add that in this case, the equilibrium of the fluid limit is significantly lower than the equilibrium of the large population limit. This result can be proven in a similar way as in the previous section. Using equation (1.7), we know that the length of the polymer at equilibrium is:

$$\ell^{\infty}_{single} = 1 - \frac{\lambda^{-}}{\pi_M(0)\lambda^{+} + \Phi_P} - \frac{\lambda^{-}\Phi_P}{(\pi_M(0)\lambda^{+} + \Phi_P)\pi_M(1)\lambda^{+}_{F}}.$$

  Using once again the assumption $\overline{F^{\infty}}(t) \underset{t\to+\infty}{\longrightarrow} f_{\infty}$ and system (1.58), we obtain the amount of actin polymerised at equilibrium, given by:

$$\ell^{\infty}_{pop} = 1 - \frac{\lambda^{-} f_{\infty}}{\pi_M(0)\lambda^{+} f_{\infty} + \Phi_P} - \frac{\lambda^{-} f_{\infty}\Phi_P}{(\pi_M(0)\lambda^{+} f_{\infty} + \Phi_P)\pi_M(1)\lambda^{+}_{F} f_{\infty}}.$$

  We have:

$$\begin{aligned}
&\ell^{\infty}_{single} \leq \ell^{\infty}_{pop} \\
&\iff \frac{\lambda^{-}}{\pi_M(0)\lambda^{+} + \Phi_P} + \frac{\lambda^{-}\Phi_P}{(\pi_M(0)\lambda^{+} + \Phi_P)\pi_M(1)\lambda^{+}_{F}} \geq \frac{\lambda^{-} f_{\infty}}{\pi_M(0)\lambda^{+} f_{\infty} + \Phi_P} + \frac{\lambda^{-} f_{\infty}\Phi_P}{(\pi_M(0)\lambda^{+} f_{\infty} + \Phi_P)\pi_M(1)\lambda^{+}_{F} f_{\infty}} \\
&\iff \frac{\pi_M(1)\lambda^{+}_{F} + \Phi_P}{\pi_M(0)\lambda^{+} + \Phi_P} \geq \frac{\pi_M(1)\lambda^{+}_{F} f_{\infty} + \Phi_P}{\pi_M(0)\lambda^{+} f_{\infty} + \Phi_P} \\
&\iff \pi_M(1)\lambda^{+}_{F} + \pi_M(0)\lambda^{+} f_{\infty} \geq \pi_M(0)\lambda^{+} + \pi_M(1)\lambda^{+}_{F} f_{\infty} \\
&\iff \pi_M(1)\lambda^{+}_{F} - \pi_M(0)\lambda^{+} \geq (\pi_M(1)\lambda^{+}_{F} - \pi_M(0)\lambda^{+}) f_{\infty}
\end{aligned}$$

  which is obvious since $f_{\infty} \in [0, 1]$.

- As explained earlier, the cases presented in Figure 3.4 and Figure 3.5 are considered irrelevant for a population of polymers.

* Figure 3.10 presents the analogous case to Figure 3.6. We notice this time that if Φ_P and λ_F^+ are sufficiently large, the equilibrium of the fluid limit and the equilibrium of the large population limit are close. Competition between polymers therefore seems to have a minor impact on the "intermediate" cases. This can be explained on the one hand by the fact that when Φ_P is large, there are always enough complexes available, so the amount of actin polymerised is limited by the rate of polymerisation with formin. On the other hand, only a part of the polymer population is associated with a formin, so there are fewer polymers competing for a large amount of complexes.

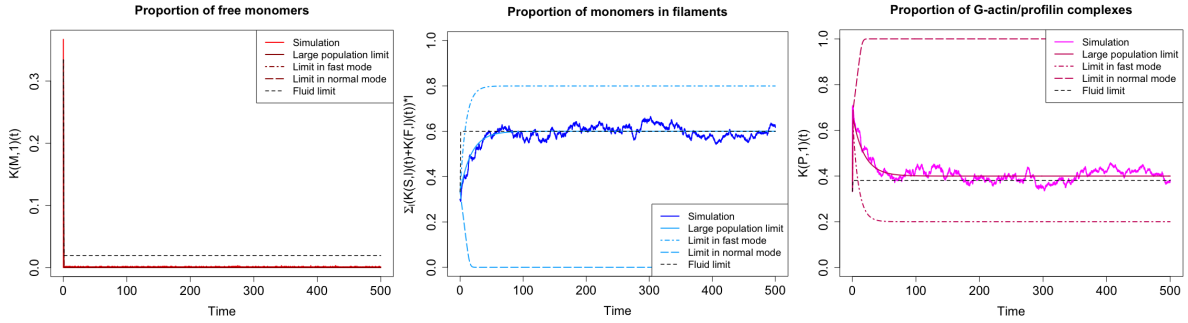


Figure 3.10 Evolution of the system described by the limiting model for a population of actin polymers with $dt = 0.001$, $\lambda^+ = 10$, $\lambda^- = 2$, $\lambda_F^+ = 10$, $\Phi_P = 100$, $\Phi_F^+ = 1$ and $\Phi_F^- = 1$, starting with 33, 4% of free monomers, 33, 3% of G-actin/profilin complexes and 33, 3% of monomers included within polymers of length 30. Left: evolution of the proportion of free monomers. Middle: evolution of the proportion of monomers included in a polymer. Right: evolution of the proportion of G-actin/profilin complexes. On each graph, we plot a single rescaled realisation of the stochastic system for $N = 1000$, the large population limit given in Theorem 1.2.2, the large population limit one would obtain by assuming that the system remains in normal mode, the large population limit one would obtain by assuming that the system remains in fast mode and the fluid limit given in Theorem 1.2.1.

* If $\lambda_F^+ - \lambda^- > 0$ is small enough, we find the same behaviour as in Figure 3.7.

3.3 - Conclusion

From this section we can conclude that competition between polymers for monomers affects polymerisation and depolymerisation in the same way but does not affect the creation of complexes. Complex creation is therefore always faster than other events and there are always enough complexes available not to slow down the polymerisation with a formin. Consequently, in a population of actin polymers, the more formins there are, the more actin is polymerised. In a population, Φ_P can no longer limit the amount of actin polymerised, and this regulation is therefore ensured by the difference in speed between elongation and depolymerisation and/or by another accessory protein (causing fragmentations for example).

4 - Model Considering all Accessory Proteins

In this section, we present a model for a population of actin polymers which takes into account spontaneous elongation and depolymerisation, nucleation, formins and profilin, capping proteins, fragmentation and branched polymers. To obtain this new model, we added nucleation, capping, fragmentation and branching to the model of Chapter I. After introducing the model, we relate our parameters to the

measured biological quantities. Using these values, we present a random realisation in order to show some possibilities offered by the implementation. We finish by introducing some perspectives.

4.1 - Biological Phenomena Considered and Assumptions

Profilin and Formins - Fast elongation. Profilin is a protein that binds to a free actin monomer to create G-actin/profilin complexes. These complexes can not be use to elongate simple polymers but are the resource for the elongation with a formin. A profilin does not release the G-actin until this monomer is polymerised.

A formin can be attached to the plus end of a polymer. Formins can accelerate elongation by working conjointly with profilin. Indeed, when a polymer elongates with a formin, the elongation does not consume a free actin monomer but a G-actin/profilin complex. A polymer associated with a formin can not be branched and since the plus end is bound, a polymer associated with a formin can not be capped. The formin associated with a polymer can be released. In this case, the polymer returns to being a simple polymer.

Arp2/3 complexes - Branching. The Arp2/3 protein complex can bind to each actin monomer of an actin polymer (as long as there is no Arp2/3 on that monomer, we do not consider steric effects or geometric restrictions for branching). After this binding, a secondary actin polymer (called secondary branch) grows on the polymer (called primary branch) and the polymer is branched. The presence of Arp2/3 on the polymer has no impact on the elongation-depolymerisation dynamics of the primary branch of the polymer. To simplify the model, we assume that the elongation rate is the same for the primary and secondary branches. Furthermore, each secondary branch can be capped but can not depolymerise. An Arp2/3 can be released in two ways. Firstly, an Arp2/3 protein can release the polymer spontaneously. Secondly, a monomer of the primary branch associated with an Arp2/3 can be depolymerised. In both cases, a split occurs and the branched polymer is split into two. A branched polymer can not be bound with a formin.

Cofilin - Fragmentation. As Arp2/3, cofilin can bind to each actin monomer of an actin polymer (as long as that monomer is not already bound to a protein). Cofilin binds mainly to ADP monomers of the polymer. When a sufficient amount of cofilin is bound to the polymer, the mechanical properties of the ADP part and of the ATP part are different and fragmentation takes place at the boundary of these parts. The ADP part is then completely depolymerised while the ATP part remains and can elongate again. We will assume that if fragmentation occurs for a simple filament or a filament associated with a formin, a random proportion of the length remains; if the filament is capped, the majority of its monomers are ADP (as it can no longer elongate) and it is assumed that the polymer will depolymerise entirely when fragmentation occurs. Finally, if a branched filament fragments, it loses all of its capped branches and only a randomly selected portion remains. As mentioned in [BBPSP14] and Figure 11, if there is a very large amount of cofilin on the filament, it becomes flexible and fragmentation does not occur. We will try to take this phenomenon into account when choosing the fragmentation rate.

Capping proteins - Capping. A capping protein can bind the plus end of a polymer. After capping, the polymer can no longer elongate. A capping protein cannot detach from a polymer. Since the plus end is already bound, a formin can not bind to a capped filament.

Nucleation. Nucleation is the process by which a new polymer is created. Nucleation can be

initiated by several proteins but in order to simplify this process, we consider that nucleation is the meeting of three actin monomers to create a single polymer of length three. We also consider that polymers of length two do not exist since a polymer must have at least three monomers to be stable.

4.2 - The Model

4.2.1 - State space and variables of interest

In order to describe the model, the same approach as for the population model in Chapter I is used. We assume that the total number of monomers on the system is fixed, equal to $N \in \mathbb{N}$. Since we are now considering branched polymers and capping proteins, we describe every element of the population by its type, its number of branching, its length and its number of free extremities (i.e. without capping protein at the plus end). More precisely, free monomers are denoted by M, G-actin/profilin complexes by P, simple actin polymers by S, actin polymers bound with a formin by F, branched actin polymers by B and linear (i.e. not branched) capped actin polymer by C. As in Chapter I, we again consider that free monomers and G-actin/profilin complexes are two types of actin polymers. Therefore, polymers of type M and P can only be of length 1 and have 0 branching and 0 free extremity (since elongation is impossible from a single monomers). Other types of polymers S, F, C and B can have any length in $\{3, 4, \cdots, N\}$. Note that the "length" of a branched polymer corresponds to the number of monomers that compose this polymer. As polymers of type S, F and C are linear, they have 0 branching and 1 free extremity for types S and F, 0 free extremity for type C. Polymers of type B of length $\ell \in \{3, 4, \cdots, N\}$ can have any number of branching $b \in \{1, 2, \cdots, \ell\}$ and any number of free extremities $e \in \{0, 1, \cdots, b+1\}$. We will therefore work in the state space ${}^\star\boldsymbol{\mathcal{I}}$ defined by:

$$
\begin{aligned}
{}^\star\boldsymbol{\mathcal{I}} \overset{def}{=} \{(M,0,1,0),(P,0,1,0)\} \cup (\{(S,0),(F,0)\} \times \mathbb{N}_{\geq 3} \times \{1\}) \cup (\{(C,0)\} \times \mathbb{N}_{\geq 3} \times \{0\}) \\
\cup (\{B\} \times \mathbb{N}^* \times \mathbb{N}_{\geq 3} \times \mathbb{N}),
\end{aligned} \tag{3.1}
$$

and for every quadruplet $(T, b, \ell, e) \in \mathcal{S}$ and every $t \in \mathbb{R}_+$, we define ${}^\star K_t^N(T, b, \ell, e)$ as the number of polymers of type T, length ℓ, with b branching and e free extremities present in the system at time t. Here we use notation: $\mathbb{N}^* = \{1, 2, \cdots\}$, $\mathbb{N} = \mathbb{N}^* \cup \{0\}$ and $\mathbb{N}_{\geq 3} = \{3, 4, \cdots\}$. The system at every time t is described by its empirical distribution ${}^\star\mathcal{Z}_t^N$ defined by:

$$
\begin{aligned}
{}^\star\mathcal{Z}_t^N \overset{def}{=} {}^\star K_t^N(M,0,1,0)\delta_{(M,0,1,0)} + {}^\star K_t^N(P,0,1,0)\delta_{(P,0,1,0)} \\
+ \sum_{l=3}^{N} {}^\star K_t^N(S,0,l,1)\delta_{(S,0,l,1)} + \sum_{l=3}^{N} {}^\star K_t^N(F,0,l,1)\delta_{(F,0,l,1)} + \sum_{l=3}^{N} {}^\star K_t^N(C,0,l,0)\delta_{(C,0,l,0)} \\
+ \sum_{l=3}^{N} \sum_{b=1}^{l} \sum_{e=0}^{b+1} {}^\star K_t^N(B,b,l,e)\delta_{(B,b,l,e)},
\end{aligned} \tag{3.2}
$$

where δ_x denotes a Dirac mass at x. Each ${}^\star\mathcal{Z}_t^N$ is thus a random measure on ${}^\star\boldsymbol{\mathcal{I}}$. We write $\mathcal{M}_{\cdot\boldsymbol{\mathcal{I}}}$ for the space of all finite measures on ${}^\star\boldsymbol{\mathcal{I}}$, endowed with the topology of weak convergence.

Let us now introduce some variables of interest. First, let ${}^\star p_l$ denote the projector $(T, b, l, e) \mapsto l$ that returns the third coordinate of an element in ${}^\star\boldsymbol{\mathcal{I}}$. Since, the total number of monomers is N, for every $t \geq 0$, we have ${}^\star K_t^N(\cdot, \cdot, l, \cdot) = 0$ if $l > N$ and

$$
\left\langle {}^\star\mathcal{Z}_t^N, {}^\star p_l \right\rangle = N. \tag{3.3}
$$

The total number of "real" polymers at time t (i.e. that are not free monomers or complexes) is denoted by ${}^\star P^N(t)$ with:

$$
{}^\star P^N(t) \overset{def}{=} \left\langle {}^\star\mathcal{Z}_t^N, \mathbb{1}_{(S,0,\cdot,1)} + \mathbb{1}_{(F,0,\cdot,1)} + \mathbb{1}_{(C,0,\cdot,0)} + \mathbb{1}_{(B,\cdot,\cdot,\cdot)} \right\rangle, \tag{3.4}
$$

and the amount of polymerised actin at time t (i.e. total number of monomers inserted in a polymer) is denoted by ${}^\star M^N_{inP}(t)$ with:

$$ {}^\star M^N_{inP}(t) \overset{def}{=} \left\langle {}^\star\mathcal{Z}^N_t, {}^\star p_l \left(\mathbb{1}_{(S,0,\cdot,1)} + \mathbb{1}_{(F,0,\cdot,1)} + \mathbb{1}_{(C,0,\cdot,0)} + \mathbb{1}_{(B,\cdot,\cdot,\cdot)} \right) \right\rangle . \tag{3.5} $$

The total number of simple polymers (respectively polymers bound with a formin, linear capped polymers, branched polymers) at time t is denoted by ${}^\star S^N(t)$ (resp. ${}^\star F^N(t)$, ${}^\star C^N(t)$, ${}^\star B^N(t)$) with:

$$ {}^\star S^N(t) \overset{def}{=} \left\langle {}^\star\mathcal{Z}^N_t, \mathbb{1}_{(S,0,\cdot,1)} \right\rangle = \sum_{\ell=3}^{N} {}^\star K^N_t(S,0,\ell,1), \tag{3.6} $$

$$ {}^\star F^N(t) \overset{def}{=} \left\langle {}^\star\mathcal{Z}^N_t, \mathbb{1}_{(F,0,\cdot,1)} \right\rangle = \sum_{\ell=3}^{N} {}^\star K^N_t(F,0,\ell,1), \tag{3.7} $$

$$ {}^\star C^N(t) \overset{def}{=} \left\langle {}^\star\mathcal{Z}^N_t, \mathbb{1}_{(C,0,\cdot,0)} \right\rangle = \sum_{\ell=3}^{N} {}^\star K^N_t(C,0,\ell,0), \tag{3.8} $$

$$ {}^\star B^N(t) \overset{def}{=} \left\langle {}^\star\mathcal{Z}^N_t, \mathbb{1}_{(B,\cdot,\cdot,\cdot)} \right\rangle = \sum_{\ell=3}^{N} \sum_{b=1}^{\ell} \sum_{e=0}^{b+1} {}^\star K^N_t(B,b,\ell,e). \tag{3.9} $$

The total number of actin monomers polymerised within simple polymers (respectively polymers bound with a formin, linear capped polymers, branched polymers) at time t is denoted by ${}^\star M^N_{inS}(t)$ (resp. ${}^\star M^N_{inF}(t)$, ${}^\star M^N_{inC}(t)$, ${}^\star M^N_{inB}(t)$) with:

$$ {}^\star M^N_{inS}(t) \overset{def}{=} \left\langle {}^\star\mathcal{Z}^N_t, {}^\star p_l \left(\mathbb{1}_{(S,0,\cdot,1)} \right) \right\rangle = \sum_{\ell=3}^{N} {}^\star K^N_t(S,0,\ell,1) \times \ell, \tag{3.10} $$

$$ {}^\star M^N_{inF}(t) \overset{def}{=} \left\langle {}^\star\mathcal{Z}^N_t, {}^\star p_l \left(\mathbb{1}_{(F,0,\cdot,1)} \right) \right\rangle = \sum_{\ell=3}^{N} {}^\star K^N_t(F,0,\ell,1) \times \ell, \tag{3.11} $$

$$ {}^\star M^N_{inC}(t) \overset{def}{=} \left\langle {}^\star\mathcal{Z}^N_t, {}^\star p_l \left(\mathbb{1}_{(C,0,\cdot,0)} \right) \right\rangle = \sum_{\ell=3}^{N} {}^\star K^N_t(C,0,\ell,0) \times \ell, \tag{3.12} $$

$$ {}^\star M^N_{inB}(t) \overset{def}{=} \left\langle {}^\star\mathcal{Z}^N_t, {}^\star p_l \left(\mathbb{1}_{(B,\cdot,\cdot,\cdot)} \right) \right\rangle = \sum_{\ell=3}^{N} \sum_{b=1}^{\ell} \sum_{e=0}^{b+1} {}^\star K^N_t(B,b,\ell,e) \times \ell. \tag{3.13} $$

4.2.2 - Description of all possible events

The process $\left({}^\star\mathcal{Z}^N_t\right)_{t\in\mathbb{R}_+}$ evolves according to the following transitions.

❄ **Elongation.**

- *Simple polymers.* Each simple polymer can elongate by catching a free monomer at rate λ^+, independently of the other polymers. For each length $\ell \in \{3,4,\cdots,N-1\}$, the rate at which spontaneous elongation happens on simple polymers of length ℓ at time t is $\lambda^+ \frac{{}^\star K^N_t(M,0,1,0)}{N} {}^\star K^N_t(S,0,\ell,1)$ and the new state of the system is:

$$ {}^\star\mathcal{Z}^N_t = {}^\star\mathcal{Z}^N_{t-} - \delta_{(S,0,\ell,1)} - \delta_{(M,0,1,0)} + \delta_{(S,0,\ell+1,1)}, $$

or equivalently:

$$\begin{cases} {}^\star K_t^N(S,0,\ell,1) = {}^\star K_{t^-}^N(S,0,\ell,1) - 1, \\ {}^\star K_t^N(S,0,\ell+1,1) = {}^\star K_{t^-}^N(S,0,\ell+1,1) + 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t^-}^N(M,0,1,0) - 1. \end{cases}$$

- *Polymers associated with a formin.* Each polymer associated with a formin can elongate by absorbing a G-actin/profilin complex at rate λ_F^+ independently of the other polymer. For each length $\ell \in \{3,4,\cdots,N-1\}$, the rate at which elongation with a formin happens on polymers of length ℓ at time t is $\lambda_F^+ \frac{{}^\star K_t^N(P,0,1,0)}{N} {}^\star K_t^N(F,0,\ell,1)$ and the new state of the system is:

$${}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t^-}^N - \delta_{(F,0,\ell,1)} - \delta_{(P,0,1,0)} + \delta_{(F,0,\ell+1,1)},$$

or equivalently:

$$\begin{cases} {}^\star K_t^N(F,0,\ell,1) = {}^\star K_{t^-}^N(F,0,\ell,1) - 1, \\ {}^\star K_t^N(F,0,\ell+1,1) = {}^\star K_{t^-}^N(F,0,\ell+1,1) + 1, \\ {}^\star K_t^N(P,0,1,0) = {}^\star K_{t^-}^N(P,0,1,0) - 1. \end{cases}$$

- *Branched polymers.* Each free extremity of a branched polymer can elongate by catching a free monomer at rate λ^+, independently of the other extremities and the other polymers. For each length $\ell \in \{3,4,\cdots,N-1\}$, for each number of branching $b \in \{1,2,\cdots,\ell\}$ and for each number of free extremities $e \in \{0,1,\cdots,b+1\}$, the rate at which elongation happens on polymers of type (B,b,ℓ,e) at time t is $\lambda^+ \frac{{}^\star K_t^N(M,0,1,0)}{N} {}^\star K_t^N(B,b,\ell,e)e$ and the new state of the system is:

$${}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t^-}^N - \delta_{(B,b,\ell,e)} - \delta_{(M,0,1,0)} + \delta_{(B,b,\ell+1,e)},$$

or equivalently:

$$\begin{cases} {}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t^-}^N(B,b,\ell,e) - 1, \\ {}^\star K_t^N(B,b,\ell+1,e) = {}^\star K_{t^-}^N(B,b,\ell+1,e) + 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t^-}^N(M,0,1,0) - 1. \end{cases}$$

⁕ **Depolymerisation.**

- *Linear polymers.* Each linear polymer (i.e. simple polymer, polymer associated to a formin and linear capped polymer) can release a monomer at rate λ^- and this monomer becomes "free". When a polymer of length 3 depolymerises, it depolymerises completely and releases 3 monomers (this is because actin dimers are unstable). For each type $T \in \{S,F,C\}$, for each length $\ell \in \{3,4,\cdots,N\}$ and for each number of extremity $e \in \{0,1\}$, the rate a which depolymerisation happens on polymers of type $(T,0,\ell,e)$ at time t is $\lambda^- {}^\star K_t^N(T,0,\ell,e)$ and the new state of the system is:

$${}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t^-}^N - \delta_{(T,0,3,e)} + 3\delta_{(M,0,1,0)} \text{ if } \ell = 3,$$

$${}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t^-}^N - \delta_{(T,0,\ell,e)} + \delta_{(M,0,1,0)} + \delta_{(T,0,\ell-1,e)} \text{ if } \ell > 3,$$

or equivalently:

$$\begin{cases} {}^\star K_t^N(T,0,3,e) = {}^\star K_{t^-}^N(T,0,3,e) - 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t^-}^N(M,0,1,0) + 3, \end{cases} \quad \text{if } \ell = 3,$$

$$\begin{cases} {}^\star K_t^N(T,0,\ell,e) = {}^\star K_{t^-}^N(T,0,\ell,e) - 1, \\ {}^\star K_t^N(T,0,\ell-1,e) = {}^\star K_{t^-}^N(T,0,\ell-1,e) + 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t^-}^N(M,0,1,0) + 1. \end{cases} \quad \text{if } \ell > 3,$$

- *Branched polymers.* Each branched polymer can release a monomer at rate λ^-, we recall that only the primary branch can depolymerise. In the same way as in the previous case, if the polymer is of length 3 and is depolymerised, then the polymer disappears and 3 monomers are released. When a branched polymer depolymerises, the detaching monomer may be bound to an Arp2/3 complex, in which case a split occurs.

  For each length $\ell \in \{3, 4, \cdots, N\}$, for each number of branching $b \in \{1, 2, \cdots, \ell\}$ and for each number of free extremities $e \in \{0, 1, \cdots, b+1\}$, the rate at which depolymerisation happens on polymers of type (B, b, ℓ, e) at time t is $\lambda^- {}^\star K_t^N(B, b, \ell, e)$. To determine whether a split occurs, we take a Bernoulli random variable $\mathfrak{s}$ such that $\mathfrak{s} \sim \text{Ber}\left(\frac{b}{\ell}\right)$; if $\mathfrak{s} = 0$ the depolymerisation is simple (as in the previous case) and if $\mathfrak{s} = 1$ a split occurs. This means that there is a high probability of splitting if $b \approx \ell$, in other words, when the polymer is very branched with very short branches. On the contrary, if the polymer is weakly branched with long branches ($b \ll \ell$), splits will be rare.

  When a split occurs, two new polymers are created from the (B, b, ℓ, e) type polymer. As a whole, one monomer of the polymer is released, one branching is lost and the number of free extremities is unchanged. As the topology of the (B, b, ℓ, e) type polymer is not known, the two new polymers will be constructed randomly according to the following procedure:

  - We randomly choose the length of the first polymer ℓ_1 according to a uniform distribution: $\ell_1 \sim \text{Unif}(\{0, 1, \cdots, \ell - 1\})$.
  - As depolymerisation leads to the release of a monomer, we have: $\ell_1 + \ell_2 = \ell - 1$ and we set the length of the second polymer ℓ_2 by: $\ell_2 = \ell - 1 - \ell_1$.
  - We randomly choose the number of branching of the first polymer b_1 according to a uniform distribution: $b_1 \sim \text{Unif}(\{\max(0, b - 1 - \ell_2), \cdots, \min(b - 1, \ell_1)\})$ which ensures $b_1 \leq \ell_1$ and $b_2 \leq \ell_2$.
  - As split results in the loss of a branching, we have: $b_1 + b_2 = b - 1$ and we set the number of branching of the second polymer b_2 by: $b_2 = b - 1 - b_1$.
  - We randomly choose the number of free extremities of the first polymer e_1 according to a uniform distribution: $e_1 \sim \text{Unif}(\{\max(0, e - b_2 - 1), \cdots, \min(e, b_1 + 1)\})$ which ensures $e_1 \leq b_1 + 1$ and $e_2 \leq b_2 + 1$.
  - We finally set the number of free extremities of the second polymer e_2 by: $e_2 = e - e_1$.

  The new state of the system is then:

$$
\begin{aligned}
{}^\star\mathcal{Z}_t^N &= {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,3,e)} + 3\delta_{(M,0,1,0)} \text{ if } \ell = 3,\\
{}^\star\mathcal{Z}_t^N &= {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,\ell,e)} + \delta_{(M,0,1,0)} + \delta_{(B,b,\ell-1,e)} \text{ if } \ell > 3 \text{ and } \mathfrak{s} = 0,\\
{}^\star\mathcal{Z}_t^N &= {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,\ell,e)} + \delta_{(M,0,1,0)} + \delta_{(B,b_1,\ell_1,e_1)} + \delta_{(B,b_2,\ell_2,e_2)} \text{ if } \ell > 3,\ \mathfrak{s} = 1,\ \ell_1 \geq 3 \text{ and } \ell_2 \geq 3,\\
{}^\star\mathcal{Z}_t^N &= {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,\ell,e)} + (\ell_2 + 1)\delta_{(M,0,1,0)} + \delta_{(B,b_1,\ell_1,e_1)} \text{ if } \ell > 3,\ \mathfrak{s} = 1,\ \ell_1 \geq 3 \text{ and } \ell_2 < 3,\\
{}^\star\mathcal{Z}_t^N &= {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,\ell,e)} + (\ell_1 + 1)\delta_{(M,0,1,0)} + \delta_{(B,b_2,\ell_2,e_2)} \text{ if } \ell > 3,\ \mathfrak{s} = 1,\ \ell_1 < 3 \text{ and } \ell_2 \geq 3,\\
{}^\star\mathcal{Z}_t^N &= {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,\ell,e)} + \ell\,\delta_{(M,0,1,0)} \text{ if } \ell > 3,\ \mathfrak{s} = 1,\ \ell_1 < 3 \text{ and } \ell_2 < 3,
\end{aligned}
$$

or equivalently:

$$\begin{cases} {}^\star K_t^N(B,b,3,e) = {}^\star K_{t-}^N(B,b,3,e) - 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) + 3, \end{cases} \quad \text{if } \ell = 3,$$

$$\begin{cases} {}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t-}^N(B,b,\ell,e) - 1, \\ {}^\star K_t^N(B,b,\ell-1,e) = {}^\star K_{t-}^N(B,b,\ell-1,e) + 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) + 1, \end{cases} \quad \text{if } \ell > 3 \text{ and } \mathfrak{s} = 0,$$

$$\begin{cases} {}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t-}^N(B,b,\ell,e) - 1, \\ {}^\star K_t^N(B,b_1,\ell_1,e_1) = {}^\star K_{t-}^N(B,b_1,\ell_1,e_1) + 1, \\ {}^\star K_t^N(B,b_2,\ell_2,e_2) = {}^\star K_{t-}^N(B,b_2,\ell_2,e_2) + 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) + 1, \end{cases} \quad \text{if } \ell > 3,\ \mathfrak{s} = 1,\ \ell_1 \geq 3 \text{ and } \ell_2 \geq 3,$$

$$\begin{cases} {}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t-}^N(B,b,\ell,e) - 1, \\ {}^\star K_t^N(B,b_1,\ell_1,e_1) = {}^\star K_{t-}^N(B,b_1,\ell_1,e_1) + 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) + 1 + \ell_2, \end{cases} \quad \text{if } \ell > 3,\ \mathfrak{s} = 1,\ \ell_1 \geq 3 \text{ and } \ell_2 < 3,$$

$$\begin{cases} {}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t-}^N(B,b,\ell,e) - 1, \\ {}^\star K_t^N(B,b_2,\ell_2,e_2) = {}^\star K_{t-}^N(B,b_2,\ell_2,e_2) + 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) + 1 + \ell_1, \end{cases} \quad \text{if } \ell > 3,\ \mathfrak{s} = 1,\ \ell_1 < 3 \text{ and } \ell_2 \geq 3,$$

$$\begin{cases} {}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t-}^N(B,b,\ell,e) - 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) + \ell, \end{cases} \quad \text{if } \ell > 3,\ \mathfrak{s} = 1,\ \ell_1 < 3 \text{ and } \ell_2 < 3.$$

⁕ **Production of G-actin/profilin complexes.** Each free monomer can bind with a profilin and create a G-actin/profilin complex at rate Φ_P, independently of the others. Therefore, the total rate of creation of complexes at time t is $\Phi_P {}^\star K_{t-}^N(M,0,1,0)$ and the new state of the system is:

$${}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t-}^N - \delta_{(M,0,1,0)} + \delta_{(P,0,1,0)},$$

or equivalently:

$$\begin{cases} {}^\star K_t^N(P,0,1,0) = {}^\star K_{t-}^N(P,0,1,0) + 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) - 1. \end{cases}$$

⁕ **Binding of a formin.** Each simple polymer can bind with a formin at rate Φ_F^+ and therefore change category independently of the other polymers. For each length $\ell \in \{3,4,\cdots,N\}$, the rate at which binding of a formin happens on simple polymers of length ℓ at time t is $\Phi_F^+ {}^\star K_t^N(S,0,\ell,1)$ and the new state of the system is:

$${}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t-}^N - \delta_{(S,0,\ell,1)} + \delta_{(F,0,\ell,1)},$$

or equivalently:

$$\begin{cases} {}^\star K_t^N(F,0,\ell,1) = {}^\star K_{t-}^N(F,0,\ell,1) + 1, \\ {}^\star K_t^N(S,0,\ell,1) = {}^\star K_{t-}^N(S,0,\ell,1) - 1. \end{cases}$$

⁕ **Release of a formin.** Each polymer associated with a formin can have this association broken at rate Φ_F^-, independently of the other polymers. For each length $\ell \in \{3,4,\cdots,N\}$, the rate at which release of a formin happens on polymers of length ℓ at time t is $\Phi_F^- {}^\star K_t^N(F,0,\ell,1)$ and the new state of the system is:

$${}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t-}^N - \delta_{(F,0,\ell,1)} + \delta_{(S,0,\ell,1)},$$

or equivalently:

$$\begin{cases} {}^\star K_t^N(S,0,\ell,1) = {}^\star K_{t^-}^N(S,0,\ell,1) + 1, \\ {}^\star K_t^N(F,0,\ell,1) = {}^\star K_{t^-}^N(F,0,\ell,1) - 1. \end{cases}$$

- **Binding of an Arp2/3 complex.**
  - *Simple and linear capped polymers.* An Arp2/3 complex can bind to each monomer of each single polymer or each capped linear polymer at rate Φ_A^+ and create a branched polymer. For each type $T \in \{S, C\}$, for each length $\ell \in \{3, 4, \cdots, N\}$ and for each number of extremities $e \in \{0, 1\}$, the rate at which binding of an Arp2/3 happens on polymers of type $(T, 0, \ell, e)$ at time t is $\Phi_A^+ {}^\star K_t^N(T, 0, \ell, e)\ell$ and the new state of the system is:

$$ {}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t^-}^N - \delta_{(T,0,\ell,e)} + \delta_{(B,1,\ell,e+1)},$$

    or equivalently:

$$\begin{cases} {}^\star K_t^N(B,1,\ell,e+1) = {}^\star K_{t^-}^N(B,1,\ell,e+1) + 1, \\ {}^\star K_t^N(T,0,\ell,e) = {}^\star K_{t^-}^N(T,0,\ell,e) - 1. \end{cases}$$

  - *Branched polymers.* An Arp2/3 complex can bind to each monomer that is not already attached to a complex of each branched polymer at rate Φ_A^+ and create a new branch. For each length $\ell \in \{3, 4, \cdots, N\}$, for each number of branching $b \in \{1, 2, \cdots, \ell - 1\}$ and for each number of free extremities $e \in \{0, 1, \cdots, b + 1\}$, the rate at which binding of an Arp2/3 happens on polymers of type (B, b, ℓ, e) at time t is $\Phi_A^+ {}^\star K_t^N(B, b, \ell, e)(\ell - b)$ and the new state of the system is:

$$ {}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t^-}^N - \delta_{(B,b,\ell,e)} + \delta_{(B,b+1,\ell,e+1)},$$

    or equivalently:

$$\begin{cases} {}^\star K_t^N(B,b+1,\ell,e+1) = {}^\star K_{t^-}^N(B,b+1,\ell,e+1) + 1, \\ {}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t^-}^N(B,b,\ell,e) - 1. \end{cases}$$

- **Release of an Arp2/3 complex.** Each Arp2/3 complex of a branched polymer can be released at rate Φ_A^- and cause the polymer to split. As a whole, there is no release of monomer, one branching is lost and the number of free extremities is unchanged. The same approach is used as for depolymerisation with splitting. For each length $\ell \in \{3, 4, \cdots, N\}$, for each number of branching $b \in \{1, 2, \cdots, \ell\}$ and for each number of free extremities $e \in \{0, 1, \cdots, b + 1\}$, the rate at which release of an Arp2/3 happens on polymers of type (B, b, ℓ, e) at time t is $\Phi_A^- {}^\star K_t^N(B, b, \ell, e)b$. As the topology of the (B, b, ℓ, e) type polymer is not know, the two new polymers will be constructed randomly according to the following procedure:
  - We randomly choose the length of the first polymer ℓ_1 according to a uniform distribution: $\ell_1 \sim \text{Unif}(\{0, 1, \cdots, \ell\})$.
  - We set the length of the second polymer ℓ_2 by: $\ell_2 = \ell - \ell_1$.
  - We randomly choose the number of branching of the first polymer b_1 according to a uniform distribution: $b_1 \sim \text{Unif}(\{\max(0, b - 1 - \ell_2), \cdots, \min(b - 1, \ell_1)\})$.
  - As a release of an Arp2/3 results in the loss of a branching, we have: $b_1 + b_2 = b - 1$ and we set the number of branching of the second polymer b_2 by: $b_2 = b - 1 - b_1$.
  - We randomly choose the number of free extremities of the first polymer e_1 according to a uniform distribution: $e_1 \sim \text{Unif}(\{\max(0, e - b_2 - 1), \cdots, \min(e, b_1 + 1)\})$.
  - We finally set the number of free extremities of the second polymer e_2 by: $e_2 = e - e_1$.

The new state of the system is then:

$$\begin{aligned}
{}^\star\mathcal{Z}_t^N &= {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,\ell,e)} + \delta_{(T_1,b_1,\ell_1,e_1)} + \delta_{(T_2,b_2,\ell_2,e_2)} \text{ if } \ell_1 \geq 3 \text{ and } \ell_2 \geq 3,\\
{}^\star\mathcal{Z}_t^N &= {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,\ell,e)} + \ell_2\delta_{(M,0,1,0)} + \delta_{(T_1,b_1,\ell_1,e_1)} \text{ if } \ell_1 \geq 3 \text{ and } \ell_2 < 3,\\
{}^\star\mathcal{Z}_t^N &= {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,\ell,e)} + \ell_1\delta_{(M,0,1,0)} + \delta_{(T_2,b_2,\ell_2,e_2)} \text{ if } \ell_1 < 3 \text{ and } \ell_2 \geq 3,\\
{}^\star\mathcal{Z}_t^N &= {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,\ell,e)} + \ell\delta_{(M,0,1,0)} \text{ if } \ell_1 < 3 \text{ and } \ell_2 < 3,
\end{aligned}$$

with $T_{1|2} = B$ if $b_{1|2} \geq 1$, $T_{1|2} = S$ if ($b_{1|2} = 0$ and $e_{1|2} = 1$) or $T_{1|2} = C$ if ($b_{1|2} = 0$ and $e_{1|2} = 0$), or equivalently:

$$\begin{cases}
{}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t-}^N(B,b,\ell,e) - 1,\\
{}^\star K_t^N(T_1,b_1,\ell_1,e_1) = {}^\star K_{t-}^N(T_1,b_1,\ell_1,e_1) + 1,\\
{}^\star K_t^N(T_2,b_2,\ell_2,e_2) = {}^\star K_{t-}^N(T_2,b_2,\ell_2,e_2) + 1,
\end{cases} \text{if } \ell_1 \geq 3 \text{ and } \ell_2 \geq 3,$$

$$\begin{cases}
{}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t-}^N(B,b,\ell,e) - 1,\\
{}^\star K_t^N(T_1,b_1,\ell_1,e_1) = {}^\star K_{t-}^N(T_1,b_1,\ell_1,e_1) + 1,\\
{}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) + \ell_2,
\end{cases} \text{if } \ell_1 \geq 3 \text{ and } \ell_2 < 3,$$

$$\begin{cases}
{}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t-}^N(B,b,\ell,e) - 1,\\
{}^\star K_t^N(T_2,b_2,\ell_2,e_2) = {}^\star K_{t-}^N(T_2,b_2,\ell_2,e_2) + 1,\\
{}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) + \ell_1,
\end{cases} \text{if } \ell_1 < 3 \text{ and } \ell_2 \geq 3,$$

$$\begin{cases}
{}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t-}^N(B,b,\ell,e) - 1,\\
{}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) + \ell,
\end{cases} \text{if } \ell_1 < 3 \text{ and } \ell_2 < 3.$$

❉ **Capping.**

- *Simple polymers.* Each simple polymer can bind with a capping protein at rate Φ_C and therefore change category independently of the other polymers. For each length $\ell \in \{3,4,\cdots,N\}$, the rate at which binding of a capping protein happens on polymers of length ℓ at time t is $\Phi_C {}^\star K_t^N(S,0,\ell,1)$ and the new state of the system is:

$$ {}^\star\mathcal{Z}_t^N = {}^\star\mathcal{Z}_{t-}^N - \delta_{(S,0,\ell,1)} + \delta_{(C,0,\ell,0)},$$

or equivalently:

$$\begin{cases}
{}^\star K_t^N(C,0,\ell,0) = {}^\star K_{t-}^N(C,0,\ell,0) + 1,\\
{}^\star K_t^N(S,0,\ell,1) = {}^\star K_{t-}^N(S,0,\ell,1) - 1.
\end{cases}$$

- *Branched polymers.* Each free extremity of a branched polymer can bind with a capping protein at rate Φ_C independently to the other extremities and the other polymers. For each length $\ell \in \{3,4,\cdots,N\}$, for each number of branching $b \in \{1,2,\cdots,\ell\}$ and for each number of free extremities $e \in \{1,2,\cdots,b+1\}$, the rate at which binding of a capping protein happens on polymers of type (B,b,ℓ,e) at time t is $\Phi_C {}^\star K_t^N(B,b,\ell,e)e$ and the new state of the system is:

$$ {}^\star\mathcal{Z}_t^N = {}^\star\mathcal{Z}_{t-}^N - \delta_{(B,b,\ell,e)} + \delta_{(B,b,\ell,e-1)},$$

or equivalently:

$$\begin{cases}
{}^\star K_t^N(B,b,\ell,e-1) = {}^\star K_{t-}^N(B,b,\ell,e-1) + 1,\\
{}^\star K_t^N(B,b,\ell,e) = {}^\star K_{t-}^N(B,b,\ell,e) - 1.
\end{cases}$$

- **Nucleation.** When 3 free monomers meet, a new polymer of length 3 is created at a rate λ_n. Therefore, the total rate of nucleation at time t is $\lambda_n {}^\star K_t^N(M,0,1,0)\frac{{}^\star K_t^N(M,0,1,0)-1}{N}\frac{{}^\star K_t^N(M,0,1,0)-2}{N}$ and the new state of the system is:

$$ {}^\star\mathcal{Z}_t^N = {}^\star\mathcal{Z}_{t-}^N - 3\delta_{(M,0,1,0)} + \delta_{(S,0,3,1)}, $$

or equivalently:

$$ \begin{cases} {}^\star K_t^N(S,0,3,1) = {}^\star K_{t-}^N(S,0,3,1) + 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) - 3. \end{cases} $$

- **Fragmentation.** Fragmentation is more complex to model, firstly because the total fragmentation rate must be carefully chosen to take into account the effects of cofilin (Figure 11) and secondly because it must be determined which portion of the polymer will be retained and which portion will be completely depolymerised.

  As cofilin binds mainly to ADP monomers, we are interested in the ADP portion of the polymer which we assume to be of length $\frac{L(t)}{D}$ at time t with $D > 1$ and where $L(t)$ is the length of the polymer at time t. Assuming that cofilin binds at rate c to an ADP monomer, the amount of cofilin bound to a polymer at time t is on average $\frac{cL(t)}{D}$. Finally, Figure 11 ensures that fragmentation occurs when there is an average amount of cofilin bound to the polymer. We then assume that the fragmentation rate at time t is of the form $\frac{cL(t)}{D}\left(L(t) - \frac{cL(t)}{D}\right) = \frac{c}{D}\left(1 - \frac{c}{D}\right)L(t)^2 = \Phi_T L(t)^2$.

  We now want to determine the length of the portion that will remain. The smaller the depolymerisation rate (λ^-) compared to the rate of elongation ($\lambda^+\frac{{}^\star K_t^N(M,0,1,0)}{N}$ or $\lambda_F^+\frac{{}^\star K_t^N(P,0,1,0)}{N}$), the more ADP monomers there are in the polymer and the smaller the portion that will remain. We then choose to use a truncated geometric law of parameter $\max\left(0, 1 - \frac{\lambda^-}{\text{rate of elongation}}\right)$ to determine the length of the remaining portion.

  - *Simple polymers.* For each length $\ell \in \{3,4,\cdots,N\}$, the rate at which fragmentation happens on simple polymers of length ℓ at time t is $\Phi_T {}^\star K_t^N(S,0,\ell,1)\ell^2$. The fragmentation of a simple polymer of length ℓ leads to the creation of a simple polymer of length $\ell' = g + 3$ where g is chosen randomly according to a geometric distribution on $\{0,1,\cdots,\ell-4\}$ of parameter $p = 1 - \frac{\lambda^- N}{\lambda^+ {}^\star K_t^N(M,0,1,0)}$, and the release of $\ell - \ell'$ monomers. The new state of the system is:

$$ {}^\star\mathcal{Z}_t^N = {}^\star\mathcal{Z}_{t-}^N - \delta_{(S,0,\ell,1)} + \delta_{(S,0,\ell',1)} + (\ell - \ell')\delta_{(M,0,1,0)}, $$

    or equivalently:

$$ \begin{cases} {}^\star K_t^N(S,0,\ell',1) = {}^\star K_{t-}^N(S,0,\ell',1) + 1, \\ {}^\star K_t^N(S,0,\ell,1) = {}^\star K_{t-}^N(S,0,\ell,1) - 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t-}^N(M,0,1,0) + \ell - \ell'. \end{cases} $$

  - *Polymers associated with a formin.* For each length $\ell \in \{3,4,\cdots,N\}$, the rate at which fragmentation happens on polymers associated with a formin of length ℓ at time t is $\Phi_T {}^\star K_t^N(F,0,\ell,1)\ell^2$. The fragmentation of a polymer associated with a formin of length ℓ leads to the creation of a polymer associated with a formin of length $\ell' = g + 3$ where g is chosen randomly according to a geometric distribution on $\{0,1,\cdots,\ell-4\}$ of parameter $p = 1 - \frac{\lambda^- N}{\lambda_F^+ {}^\star K_t^N(P,0,1,0)}$, and the release of $\ell - \ell'$ monomers. The new state of the system is:

$$ {}^\star\mathcal{Z}_t^N = {}^\star\mathcal{Z}_{t-}^N - \delta_{(F,0,\ell,1)} + \delta_{(F,0,\ell',1)} + (\ell - \ell')\delta_{(M,0,1,0)}, $$

or equivalently:

$$\begin{cases} {}^\star K_t^N(F,0,\ell',1) = {}^\star K_{t^-}^N(F,0,\ell',1) + 1, \\ {}^\star K_t^N(F,0,\ell,1) = {}^\star K_{t^-}^N(F,0,\ell,1) - 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t^-}^N(M,0,1,0) + \ell - \ell'. \end{cases}$$

- *Capped polymers.* For each length $\ell \in \{3,4,\cdots,N\}$, the rate at which fragmentation happens on capped polymers of length ℓ at time t is $\Phi_T {}^\star K_t^N(C,0,\ell,0)\ell^2$. Since when the polymer is capped elongation is impossible, we assume that all momomers of the polymer are ADP and that the polymer depolymerises entirely when fragmentation occurs. The new state of the system is:

$$ {}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t^-}^N - \delta_{(C,0,\ell,0)} + \ell\,\delta_{(M,0,1,0)},$$

or equivalently:

$$\begin{cases} {}^\star K_t^N(C,0,\ell,0) = {}^\star K_{t^-}^N(C,0,\ell,0) - 1, \\ {}^\star K_t^N(M,0,1,0) = {}^\star K_{t^-}^N(M,0,1,0) + \ell. \end{cases}$$

- *Branched polymers.* For each length $\ell \in \{3,4,\cdots,N\}$, for each number of branching $b \in \{1,2,\cdots,\ell\}$ and for each number of free extremities $e \in \{0,1,\cdots,b+1\}$, the rate at which fragmentation happens on branched polymers of type (B,b,ℓ,e) at time t is $\Phi_T {}^\star K_t^N(B,b,\ell,e)(\ell-b)^2$ (since we assume that cofilin is unable to bind to a monomer that is already bound to an Arp2/3 complex).

  As in the previous case, we assume that the capped branches depolymerise completely. The polymer will therefore lose at least $\max(1,c)$ monomers with $c = b+1-e$ the number of capped extremities. We now need to find a way to determine n_p the number of polymers obtained from the fragmentation. If $e = 0$, then all extremities are capped and the polymer depolymerizes entirely, so $n_p = 0$. If $e > 0$, we randomly choose n_p according to a uniform distribution: $n_p \sim \text{Unif}\left(\left\{1,2,\cdots,\min\left(e,\left\lfloor\frac{\ell-\max(1,c)}{3}\right\rfloor\right)\right\}\right)$. The condition $n_p \leq \left\lfloor\frac{\ell-\max(1,c)}{3}\right\rfloor$ ensures that we can create n_p "real" polymers of length greater than or equal to 3. We then determine ℓ' the number of monomers that will remain polymerised after fragmentation. Since the polymer loses its c capped branches, there are still $b-c = e-1$ branching to distribute after fragmentation; but as n_p polymers are created, n_p-1 branching (of these $e-1$) are already distributed and $b' = e - n_p$ branching remains. Therefore, we must have $\ell' > e - n_p$ and using the same idea as before, we choose $\ell' = g + \max(3n_p, e-n_p)$ where g is chosen randomly according to a geometric distribution on $\left\{0,1,\cdots,\ell - \max(1,c) - \max(3n_p, e-n_p)\right\}$ of parameter $p = 1 - \frac{\lambda^- N}{\lambda^+ {}^\star K_t^N(M,0,1,0)e}$. We could randomly choose the length ℓ_i and number of branching b_i of each of the n_p polymers according to a uniform distribution, but this would add a lot of randomness and make the model difficult to study. We then decide to create n_p similar polymers by choosing:

  - $\forall i \in \{1,2,\cdots,n_p-1\}$, $\ell_i = \left\lfloor\frac{\ell'}{n_p}\right\rfloor$ and $b_i = \left\lfloor\frac{b'}{n_p}\right\rfloor$,
  - $\ell_{n_p} = \ell' - (n_p-1)\ell_1$ and $b_{n_p} = b' - (n_p-1)b_1$,
  - $\forall i \in \{1,2,\cdots,n_p\}$, $e_i = b_i + 1$.

  The new state of the system is:

$$ {}^\star \mathcal{Z}_t^N = {}^\star \mathcal{Z}_{t^-}^N - \delta_{(B,b,\ell,e)} + (n_p-1)\delta_{(T_1,b_1,\ell_1,e_1)} + \delta_{(T_{n_p},b_{n_p},\ell_{n_p},e_{n_p})} + (\ell-\ell')\delta_{(M,0,1,0)},$$

with $T_{1|n_p} = B$ if $b_{1|n_p} \geq 1$ or $T_{1|n_p} = S$ if $b_{1|n_p} = 0$, or equivalently:

$$\begin{cases} {}^\star K_t^N(T_1, b_1, \ell_1, e_1) = {}^\star K_{t-}^N(T_1, b_1, \ell_1, e_1) + n_p - 1, \\ {}^\star K_t^N(T_{n_p}, b_{n_p}, \ell_{n_p}, e_{n_p}) = {}^\star K_{t-}^N(T_{n_p}, b_{n_p}, \ell_{n_p}, e_{n_p}) + 1, \\ {}^\star K_t^N(B, b, \ell, e) = {}^\star K_{t-}^N(B, b, \ell, e) - 1, \\ {}^\star K_t^N(M, 0, 1, 0) = {}^\star K_{t-}^N(M, 0, 1, 0) + \ell - \ell'. \end{cases}$$

Figure 3.11 summarizes all the events described above.

4.3 - Implementation

Since we want to study this model through simulations only, we have implemented this model in C++. We then adapted the code developed for the first model (presented in Appendix A); however, this did not allow us to produce satisfactory simulations. Indeed, in the case of the model in Chapter I, we use an array of size $\mathcal{O}(2N)$ (corresponding to the two types of polymers — simple and associated with a formin — for all lengths from 1 to N) to collect the quantity of each type of polymer. In the case of the model that takes into account all accessory proteins, we consider branched polymers and therefore need an array of size $\mathcal{O}(N^3)$ (corresponding to each possible combination (b, ℓ, e) for all length l, number of branches b and number of free extremities e from 1 to N). This space complexity then limits our simulations to values of $N \approx 100$ not large enough to obtain satisfactory results.

We were able to solve this problem by noticing that our array was sparse. We then rewrote the C++ code by replacing arrays with maps. Thanks to the class std :: map, we are able to store only the non-zero amounts of polymers and the associated types and thus reduce considerably the space complexity. The code presented in Appendix C allows us to obtain simulations for large enough and relevant values of N.

4.4 - Link between Model Parameters and Biological Values

One of the objectives of this model and its implementation is to answer biological questions. We then noted, for each phenomenon considered in our model, the values presented in the literature. These values are listed in Table 3.1.

In order to relate our parameters to these values, we used homogeneity checks. As all our parameters correspond to a number of events occurring in one unit of time, they must all be expressed in s^{-1}. In the literature, dissociation rates are already expressed in s^{-1} but association rates are mostly expressed in $\mu M^{-1}s^{-1}$ where the unit M corresponds to the molar concentration ($1\mu M = 10^{-6} mol/L$). When several particles are involved in the association (as in nucleation or association of an Arp2/3 comlpex with a polymer), the corresponding rates are expressed in $\mu M^{-2}s^{-1}$ or $\mu M^{-3}s^{-1}$.

- **Spontaneous elongation.** As we neither differentiate ADP and ATP monomers nor the pointed and barbed end of the filament, a weighted average is made to obtain an average elongation speed for a single polymer. This average elongation speed being expressed in $\mu M^{-1}s^{-1}$, we multiply it by the G-actin global cytoplasmic concentration to obtain our parameter λ^+.

$$\lambda^+ = \left(\left(k_{bT}^+ + k_{pT}^+\right) P_{cATP} + \left(k_{bD}^+ + k_{pD}^+\right)(1 - P_{cATP})\right)[Gact] \approx 130\ s^{-1}.$$

- **Elongation with a formin.** As the elongation speed with a formin in the presence of profilin is expressed in $\mu M^{-1}s^{-1}$, we multiply this value by the G-actin global cytoplasmic concentration

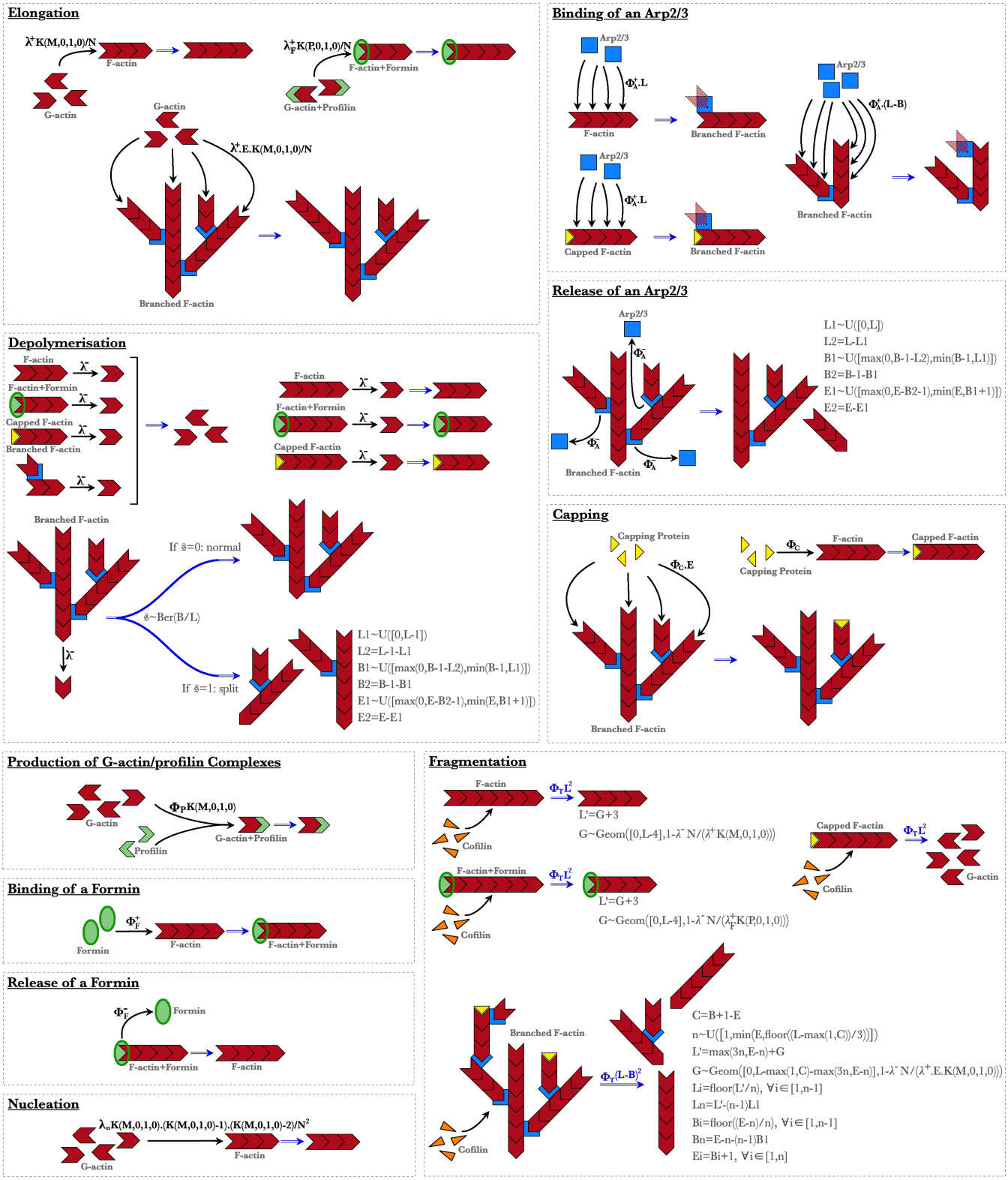


Figure 3.11 Schematic representation of the model considering all accessory proteins. The black arrows represent potential protein binding. The result of the transformation is indicated by the blue double arrow. The protein binding rates are shown in black. In the case of fragmentation, the rate of realisation of the event is specified in blue. For the branching event (binding of an Arp2/3 complex), the transparent unit corresponds to a site that can receive a monomer but there is no monomer yet on the new Arp2/3 complex. In cases where polymers are created randomly, the procedures are indicated in grey.

Description	Symbol	Value	References
G-actin global cytoplasmic concentration	$[Gact]$	10 μM	
Filament length increment per monomer	$\Delta\ell$	2.2 nm	[MEK02b]
Average length of filament	ℓ	6 μm	
ATP-G-actin association at barbed ends	k_{bT}^{+}	11.6 $\mu M^{-1}s^{-1}$	[BODM04] ([Pol86])
ATP-G-actin association at pointed ends	k_{pT}^{+}	1.3 $\mu M^{-1}s^{-1}$	[BODM04] ([Pol86])
ADP-G-actin association at barbed ends	k_{bD}^{+}	3.8 $\mu M^{-1}s^{-1}$	[BODM04] ([Pol86])
ADP-G-actin association at pointed ends	k_{pD}^{+}	0.16 $\mu M^{-1}s^{-1}$	[BODM04] ([Pol86])
Proportion of ATP-G-actin in cytoplasm	P_{cATP}	99%	
ATP-G-actin dissociation at barbed ends	k_{bT}^{-}	1.4 s^{-1}	[BODM04] ([Pol86])
ATP-G-actin dissociation at pointed ends	k_{pT}^{-}	0.8 s^{-1}	[BODM04] ([Pol86])
ADP-G-actin dissociation at barbed ends	k_{bD}^{-}	7.2 s^{-1}	[BODM04] ([Pol86])
ADP-G-actin dissociation at pointed ends	k_{pD}^{-}	0.3 s^{-1}	[BODM04]
Proportion of ATP-G-actin in filaments	P_{fATP}	10%	
Formin-initiated association at barbed ends (in the presence of profilin)	k_{F}^{+}	660 $\mu M^{-1}s^{-1}$	[HNM+09] [NSK08] [CPGP+21] ([RLCD+04])
Spontaneous nucleation of filaments	k_{Nucl}	$2 \cdot 10^{-8}$ $\mu M^{-2}s^{-1}$	[HNM+09] ([SM01])
Profilin-ATP-G-actin association rate	$k_{Prof\,T}^{+}$	1 $\mu M^{-1}s^{-1}$	[BODM04]
Profilin global cytoplasmic concentration	$[Prof]$	10 μM	[MEK02b]
Association of formin to barbed ends	k_{Form}^{+}	50 $\mu M^{-1}s^{-1}$	[CS17] ([RLCD+04])
Detachment of formin from barbed ends	k_{Form}^{-}	$7.5 \cdot 10^{-1}$ s^{-1}	[CPGP+21] [HNM+09]
Formin global cytoplasmic concentration	$[Form]$	0.04 μM	[WP05]
Arp2/3 association to F-actin	k_{Arp}^{+}	$5.4 \cdot 10^{-4}$ $\mu M^{-3}s^{-1}$	[HNM+09] ([CWC04])
Dissociation of Arp2/3 from pointed ends	k_{Arp}^{-}	$1.8 \cdot 10^{-3}$ s^{-1}	[HNM+09] ([CWC04])
Arp2/3 global cytoplasmic concentration	$[Arp]$	4 μM	[WP05]
Capping of barbed ends	k_{Cap}^{+}	3.0 $\mu M^{-1}s^{-1}$	[HNM+09] ([SJC96])
Capping protein global cytoplasmic conc.	$[Cap]$	1.19 μM	[WP05]
Severing rate	k_{Frag}	10^{-3} $\mu m^{-1}s^{-1}$	[CP11]
ADF/Cofilin global cytoplasmic conc.	$[Cof]$	10 μM	[MEK02b]

Table 3.1 Biological values for each protein and phenomenon considered in the model.

to obtain our parameter λ_F^+.

$$\lambda_F^+ = k_F^+[Gact] = 6600\ s^{-1}.$$

- **Depolymerisation.** As for elongation, we consider that our depolymerisation rate is global. To obtain our parameter λ^-, we then calculate a weighted average of the different depolymerisation rates.

$$\lambda^- = \left(k_{bT}^- + k_{pT}^-\right) P_{fATP} + \left(k_{bD}^- + k_{pD}^-\right)(1 - P_{fATP}) \approx 7\ s^{-1}.$$

- **Nucleation.** As the rate of spontaneous nucleation of filaments is expressed in $\mu M^{-2}s^{-1}$, to obtain our parameter λ_n, we multiply this value by $[Gact]^2$ where $[Gact]$ is the G-actin global cytoplasmic concentration.

$$\lambda_n = k_{Nucl}[Gact]^2 = 2 \times 10^{-6}\ s^{-1}.$$

- **Production of G-actin/profilin complexes.** As the rate of association of profilin with G-actin is expressed in $\mu M^{-1}s^{-1}$, we multiply this value by the profilin global cytoplasmic concentration to obtain our parameter Φ_P.

$$\Phi_P = k_{Prof\,T}^+[Prof] = 10\ s^{-1}.$$

⁕ **Binding of a formin.** As the rate of association of formin to barbed ends is expressed in $\mu M^{-1}s^{-1}$, we multiply this value by the formin global cytoplasmic concentration to obtain our parameter Φ_F^+.

$$\Phi_F^+ = k_{Form}^+[Form] = 2\ s^{-1}.$$

⁕ **Release of a formin.** As the rate of detachment of formin from barbed ends is already expressed in s^{-1}, we have:

$$\Phi_F^- = k_{Form}^- = 7.5 \times 10^{-1}\ s^{-1}.$$

⁕ **Binding of an Arp2/3.** As the rate of association of Arp2/3 to F-actin is expressed in $\mu M^{-3}s^{-1}$, to obtain our parameter Φ_A^+, we multiply this value by $[Arp]^3$ where $[Arp]$ is the Arp2/3 global cytoplasmic concentration.

$$\Phi_A^+ = k_{Arp}^+[Arp]^3 \approx 3.5 \times 10^{-2}\ s^{-1}.$$

⁕ **Release of an Arp2/3.** As the rate of dissociation of Arp2/3 from F-actin is already expressed in s^{-1}, we have:

$$\Phi_A^- = k_{Arp}^- = 1.8 \times 10^{-3}\ s^{-1}.$$

⁕ **Capping.** As the rate of capping of barbed ends is expressed in $\mu M^{-1}s^{-1}$, we multiply this value by the capping protein global cytoplasmic concentration to obtain our parameter Φ_C.

$$\Phi_C = k_{Cap}^+[Cap] \approx 3.6\ s^{-1}.$$

⁕ **Fragmentation.** As the severing rate is expressed in $\mu m^{-1}s^{-1}$ and our total fragmentation rate already takes into account the length of the polymer, we multiply this value by the filament length increment per monomer (length of a monomer) to obtain our parameter Φ_T.

$$\Phi_T = k_{Frag}\Delta\ell = 2,2 \times 10^{-6}\ s^{-1}.$$

Table 3.2 summarises the approximate values of each parameter.

Description	Symbol	Value
Rate of spontaneous elongation	λ^+	$130\ s^{-1}$
Rate of elongation with a formin	λ_F^+	$6600\ s^{-1}$
Rate of depolymerisation	λ^-	$7\ s^{-1}$
Rate of nucleation	λ_n	$2 \cdot 10^{-6}\ s^{-1}$
Rate of production of G-actin/profilin complexes	Φ_P	$10\ s^{-1}$
Rate of binding of a formin	Φ_F^+	$2\ s^{-1}$
Rate of release of a formin	Φ_F^-	$7.5 \cdot 10^{-1}\ s^{-1}$
Rate of binding of an Arp2/3	Φ_A^+	$3.5 \cdot 10^{-2}\ s^{-1}$
Rate of release of an Arp2/3	Φ_A^-	$1.8 \cdot 10^{-3}\ s^{-1}$
Rate of capping	Φ_C	$3.6\ s^{-1}$
Rate of fragmentation	Φ_T	$2.2 \cdot 10^{-6}\ s^{-1}$

Table 3.2 Approximate values for the model parameters, obtained from the biological values.

4.5 - Observation of a Random Realisation

In this section we present the results obtained using the parameters found in the previous section. The objective of this section is more to illustrate the possibilities offered by the implementation than to draw conclusions on the behaviour of the model. Indeed, we are only considering one random realisation and we have seen that the results obtained using the same parameters and initial values are very variable.

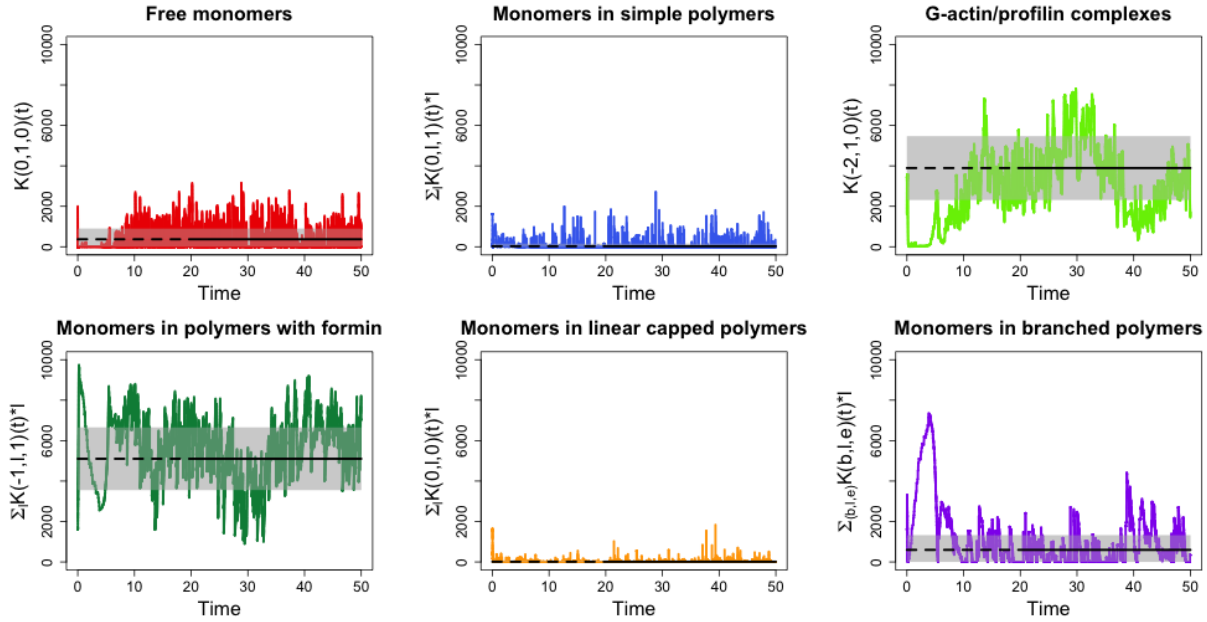


Figure 3.12 Evolution of the system (into 6 sub-populations) described by the model with all accessory proteins with $N = 10000$, $\lambda^+ = 130$, $\lambda^- = 7$, $\lambda_F^+ = 6600$, $\lambda_n = 2 \cdot 10^{-6}$, $\Phi_P = 10$, $\Phi_F^+ = 2$, $\Phi_F^- = 0.75$, $\Phi_A^+ = 0.035$, $\Phi_A^- = 0.0018$, $\Phi_C = 3.6$ and $\Phi_T = 2.2 \cdot 10^{-6}$, starting with 20% of free monomers, 16% of G-actin/profilin complexes and 64% of monomers included within polymers (equitably distributed between simple polymers, polymers bound with a formin, linear capped polymers and branched polymers) of length 40 (and with 6 branching and 3 free extremities for branched polymers). Top left: evolution of the number of free monomers. Top middle: evolution of the number of monomers included in a simple polymer. Top right: evolution of the number of G-actin/profilin complexes. Bottom left: evolution of the number of monomers included in a polymer bound with a formin. Bottom middle: evolution of the number of monomers included in a linear capped polymer. Bottom right: evolution of the number of monomers included in a branched polymer. On each graph, we plot a single realisation of the stochastic system in color, the average value in black and the error interval around the mean in grey (see text).

Figure 3.12 shows the repartition of the N monomers over time in the six sub-populations considered. For all the simulations presented in the following, we chose $N = 10000$, which seems to be a good trade-off between a sufficiently large number of monomers and a not too long execution time.

Figure 3.12 is divided into six graphs. On the top left (in red), we plot the number of free monomers $t \mapsto {}^\star K_t^N(M, 0, 1, 0)$. In the middle top row (in blue), we plot the number of monomers polymerised within simple polymers $t \mapsto {}^\star M_{inS}^N(t)$. On the top right (in light green), we plot the number of G-actin/profilin complexes $t \mapsto {}^\star K_t^N(P, 0, 1, 0)$. On the bottom left (in dark green), we plot the number of monomers polymerised within polymers bound with a formin $t \mapsto {}^\star M_{inF}^N(t)$. In the middle bottom row (in yellow), we plot the number of monomers polymerised within linear capped polymers $t \mapsto {}^\star M_{inC}^N(t)$. On the bottom right (in purple), we plot the number of monomers polymerised within branched poly-

mers $t \mapsto {}^\star M^N_{inB}(t)$. On each graph, the average value is plotted in black. This value is calculated using only the last two thirds of the time interval (the dotted part is therefore not included in the calculation). We also show the error interval $[m - \sigma, m + \sigma]$ (where m is the average value and σ is the standard deviation) of this value in grey.

In this realisation, it would seem that an equilibrium is reached with 50% of the monomers within polymers bound with a formin, 40% of the monomers within G-actin/profilin complexes, and the rest distributed equally between branched polymers and free monomers. On many other realisations, we observed a different equilibrium in which all polymers had disappeared (${}^\star M^N_{inP} = 0$) and all monomers were in G-actin/profilin complexes. However, by choosing a larger N, we have noticed that this "null" equilibrium is only very rarely achieved.

Figure 3.13 shows the percentage of G-actin (free monomers + G-actin/profilin complexes) and the percentage of F-actin ($t \mapsto {}^\star M^N_{inP}(t)$) over time. By grouping the quantities of monomers like this, the equilibrium that seemed to exist in Figure 3.12 is no longer so clear. It would appear that in this case about 60% of the actin is in polymerised form. This is a little high, according to the biological data, there would be only 30% of polymerised actin.

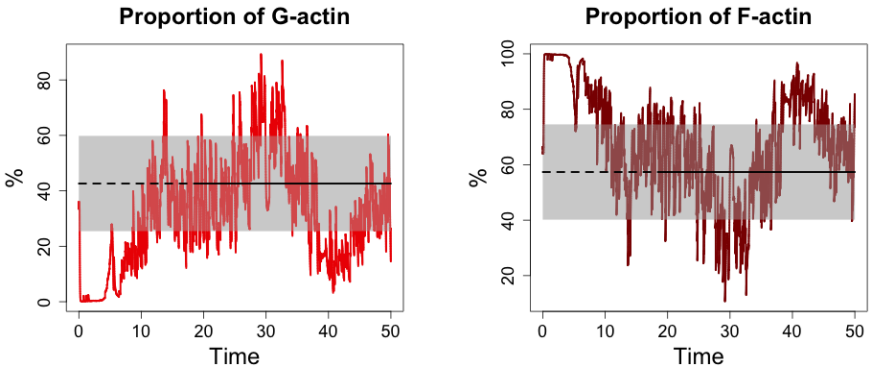


Figure 3.13 Percentage of G-actin and F-actin over time with $N = 10000$, $\lambda^+ = 130$, $\lambda^- = 7$, $\lambda^+_F = 6600$, $\lambda_n = 2 \cdot 10^{-6}$, $\Phi_P = 10$, $\Phi^+_F = 2$, $\Phi^-_F = 0.75$, $\Phi^+_A = 0.035$, $\Phi^-_A = 0.0018$, $\Phi_C = 3.6$ and $\Phi_T = 2.2 \cdot 10^{-6}$, starting with 20% of free monomers, 16% of G-actin/profilin complexes and 64% of monomers included within polymers (equitably distributed between simple polymers, polymers bound with a formin, linear capped polymers and branched polymers) of length 40 (and with 6 branching and 3 free extremities for branched polymers). Left: percentage of G-actin. Right: percentage of F-actin. On each graph, we plot a single realisation of the stochastic system in color, the average value in black and the error interval around the mean in grey.

Figure 3.14 is divided into eight graphs. For each four sub-populations of polymers (simple, bound with a formin, linear capped and branched), we plot on the top graph the percentage of polymerised actin in this sub-population (${}^\star M^N_{inS}(t)/{}^\star M^N_{inP}(t)$ in blue ,${}^\star M^N_{inF}(t)/{}^\star M^N_{inP}(t)$ in green , ${}^\star M^N_{inC}(t)/{}^\star M^N_{inP}(t)$ in yellow and ${}^\star M^N_{inB}(t)/{}^\star M^N_{inP}(t)$ in purple) and on the bottom graph the percentage of polymers in this sub-population (${}^\star S^N(t)/{}^\star P^N(t)$, ${}^\star F^N(t)/{}^\star P^N(t)$, ${}^\star C^N(t)/{}^\star P^N(t)$ and ${}^\star B^N(t)/{}^\star P^N(t)$). The average value and its error interval are also plot on each graph. This figure allows us to compare the four sub-populations but also to deduce whether the polymers of each sub-population are rather long or short.

We observe on this realisation that the monomers are mainly in polymers bound with a formin and branched polymers (about 90 % in polymers bound with a formin and about 10 % in branched polymers). We also observe that about 70% of the polymers are bound with a formin, 25% are branched and 5% are linear capped. We then deduce that polymers bound with a formin are rather long while the others are rather short.

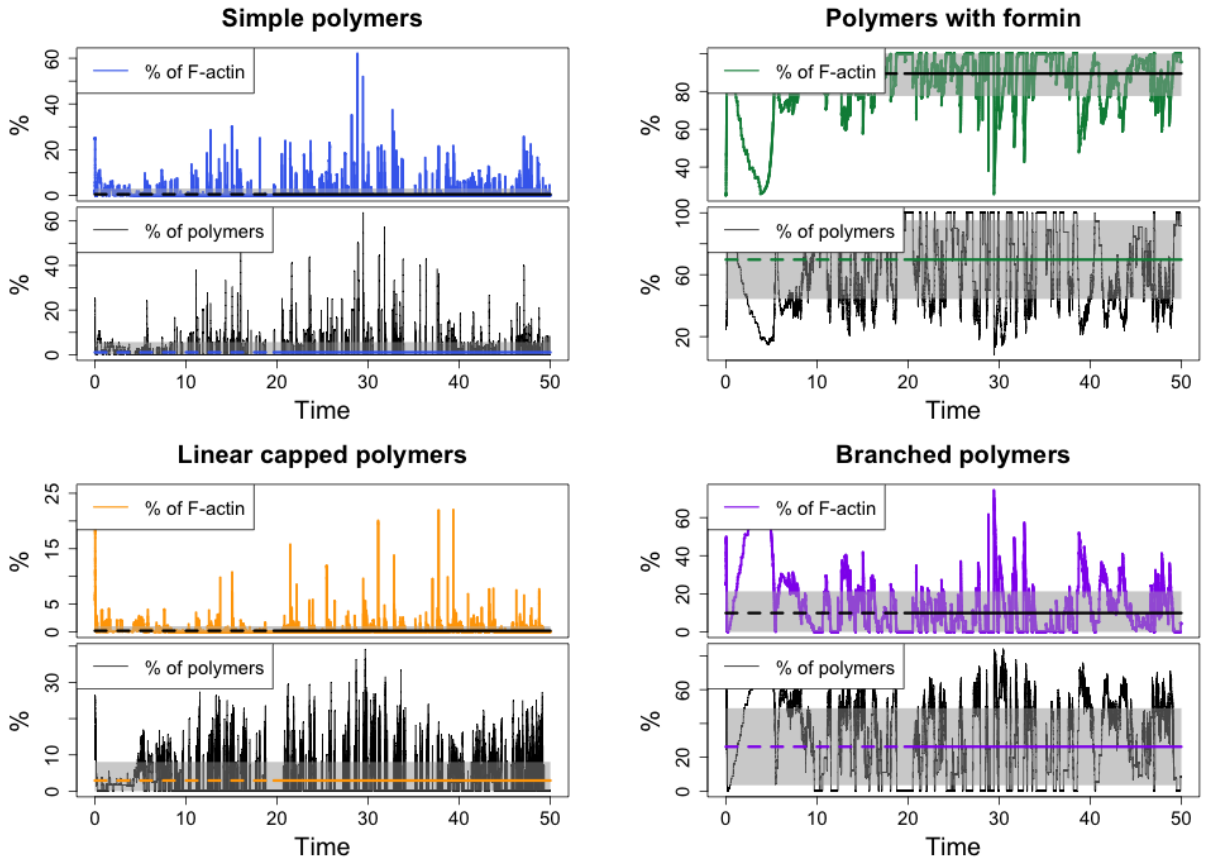


Figure 3.14 Percentage of polymerised actin and polymers in each of the 4 polymer sub-populations over time with $N = 10000$, $\lambda^+ = 130$, $\lambda^- = 7$, $\lambda_F^+ = 6600$, $\lambda_n = 2 \cdot 10^{-6}$, $\Phi_P = 10$, $\Phi_F^+ = 2$, $\Phi_F^- = 0.75$, $\Phi_A^+ = 0.035$, $\Phi_A^- = 0.0018$, $\Phi_C = 3.6$ and $\Phi_T = 2.2 \cdot 10^{-6}$, starting with 20% of free monomers, 16% of G-actin/profilin complexes and 64% of monomers included within polymers (equitably distributed between simple polymers, polymers bound with a formin, linear capped polymers and branched polymers) of length 40 (and with 6 branching and 3 free extremities for branched polymers). Top left: percentage of polymerised actin (top) in simple polymers and percentage of simple polymers (bottom). Top right: percentage of polymerised actin (top) in polymers bound with a formin and percentage of polymers bound with a formin (bottom). Bottom left: percentage of polymerised actin (top) in linear capped polymers and percentage of linear capped polymers (bottom). Bottom right: percentage of polymerised actin (top) in branched polymers and percentage of branched polymers (bottom).

Figure 3.15 shows the distribution of polymer lengths at final time. For this realisation, there are very few polymers left, this is probably due to a too small value of N. Nevertheless, it seems that polymers bound with a formin are quite long (up to 1000 monomers), whereas branched polymers have a medium length (up to 350 monomers).

Finally, Figure 3.16 gives information on the topology of branched polymers. On the left, we plot (in purple) the evolution of the average number of branching per polymer over time. On the right, we plot (in yellow) the evolution of the average number of free extremities per polymer over time. In this case, the polymers appear to have few branching (less than 10 on average) with less than half of the extremities free (2-3 on average).

As mentioned earlier, there is significant variability in the results for this set of parameters. It would therefore be very useful to determine an average behaviour as we did in Chapter I. Based on this realisation, the results obtained are different from those expected and observed in biology. It would

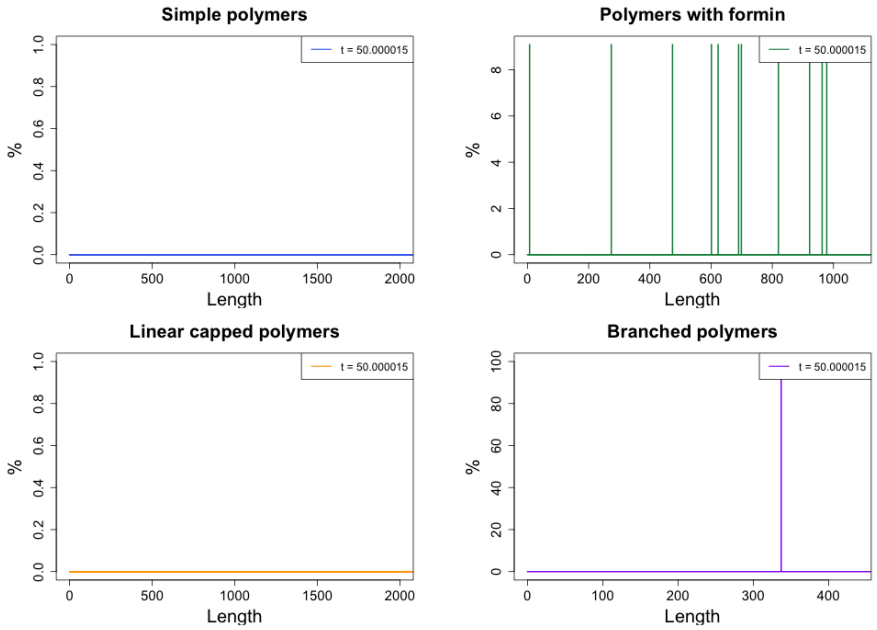


Figure 3.15 Distribution of the polymer length in each of the 4 polymer sub-populations at final time with $N = 10000$, $\lambda^+ = 130$, $\lambda^- = 7$, $\lambda_F^+ = 6600$, $\lambda_n = 2 \cdot 10^{-6}$, $\Phi_P = 10$, $\Phi_F^+ = 2$, $\Phi_F^- = 0.75$, $\Phi_A^+ = 0.035$, $\Phi_A^- = 0.0018$, $\Phi_C = 3.6$ and $\Phi_T = 2.2 \cdot 10^{-6}$, starting with 20% of free monomers, 16% of G-actin/profilin complexes and 64% of monomers included within polymers (equitably distributed between simple polymers, polymers bound with a formin, linear capped polymers and branched polymers) of length 40 (and with 6 branching and 3 free extremities for branched polymers). Top left: distribution of the simple polymer length. Top right: distribution of the polymer bound with a formin length. Bottom left: distribution of the linear capped polymer length. Bottom right: distribution of the branched polymer length.

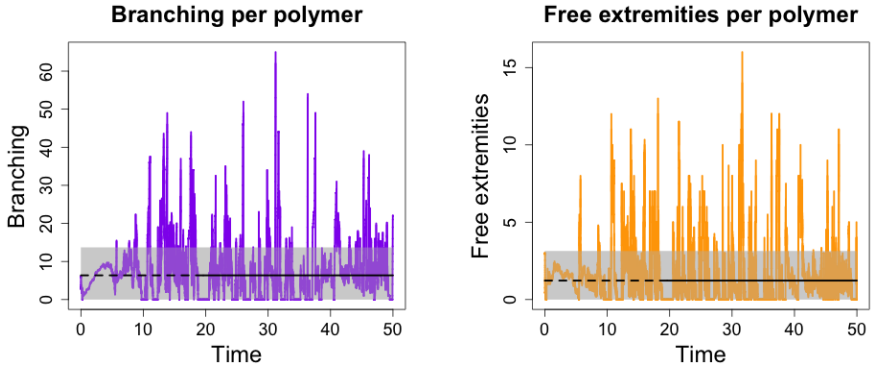


Figure 3.16 Average number of branching and of free extremities per polymer over time with $N = 10000$, $\lambda^+ = 130$, $\lambda^- = 7$, $\lambda_F^+ = 6600$, $\lambda_n = 2 \cdot 10^{-6}$, $\Phi_P = 10$, $\Phi_F^+ = 2$, $\Phi_F^- = 0.75$, $\Phi_A^+ = 0.035$, $\Phi_A^- = 0.0018$, $\Phi_C = 3.6$ and $\Phi_T = 2.2 \cdot 10^{-6}$, starting with 20% of free monomers, 16% of G-actin/profilin complexes and 64% of monomers included within polymers (equitably distributed between simple polymers, polymers bound with a formin, linear capped polymers and branched polymers) of length 40 (and with 6 branching and 3 free extremities for branched polymers). Left: average number of branching per polymer. Right: average number of free extremities per polymer. On each graph, we the average value in black and the error interval around the mean in grey.

therefore also be useful to conduct a numerical exploration of this model. This would allow us to determine which parameters are robust and possibly adapt the rounding value for the most sensitive parameters. It would also allow us to determine the impact of each protein on the dynamics of the system.

4.6 - Perspectives

4.6.1 - Qualitative study of behaviour and concrete biological applications

The main objective of this model and its implementation was to answer concrete biological questions. In particular, we intended to study the following three topics:

1. During cell division, there is a modulation of actin polymerisation. We would like to determine the effects of a switch from a mixed population — composed of linear polymers (mainly polymers bound with a formin) and branched polymers — to a population composed exclusively of linear polymers. In our model, this would correspond to a switch from $\Phi_A^+ > 0$ to $\Phi_A^+ = 0$, and our implementation would answer the following questions:
   - How long does it take to lose all the branched polymers?
   - What is the resilience time?
   - What is the impact on polymer length?
   - What is the impact on the amount of polymerised actin?
2. During pulsed contractions in C. elegans, there is a periodic modulation (as a square wave) of the concentration of formin, which implies a periodic modulation of the rate of binding of a formin to a polymer. There is also a modulation of the concentration of capping proteins and thus of the rate of capping.
   (a) If we are only considering the modulation of the formin concentration, this corresponds in our model to alternating between the normal value of Φ_F^+ for about 30 seconds and a higher value (e.g. $10 \cdot \Phi_F^+$) for about 5-10 seconds.
   (b) In a second step, we could take into account the modulation of the capping protein concentration. This would then correspond to alternating between normal Φ_F^+ and Φ_C values for about 30 seconds and higher values (e.g. $10 \cdot \Phi_F^+$ and $10 \cdot \Phi_C$) for about 5-10 seconds, with a delay between the modulation of the formin concentration and the modulation of the capping protein concentration of about 2-5 seconds.

   Our model would answer the following questions:
   - Is there a new steady state?
   - What is the impact on polymer length?
   - What is the impact on the amount of polymerised actin?

Although we have tried to answer these questions, we have not been able to draw any conclusions because the results obtained are not consistent. This can be due to various causes, such as a value of N that is too small or parameter values that are too approximate. While taking a larger value for N may be difficult due to the complexity in time of the implementation, a qualitative study of the behaviour should allow the parameters to be adapted.

4.6.2 - Mathematical conjecture

We also intended to use our implementation to validate the following mathematical conjecture. Conjecture 3.4.1 states that if we study the large population limit, the measure-valued process $\left(\frac{1}{N}{}^\star\mathcal{Z}_t^N\right)_{t\in\mathbb{R}_+}$ converges in distribution to a deterministic limit which can be characterised by a system of ordinary differential equations.

Conjecture 3.4.1
Suppose that the initial condition $\frac{1}{N}{}^\star\mathcal{Z}_0^N$ converges in distribution to a deterministic limit ${}^\star\mathcal{Z}_0$ as $N \longrightarrow +\infty$. Then the process $\left(\frac{1}{N}{}^\star\mathcal{Z}_t^N\right)_{t\in\mathbb{R}_+}$ converges in distribution as $N \longrightarrow +\infty$, to a deterministic limit $({}^\star\mathcal{Z}_t)_{t\in\mathbb{R}_+}$ which can be characterised as follows.

If for every $(T, b, l, e) \in {}^\star\mathcal{I}$ we write ${}^\star k_t(T, b, l, e)$ for the mass ${}^\star\mathcal{Z}_t(\{(T, b, l, e)\})$ given by ${}^\star\mathcal{Z}_t$ to the point (T, b, l, e), then:

$$ {}^\star k_0(T, b, l, e) = {}^\star\mathcal{Z}_0(\{(T, b, l, e)\}), $$

and

$$
\begin{aligned}
&\frac{d}{dt}{}^\star k_t(M,0,1,0) = -\lambda^+ {}^\star k_t(M,0,1,0){}^\star S^\infty(t) - \lambda^+ {}^\star k_t(M,0,1,0)\sum_{l=3}^{+\infty}\sum_{b=1}^{l}\sum_{e=1}^{b+1} e\,{}^\star k_t(B,b,l,e)\\
&+ 2\lambda^-\left({}^\star k_t(S,0,3,1) + {}^\star k_t(F,0,3,1) + {}^\star k_t(C,0,3,0) + \sum_{b=1}^{3}\sum_{e=0}^{b+1}{}^\star k_t(B,b,3,e)\right)\\
&+ \lambda^-\left({}^\star S^\infty(t) + {}^\star F^\infty(t) + {}^\star C^\infty(t) + {}^\star B^\infty(t)\right)\\
&+ \lambda^-\sum_{l=4}^{+\infty}\sum_{b=1}^{l}\sum_{e=0}^{b+1}{}^\star k_t(B,b,l,e)\sum_m (m-1)\mathbb{P}\text{ (release } m \text{ monomers by depolymerisation with split of a } (B,b,l,e))\\
&- \lambda_n\left({}^\star k_t(M,0,1,0)\right)^3 - \Phi_P\,{}^\star k_t(M,0,1,0)\\
&+ \Phi_A^-\sum_{l=3}^{+\infty}\sum_{b=1}^{l}\sum_{e=0}^{b+1} b\,{}^\star k_t(B,b,l,e)\sum_m m\mathbb{P}\text{ (release } m \text{ monomers by releasing an Arp2/3 from a } (B,b,l,e))\\
&+ \Phi_T\sum_{l=3}^{+\infty} l^2\,{}^\star k_t(S,0,l,1)\sum_m m\mathbb{P}\text{ (release } m \text{ monomers by fragmentation of a } (S,0,l,1))\\
&+ \Phi_T\sum_{l=3}^{+\infty} l^2\,{}^\star k_t(F,0,l,1)\sum_m m\mathbb{P}\text{ (release } m \text{ monomers by fragmentation of a } (F,0,l,1))\\
&+ \Phi_T\sum_{l=3}^{+\infty} l^3\,{}^\star k_t(C,0,l,0)\\
&+ \Phi_T\sum_{l=3}^{+\infty}\sum_{b=1}^{l}\sum_{e=0}^{b+1}(l-b)^2\,{}^\star k_t(B,b,l,e)\sum_m m\mathbb{P}\text{ (release } m \text{ free monomers by fragmentation of a } (B,b,l,e))
\end{aligned}
$$

$$ \frac{d}{dt}{}^\star k_t(P,0,1,0) = \Phi_P\,{}^\star k_t(M,0,1,0) - \lambda_F^+\,{}^\star k_t(P,0,1,0)\,{}^\star F^\infty(t) $$

$$
\begin{aligned}
&\forall L \geq 3,\ \frac{d}{dt}{}^{\star}k_t(S,0,L,1)\\
&= \lambda_n {}^{\star}k_t(M,0,1,0)^3 \mathbb{1}_{\{L=3\}} + \lambda^+ {}^{\star}k_t(M,0,1,0) {}^{\star}k_t(S,0,L-1,1)\mathbb{1}_{\{L>3\}} + \lambda^- {}^{\star}k_t(S,0,L+1,1) + \Phi_F^- {}^{\star}k_t(F,0,L,1)\\
&+ \lambda^- \sum_{l=L+1}^{+\infty}\sum_{b=1}^{l}\sum_{e=0}^{b+1} {}^{\star}k_t(B,b,l,e)\mathbb{P}\text{ (create a }(S,0,L,1)\text{ by depolymerisation with split of a }(B,b,l,e))\\
&+ \Phi_A^- \sum_{l=L}^{+\infty}\sum_{b=1}^{l}\sum_{e=0}^{b+1} b\,{}^{\star}k_t(B,b,l,e)\mathbb{P}\text{ (create a }(S,0,L,1)\text{ by releasing an Arp2/3 from a }(B,b,l,e))\\
&+ \Phi_T \sum_{l=L+1}^{+\infty} l^2 {}^{\star}k_t(S,0,l,1)\mathbb{P}\text{ (create a }(S,0,L,1)\text{ by fragmentation of a }(S,0,l,1))\\
&+ \Phi_T \sum_{l=L+1}^{+\infty}\sum_{b=1}^{l}\sum_{e=0}^{b+1}(l-b)^2 {}^{\star}K_t^N(B,b,l,e)\mathbb{P}\text{ (create a }(S,0,L,1)\text{ by fragmentation of a }(B,b,l,e))\\
&- \lambda^+ {}^{\star}k_t(M,0,1,0) {}^{\star}k_t(S,0,L,1) - \lambda^- {}^{\star}k_t(S,0,L,1) - \Phi_F^+ {}^{\star}k_t(S,0,L,1) - L\Phi_A^+ {}^{\star}k_t(S,0,L,1)\\
&- \Phi_C {}^{\star}k_t(S,0,L,1) - L^2\Phi_T {}^{\star}k_t(S,0,L,1),
\end{aligned}
$$

$$
\begin{aligned}
&\forall L \geq 3,\ \frac{d}{dt}{}^{\star}k_t(F,0,L,1) = \lambda_F^+ {}^{\star}k_t(P,0,1,0) {}^{\star}k_t(F,0,L-1,1)\mathbb{1}_{\{L>3\}} + \lambda^- {}^{\star}k_t(F,0,L+1,1) + \Phi_F^+ {}^{\star}k_t(S,0,L,1)\\
&+ \Phi_T \sum_{l=4}^{+\infty} l^2 {}^{\star}k_t(F,0,l,1)\mathbb{P}\text{ (create a }(F,0,L,1)\text{ by fragmentation of a }(F,0,l,1))\\
&- \lambda_F^+ {}^{\star}k_t(P,0,1,0) {}^{\star}k_t(F,0,L,1) - \lambda^- {}^{\star}k_t(F,0,L,1) - \Phi_F^- {}^{\star}k_t(F,0,L,1) - L^2\Phi_T {}^{\star}k_t(F,0,L,1),
\end{aligned}
$$

$$
\begin{aligned}
&\forall L \geq 3,\ \frac{d}{dt}{}^{\star}k_t(C,0,L,0)\\
&= \lambda^- {}^{\star}k_t(C,0,L+1,0) + \Phi_C {}^{\star}k_t(S,0,L,1)\\
&+ \lambda^- \sum_{l=L+1}^{+\infty}\sum_{b=1}^{l}\sum_{e=0}^{b+1} {}^{\star}k_t(B,b,l,e)\mathbb{P}\text{ (create a }(C,0,L,0)\text{ by depolymerisation with split of a }(B,b,l,e))\\
&+ \Phi_A^- \sum_{l=L}^{+\infty}\sum_{b=1}^{l}\sum_{e=0}^{b+1} b\,{}^{\star}k_t(B,b,l,e)\mathbb{P}\text{ (create a }(C,0,L,0)\text{ by releasing an Arp2/3 from a }(B,b,l,e))\\
&- \lambda^- {}^{\star}k_t(C,0,L,0) - L\Phi_A^+ {}^{\star}k_t(C,0,L,0) - L^2\Phi_T {}^{\star}k_t(C,0,L,0),
\end{aligned}
$$

$$
\begin{aligned}
&\forall B \geq 1, B \leq L, \forall E \geq 1, E \leq B+1,\ \frac{d}{dt}{}^{\star}k_t(B,B,L,E)\\
&= E\lambda^+ {}^{\star}k_t(M,0,1,0) {}^{\star}k_t(B,B,L-1,E)\mathbb{1}_{\{L>3,B>1,E\geq 1\}} + (L-B+1)\Phi_A^+ {}^{\star}k_t(B,B-1,L,E-1)\mathbb{1}_{\{B>1,E\leq 1\}}\\
&+ \Phi_A^+ {}^{\star}k_t(C,0,L,0)\mathbb{1}_{\{B=1,E=1\}} + \Phi_A^+ {}^{\star}k_t(S,0,L,1)\mathbb{1}_{\{B=1,E=2\}} + (E+1)\Phi_C {}^{\star}k_t(B,B,L,E+1)\mathbb{1}_{\{E\leq B\}}\\
&+ \lambda^- {}^{\star}k_t(B,B,L+1,E)\mathbb{P}\text{ (depolymerisation without split of a }(B,B,L+1,E))\\
&+ \lambda^- \sum_{l=L+1}^{+\infty}\sum_{b=B+1}^{l}\sum_{e=E}^{b+1} {}^{\star}k_t(B,b,l,e)\mathbb{P}\text{ (create a }(B,B,L,E)\text{ by depolymerisation with split of a }(B,b,l,e))\\
&+ \Phi_A^- \sum_{l=L}^{+\infty}\sum_{b=B+1}^{l}\sum_{e=E}^{b+1} b\,{}^{\star}k_t(B,b,l,e)\mathbb{P}\text{ (create a }(B,B,L,E)\text{ by releasing an Arp2/3 from a }(B,b,l,e))\\
&+ \Phi_T \sum_{l=L+1}^{+\infty}\sum_{b=B}^{l}\sum_{e=E}^{b+1}(l-b)^2 {}^{\star}K_t^N(B,b,l,e)\mathbb{P}\text{ (create a }(B,B,L,E)\text{ by fragmentation of a }(B,b,l,e))\,\mathbb{1}_{\{E>0\}}\\
&- E\lambda^+ {}^{\star}k_t(M,0,1,0) {}^{\star}k_t(B,B,L,E) - \lambda^- {}^{\star}k_t(B,B,L,E) - (L-B)\Phi_A^+ {}^{\star}k_t(B,B,L,E)\\
&- B\Phi_A^- {}^{\star}k_t(B,B,L,E) - E\Phi_C {}^{\star}k_t(B,B,L,E) - (L-B)^2\Phi_T {}^{\star}k_t(B,B,L,E),
\end{aligned}
$$

with:

- $\mathbb{P}$ (depolymerisation without split of a $(B, B, L+1, E)$) $= \mathbb{P}\left(\text{Ber}\left(\frac{B}{L+1}\right) = 0\right) = 1 - \frac{B}{L+1}$.
- $\mathbb{P}$ (create a (B, B, L, E) by depolymerisation with split of a (B, b, l, e))

$$
\begin{aligned}
&= \mathbb{P}\left(\text{Ber}\left(\frac{b}{l}\right) = 1\right) \mathbb{P}\text{ (create a (B,B,L,E) by depolymerisation from a } (B, b, l, e)) \\
&= \frac{b}{l} \cdot \mathbb{P}\left((l_1 = L, b_1 = B, e_1 = E) \quad \text{or} \quad (l_2 = l - 1 - l_1 = L, b_2 = b - 1 - b_1 = B, e_2 = e - e_1 = E)\right) \\
&= \frac{b}{l} \cdot \mathbb{P}\left((l_1 = L, b_1 = B, e_1 = E) \quad \text{or} \quad (l_1 = l - 1 - L, b_1 = b - 1 - B, e_1 = e - E)\right) \\
&= \begin{cases} \frac{b}{l} \cdot \mathbb{P}\left((l_1 = L, b_1 = B, e_1 = E)\right) + \mathbb{P}\left((l_1 = l - 1 - L, b_1 = b - 1 - B, e_1 = e - E)\right) & \text{if } 2L \neq l - 1, \\ \frac{b}{l} \cdot \mathbb{P}\left((l_1 = L, b_1 = B, e_1 = E)\right) & \text{if } 2L = l - 1, \end{cases}
\end{aligned}
$$

with:

$$\mathbb{P}(l_1 = L, b_1 = B, e_1 = E) = \mathbb{P}(e_1 = E \mid l_1 = L, b_1 = B)\, \mathbb{P}(b_1 = B \mid l_1 = L)\, \mathbb{P}(l_1 = L),$$

and:

$$\mathbb{P}(l_1 = L) = \begin{cases} \frac{1}{l} & \text{if } L \in \{0, \cdots, l-1\}, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbb{P}(b_1 = B \mid l_1 = L) = \begin{cases} \frac{1}{\min(b-1, L) - \max(0, b-l+L) + 1} & \text{if } B \in \{\max(0, b-l+L), \cdots, \min(b-1, L)\}, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbb{P}(e_1 = E \mid l_1 = L, b_1 = B) = \begin{cases} \frac{1}{\min(e, B+1) - \max(0, e-b+B) + 1} & \text{if } E \in \{\max(0, e-b+B), \cdots, \min(e, B+1)\}, \\ 0 & \text{otherwise.} \end{cases}$$

- $\mathbb{P}$ (create a (B, B, L, E) by releasing an Arp2/3 from a (B, b, l, e))

$$
\begin{aligned}
&= \mathbb{P}\left((l_1 = L, b_1 = B, e_1 = E) \quad \text{or} \quad (l_1 = l - L, b_1 = b - 1 - B, e_1 = e - E)\right) \\
&= \begin{cases} \mathbb{P}\left((l_1 = L, b_1 = B, e_1 = E)\right) + \mathbb{P}\left((l_1 = l - L, b_1 = b - 1 - B, e_1 = e - E)\right) & \text{if } 2L \neq l, \\ \mathbb{P}\left((l_1 = L, b_1 = B, e_1 = E)\right) & \text{if } 2L = l, \end{cases}
\end{aligned}
$$

with:

$$\mathbb{P}(l_1 = L, b_1 = B, e_1 = E) = \mathbb{P}(e_1 = E \mid l_1 = L, b_1 = B)\, \mathbb{P}(b_1 = B \mid l_1 = L)\, \mathbb{P}(l_1 = L),$$

and:

$$\mathbb{P}(l_1 = L) = \begin{cases} \frac{1}{l+1} & \text{if } L \in \{0, \cdots, l\}, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbb{P}(b_1 = B \mid l_1 = L) = \begin{cases} \frac{1}{\min(b-1, L) - \max(0, b-l+L-1) + 1} & \text{if } B \in \{\max(0, b-l+L-1), \cdots, \min(b-1, L)\}, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbb{P}(e_1 = E \mid l_1 = L, b_1 = B) = \begin{cases} \frac{1}{\min(e, B+1) - \max(0, e-b+B) + 1} & \text{if } E \in \{\max(0, e-b+B), \cdots, \min(e, B+1)\}, \\ 0 & \text{otherwise.} \end{cases}$$

$\circledast$ $\mathbb{P}$ (create a (B, B, L, E) by fragmentation of a (B, b, l, e))

$$= \mathbb{P}\left((l_1 = L, b_1 = B, e_1 = E) \text{ or } \left(l_{n_p} = L, b_{n_p} = B, e_{n_p} = E\right)\right)$$

$$= \sum_{n=1}^{n_{max}} \mathbb{P}\left(n_p = n\right) \mathbb{P}\Bigg(\left(\left\lfloor \frac{l'}{n} \right\rfloor = L, \left\lfloor \frac{e-n}{n} \right\rfloor = B, \left\lfloor \frac{e-n}{n} \right\rfloor + 1 = E\right)$$

$$\text{or } \left(l' - (n-1)\left\lfloor \frac{l'}{n} \right\rfloor = L, e - n - (n-1)\left\lfloor \frac{e-n}{n} \right\rfloor = B, e - n - (n-1)\left\lfloor \frac{e-n}{n} \right\rfloor + 1 = E\right) \Bigg| \, n_p = n\Bigg)$$

with:

$$n_{max} = \min\left(e, \left\lfloor \frac{l - \max(1, b + 1 - e)}{3} \right\rfloor\right),$$

and

$$\mathbb{P}\left(n_p = n\right) = \frac{1}{n_{max}}.$$

We have:

$$\mathbb{P}\Bigg(\left(\left\lfloor \frac{l'}{n} \right\rfloor = L, \left\lfloor \frac{e-n}{n} \right\rfloor = B, \left\lfloor \frac{e-n}{n} \right\rfloor + 1 = E\right)$$

$$\text{or } \left(l' - (n-1)\left\lfloor \frac{l'}{n} \right\rfloor = L, e - n - (n-1)\left\lfloor \frac{e-n}{n} \right\rfloor = B, e - n - (n-1)\left\lfloor \frac{e-n}{n} \right\rfloor + 1 = E\right) \Bigg| \, n_p = n\Bigg)$$

$$= \mathbb{P}\left(\left\lfloor \frac{l'}{n} \right\rfloor = L, \left\lfloor \frac{e-n}{n} \right\rfloor = B, \left\lfloor \frac{e-n}{n} \right\rfloor + 1 = E \,\middle|\, n_p = n\right)$$

$$+ \mathbb{P}\left(l' - (n-1)\left\lfloor \frac{l'}{n} \right\rfloor = L, e - n - (n-1)\left\lfloor \frac{e-n}{n} \right\rfloor = B, e - n - (n-1)\left\lfloor \frac{e-n}{n} \right\rfloor + 1 = E \,\middle|\, n_p = n\right)$$

$$- \mathbb{P}\Bigg(\left\lfloor \frac{l'}{n} \right\rfloor = l' - (n-1)\left\lfloor \frac{l'}{n} \right\rfloor = L, \left\lfloor \frac{e-n}{n} \right\rfloor = e - n - (n-1)\left\lfloor \frac{e-n}{n} \right\rfloor = B,$$

$$\left\lfloor \frac{e-n}{n} \right\rfloor + 1 = e - n - (n-1)\left\lfloor \frac{e-n}{n} \right\rfloor + 1 = E \,\Bigg|\, n_p = n\Bigg),$$

with:

$$\mathbb{P}\left(\left\lfloor \frac{l'}{n} \right\rfloor = L, \left\lfloor \frac{e-n}{n} \right\rfloor = B, \left\lfloor \frac{e-n}{n} \right\rfloor + 1 = E \,\middle|\, n_p = n\right)$$

$$= \mathbb{P}\left(\left\lfloor \frac{l'}{n} \right\rfloor = L \,\middle|\, n_p = n\right) \mathbb{P}\left(\left\lfloor \frac{e-n}{n} \right\rfloor = B, \left\lfloor \frac{e-n}{n} \right\rfloor + 1 = E \,\middle|\, n_p = n\right)$$

$$= \mathbb{P}\left(nL \le l' < n(L+1) \mid n_p = n\right) \mathbb{P}\left(\left\lfloor \frac{e-n}{n} \right\rfloor = B \,\middle|\, n_p = n\right) \mathbb{P}\left(B + 1 = E \,\middle|\, n_p = n, \left\lfloor \frac{e-n}{n} \right\rfloor = B\right),$$

$$= \mathbb{P}\left(nL - \max(3n, e-n) \le g_B < n(L+1) - \max(3n, e-n) \mid n_p = n\right)$$

$$\mathbb{P}\left(nB \le e - n < n(B+1) \mid n_p = n\right) \mathbb{P}\left(B + 1 = E \,\middle|\, n_p = n, \left\lfloor \frac{e-n}{n} \right\rfloor = B\right),$$

$$= \sum_{k=nL-\max(3n,e-n)}^{n(L+1)-\max(3n,e-n)-1} \mathbb{P}\left(g_B = k \mid n_p = n\right) \mathbb{1}_{\{nB \le e-n < n(B+1)\}} \mathbb{1}_{\{B+1=E\}},$$

where:

$$g_{Bmax} = l - \max(1, b + 1 - e) - \max(3n, e - n),$$

$$\mathbb{P}\left(g_B = k \mid n_p = n\right) = \begin{cases} p_B(1-p_B)^{k-1} & \text{if } k \in \{1, \cdots, g_{Bmax}\}, \\ (1-p_B)^{g_{Bmax}} & \text{if } k = 0, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$p_B = 1 - 1 - \frac{\lambda^- N}{e\lambda^+ \star K_t^N(M,0,1,0)}.$$

We also have:

$$\begin{aligned}
&\mathbb{P}\left(l' - (n-1)\left\lfloor\frac{l'}{n}\right\rfloor = L, e - n - (n-1)\left\lfloor\frac{e-n}{n}\right\rfloor = B, e - n - (n-1)\left\lfloor\frac{e-n}{n}\right\rfloor + 1 = E \,\middle|\, n_p = n\right)\\
&= \mathbb{P}\left(l' - (n-1)\left\lfloor\frac{l'}{n}\right\rfloor = L \,\middle|\, n_p = n\right)\mathbb{P}\left(e - n - (n-1)\left\lfloor\frac{e-n}{n}\right\rfloor = B \,\middle|\, n_p = n\right)\\
&\qquad\qquad \mathbb{P}\left(B + 1 = E \,\middle|\, n_p = n, e - n - (n-1)\left\lfloor\frac{e-n}{n}\right\rfloor = B\right)\\
&= \sum_{k=0}^{g_{Bmax}} \mathbb{P}\left(g_B = k \mid n_p = n\right) \mathbb{1}_{\left\{k+\max(3n,e-n)-(n-1)\left\lfloor\frac{k+\max(3n,e-n)}{n}\right\rfloor = L\right\}} \mathbb{1}_{\left\{e-n-(n-1)\left\lfloor\frac{e-n}{n}\right\rfloor = B\right\}} \mathbb{1}_{\{B+1=E\}},
\end{aligned}$$

and

$$\begin{aligned}
&\mathbb{P}\Bigg(\left\lfloor\frac{l'}{n}\right\rfloor = l' - (n-1)\left\lfloor\frac{l'}{n}\right\rfloor = L, \left\lfloor\frac{e-n}{n}\right\rfloor = e - n - (n-1)\left\lfloor\frac{e-n}{n}\right\rfloor = B,\\
&\qquad\qquad \left\lfloor\frac{e-n}{n}\right\rfloor + 1 = e - n - (n-1)\left\lfloor\frac{e-n}{n}\right\rfloor + 1 = E \,\middle|\, n_p = n\Bigg),\\
&= \mathbb{P}\left(l' = nL, e - n = nB, B + 1 = E \mid n_p = n\right),\\
&= \mathbb{P}\left(l' = nL \mid n_p = n\right)\mathbb{P}\left(e - n = nB \mid n_p = n\right)\mathbb{1}_{\{B+1=E\}},\\
&= \mathbb{P}\left(g_B = nL - \max(3n, e-n) \mid n_p = n\right)\mathbb{1}_{\{e-n=nB\}}\mathbb{1}_{\{B+1=E\}}.
\end{aligned}$$

- $\mathbb{P}$ (create a $(C,0,L,0)$ by depolymerisation with split of a (B,b,l,e))

  $= \mathbb{P}$ (create a (B,B,L,E) by depolymerisation with split of a (B,b,l,e)) with $B = 0, E = 0$.

- $\mathbb{P}$ (create a $(C,0,L,0)$ by releasing an Arp2/3 from a (B,b,l,e))

  $= \mathbb{P}$ (create a (B,B,L,E) by releasing an Arp2/3 from a (B,b,l,e)) with $B = 0, E = 0$.

- $\mathbb{P}$ (create a $(F,0,L,1)$ by fragmentation of a $(F,0,l,1)$)

$$= \mathbb{P}\left(l' = g_F + 3 = L\right) = \mathbb{P}\left(g_F = L - 3\right),$$

with:

$$\mathbb{P}\left(g_F = k\right) = \begin{cases} p_F\left(1-p_F\right)^{k-1} & \text{if } k \in \{1,\cdots,l-4\},\\ \left(1-p_F\right)^{l-4} & \text{if } k = 0,\\ 0 & \text{otherwise,}\end{cases}$$

and

$$p_F = 1 - \frac{\lambda^- N}{\lambda_F^+ \star K_t^N(P,0,1,0)}.$$

- $\mathbb{P}$ (create a $(S,0,L,1)$ by depolymerisation with split of a (B,b,l,e))

  $= \mathbb{P}$ (create a (B,B,L,E) by depolymerisation with split of a (B,b,l,e)) with $B = 0, E = 1$.

- $\mathbb{P}$ (create a $(S,0,L,1)$ by releasing an Arp2/3 from a (B,b,l,e))

  $= \mathbb{P}$ (create a (B,B,L,E) by releasing an Arp2/3 from a (B,b,l,e)) with $B = 0, E = 1$.

- $\mathbb{P}$ (create a $(S, 0, L, 1)$ by fragmentation of a $(S, 0, l, 1)$)

$$= \mathbb{P}\left(l' = g_S + 3 = L\right) = \mathbb{P}\left(g_S = L - 3\right),$$

with:

$$\mathbb{P}\left(g_S = k\right) = \begin{cases} p_S\left(1 - p_S\right)^{k-1} & \text{if } k \in \{1, \cdots, l-4\}, \\ \left(1 - p_S\right)^{l-4} & \text{if } k = 0, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$p_S = 1 - \frac{\lambda^- N}{\lambda^+ \star K_t^N(M, 0, 1, 0)}.$$

- $\mathbb{P}$ (create a $(S, 0, L, 1)$ by fragmentation of a (B, b, l, e))

$$= \mathbb{P}\text{ (create a } (B, B, L, E) \text{ by fragmentation of a } (B, b, l, e)) \quad \text{with } B = 0, E = 1.$$

- $\sum_m (m-1)\mathbb{P}$ (release m monomers by depolymerisation with split of a (B, b, l, e))

$$\begin{aligned}
&= \mathbb{P}\left(l_1 \geq 3, l_2 = 1\right) + 2\mathbb{P}\left(l_1 \geq 3, l_2 = 2\right) + \mathbb{P}\left(l_1 = 1, l_2 \leq 3\right) + 2\mathbb{P}\left(l_1 = 2, l_2 \leq 3\right) \\
&+ \mathbb{P}\left(l_1 = 0, l_2 = 1\right) + 2\mathbb{P}\left(l_1 = 0, l_2 = 2\right) + \mathbb{P}\left(l_1 = 1, l_2 = 0\right) + 2\mathbb{P}\left(l_1 = 1, l_2 = 1\right) + 3\mathbb{P}\left(l_1 = 1, l_2 = 2\right) \\
&+ 2\mathbb{P}\left(l_1 = 2, l_2 = 0\right) + 3\mathbb{P}\left(l_1 = 2, l_2 = 1\right) + 4\mathbb{P}\left(l_1 = 2, l_2 = 2\right) \\
&= \mathbb{P}\left(l_1 = l - 2, l - 2 \geq 3\right) + 2\mathbb{P}\left(l_1 = l - 3, l - 3 \geq 3\right) + \mathbb{P}\left(l_1 = 1, l - 2 \leq 3\right) + 2\mathbb{P}\left(l_1 = 2, l - 3 \leq 3\right) \\
&+ \mathbb{P}\left(l_1 = 0, l - 1 = 1\right) + 2\mathbb{P}\left(l_1 = 0, l - 1 = 2\right) \\
&+ \mathbb{P}\left(l_1 = 1, l - 2 = 0\right) + 2\mathbb{P}\left(l_1 = 1, l - 2 = 1\right) + 3\mathbb{P}\left(l_1 = 1, l - 2 = 2\right) \\
&+ 2\mathbb{P}\left(l_1 = 2, l - 3 = 0\right) + 3\mathbb{P}\left(l_1 = 2, l - 3 = 1\right) + 4\mathbb{P}\left(l_1 = 2, l - 3 = 2\right) \\
&= \frac{2}{l}\mathbb{1}_{\{l \geq 5\}} + \frac{4}{l}\mathbb{1}_{\{l \geq 6\}} + \frac{6}{l}\mathbb{1}_{\{l=3\}} + \frac{6}{l}\mathbb{1}_{\{l=4\}} + \frac{4}{l}\mathbb{1}_{\{l=5\}}
\end{aligned}$$

- $\sum_m m\mathbb{P}$ (release m monomers by releasing an Arp2/3 from a (B, b, l, e))

$$\begin{aligned}
&= \mathbb{P}\left(l_1 \geq 3, l_2 = 1\right) + 2\mathbb{P}\left(l_1 \geq 3, l_2 = 2\right) \\
&+ \mathbb{P}\left(l_1 = 1, l_2 \leq 3\right) + 2\mathbb{P}\left(l_1 = 2, l_2 \leq 3\right) \\
&+ \mathbb{P}\left(l_1 = 0, l_2 = 1\right) + 2\mathbb{P}\left(l_1 = 0, l_2 = 2\right) + \mathbb{P}\left(l_1 = 1, l_2 = 0\right) + 2\mathbb{P}\left(l_1 = 1, l_2 = 1\right) + 3\mathbb{P}\left(l_1 = 1, l_2 = 2\right) \\
&+ 2\mathbb{P}\left(l_1 = 2, l_2 = 0\right) + 3\mathbb{P}\left(l_1 = 2, l_2 = 1\right) + 4\mathbb{P}\left(l_1 = 2, l_2 = 2\right) \\
&= \mathbb{P}\left(l_1 = l - 1, l - 1 \geq 3\right) + 2\mathbb{P}\left(l_1 = l - 2, l - 2 \geq 3\right) + \mathbb{P}\left(l_1 = 1, l - 1 \leq 3\right) + 2\mathbb{P}\left(l_1 = 2, l - 2 \leq 3\right) \\
&+ \mathbb{P}\left(l_1 = 0, l = 1\right) + 2\mathbb{P}\left(l_1 = 0, l = 2\right) \\
&+ \mathbb{P}\left(l_1 = 1, l - 1 = 0\right) + 2\mathbb{P}\left(l_1 = 1, l - 1 = 1\right) + 3\mathbb{P}\left(l_1 = 1, l - 1 = 2\right) \\
&+ 2\mathbb{P}\left(l_1 = 2, l - 2 = 0\right) + 3\mathbb{P}\left(l_1 = 2, l - 2 = 1\right) + 4\mathbb{P}\left(l_1 = 2, l - 2 = 2\right) \\
&= \frac{2}{l+1}\mathbb{1}_{\{l \geq 4\}} + \frac{4}{l+1}\mathbb{1}_{\{l \geq 5\}} + \frac{6}{l+1}\mathbb{1}_{\{l=3\}} + \frac{4}{l+1}\mathbb{1}_{\{l=4\}}
\end{aligned}$$

- $\sum_m m\mathbb{P}$ (release m monomers by fragmentation of a $(S, 0, l, 1)$)

$$= \sum_{m=1}^{l-3} m\mathbb{P}\left(l' = g_S + 3 = l - m\right) = \sum_{m=1}^{l-3} m\mathbb{P}\left(g_S = l - m - 3\right).$$

- $\sum_m m\mathbb{P}$ (release m monomers by fragmentation of a $(F, 0, l, 1)$)

$$= \sum_{m=1}^{l-3} m\mathbb{P}\left(l' = g_F + 3 = l - m\right) = \sum_{m=1}^{l-3} m\mathbb{P}\left(g_F = l - m - 3\right).$$

⁕ $\sum_m m\mathbb{P}$ (release m free monomers by fragmentation of a (B, b, l, e))

$$= \sum_{n=1}^{n_{max}} \mathbb{P}\left(n_p = n\right) \sum_{m=\max(1,b+1-e)}^{l-\max(3n,e-n)} m\mathbb{P}\left(l' = g_B + \max(3n, e-n) = l - m \mid n_p = n\right)$$

$$= \sum_{n=1}^{n_{max}} \sum_{m=\max(1,b+1-e)}^{l-\max(3n,e-n)} \frac{m}{n_{max}} \mathbb{P}\left(g_B = l - m - \max(3n, e-n) \mid n_p = n\right).$$

This deterministic system would allow us to eliminate the variability of the results obtained with the stochastic model. This would then allow us to improve the results of a qualitative study of behaviour.

This system of differential equations has been implemented, but unfortunately we were unable to exploit the code. Indeed, by solving the problem of complexity in space, we created a problem of complexity in time. The execution of this code is extremely slow. On the other hand, this code does not allow us to retrieve the results verified in Chapter 1, so there must be an error or a rounding error problem.

Appendix A

C++ Code for the Simulations in Chapter I

Contents

1 - Model for a Single Linear Actin Polymer

Fonction C++ prenant en arguments :

- int L0init : le nombre initial de monomères libres
- int L1init : la taille initiale du filament (en monomère)
- int L2init : le nombre initial de complexes actin-G/profilin
- int N : le nombre total de monomères
- double tmax : le temps final
- double lambda_plus : le taux d'élongation spontanée
- double lambda_moins : le taux de dépolymérisation
- double lambdaF_plus : le taux d'élongation avec une formine
- double phiP : le taux de création de complexes actin-G/profilin
- double phiF_plus : le taux de fixation d'une formine sur un polymère
- double phiF_moins : le taux de libération d'une formine

créant et remplissant un fichier contenant le vecteur

$$(L_0(t), L_1(t), L_2(t), M(t), t)$$

pour chaque temps $t \in [0, t_{max}]$.

```
void ModeleSingleFil(int L0init, int L1init, int L2init, int N,
double tmax, double lambda_plus, double lambda_moins, double lambdaF_plus,
double phiP, double phiF_plus, double phiF_moins) {
```

```
4     FILE* outparametre = fopen("Parameters.txt","w");
5
6     fprintf(outparametre,"N = %i", N);
7     fprintf(outparametre, "\n");
8     fprintf(outparametre,"lambda_plus = %f", lambda_plus);
9     fprintf(outparametre, "\n");
10    fprintf(outparametre,"lambda_moins = %f", lambda_moins);
11    fprintf(outparametre, "\n");
12    fprintf(outparametre,"lambdaF_plus = %f", lambdaF_plus);
13    fprintf(outparametre, "\n");
14    fprintf(outparametre,"phiP = %f", phiP);
15    fprintf(outparametre, "\n");
16    fprintf(outparametre,"phiF_plus = %f", phiF_plus);
17    fprintf(outparametre, "\n");
18    fprintf(outparametre,"phiF_moins = %f", phiF_moins);
19
20    fclose(outparametre);
21
22    FILE* output = fopen("Data.txt","w");
23
24    int L0_0 = L0init;
25    fprintf(output, "%i ", L0_0);
26    int L1_0 = L1init;
27    fprintf(output, "%i ", L1_0);
28    int L2_0 = L2init;
29    fprintf(output, "%i ", L2_0);
30    int M_0 = 0;  //Initialiser à  M_0=1 pour commencer dans le mode rapide
31    fprintf(output, "%i ", M_0);
32    double T_0 = 0;
33    fprintf(output, "%f ", T_0);
34    fprintf(output, "\n");
35
36    int L0_1;
37    int L1_1;
38    int L2_1;
39    int M_1;
40    double T_1;
41
42    double taux_elongation_simple;
43    double taux_elongation_formine;
44    double taux_depolymerisation;
45    double taux_creation_complexe;
46    double taux_fixation_formine;
47    double taux_liberation_formine;
48    double taux_total;
49
50    int IfContinue = 1;
51    int des4faces;
52    int des3faces;
53    double p;
54    bool des_pipe;
55
56    L0_1 = L0_0;
57    L1_1 = L1_0;
58    L2_1 = L2_0;
59    M_1 = M_0;
60
61    while (T_0 < tmax && IfContinue == 1 ) {
62        if(L1_0 > 0) {
63            if(M_0 == 0) {
64                taux_elongation_simple = lambda_plus*double(L0_0)/N;
```

```
65                taux_depolymerisation = lambda_moins;
66                taux_creation_complexe = phiP*double(L0_0)/N;
67                taux_fixation_formine = phiF_plus;
68                taux_total = taux_elongation_simple+taux_depolymerisation
69                +taux_creation_complexe+taux_fixation_formine;
70
71                if (taux_total == 0) {
72                    IfContinue = 0; }
73                else {
74                    while(L0_1 == L0_0 && L1_1 == L1_0 && L2_1 == L2_0
75                    && M_1 == M_0) {
76                        des4faces = floor(4*rand01()+1);
77
78                        if(des4faces == 1) {
79                            p = taux_elongation_simple/taux_total;
80                            des_pipe = (rand01() <= p);
81                            if(des_pipe == 1) {
82                                L0_1 = L0_0-1;
83                                L1_1 = L1_0+1; } }
84                        else if(des4faces == 2) {
85                            p = taux_depolymerisation/taux_total;
86                            des_pipe = (rand01() <= p);
87                            if(des_pipe == 1) {
88                                L0_1 = L0_0+1;
89                                L1_1 = L1_0-1; } }
90                        else if(des4faces == 3) {
91                            p = taux_creation_complexe/taux_total;
92                            des_pipe = (rand01() <= p);
93                            if(des_pipe == 1) {
94                                L0_1 = L0_0-1;
95                                L2_1 = L2_0+1; } }
96                        else {
97                            p = taux_fixation_formine/taux_total;
98                            des_pipe = (rand01() <= p);
99                            if(des_pipe == 1) {
100                               M_1 = M_0+1; } } }
101
102                   T_1 = T_0+rand_exp(taux_total);
103
104                   fprintf(output, "%i ", L0_1);
105                   fprintf(output, "%i ", L1_1);
106                   fprintf(output, "%i ", L2_1);
107                   fprintf(output, "%i ", M_1);
108                   fprintf(output, "%f ", T_1);
109                   fprintf(output, "\n");
110
111                   L0_0 = L0_1;
112                   L1_0 = L1_1;
113                   L2_0 = L2_1;
114                   M_0 = M_1;
115                   T_0 = T_1; } }
116           else {
117               taux_elongation_formine = lambdaF_plus*double(L2_0)/N;
118               taux_depolymerisation = lambda_moins;
119               taux_creation_complexe = phiP*double(L0_0)/N;
120               taux_liberation_formine = phiF_moins;
121               taux_total = taux_elongation_formine+taux_depolymerisation
122               +taux_creation_complexe+taux_liberation_formine;
123
124               if (taux_total == 0) {
125                   IfContinue = 0; }
```

```
126                 else {
127                     while(L0_1 == L0_0 && L1_1 == L1_0 && L2_1 == L2_0
128                     && M_1 == M_0) {
129                         des4faces=floor(4*rand01()+1);
130 
131                         if (des4faces == 1) {
132                             p = taux_elongation_formine/taux_total;
133                             des_pipe = (rand01() <= p);
134                             if (des_pipe == 1) {
135                                 L1_1 = L1_0+1;
136                                 L2_1 = L2_0-1; } }
137                         else if (des4faces == 2) {
138                             p = taux_depolymerisation/taux_total;
139                             des_pipe = (rand01() <= p);
140                             if (des_pipe == 1) {
141                                 L0_1 = L0_0+1;
142                                 L1_1 = L1_0-1; } }
143                         else if (des4faces == 3) {
144                             p = taux_creation_complexe/taux_total;
145                             des_pipe = (rand01() <= p);
146                             if(des_pipe == 1) {
147                                 L0_1 = L0_0-1;
148                                 L2_1 = L2_0+1; } }
149                         else {
150                             p = taux_liberation_formine/taux_total;
151                             des_pipe = (rand01() <= p);
152                             if(des_pipe == 1) {
153                                 M_1 = M_0-1; } } }
154                     T_1 = T_0+rand_exp(taux_total);
155 
156                     fprintf(output, "%i ", L0_1);
157                     fprintf(output, "%i ", L1_1);
158                     fprintf(output, "%i ", L2_1);
159                     fprintf(output, "%i ", M_1);
160                     fprintf(output, "%f ", T_1);
161                     fprintf(output, "\n");
162 
163                     L0_0 = L0_1;
164                     L1_0 = L1_1;
165                     L2_0 = L2_1;
166                     M_0 = M_1;
167                     T_0 = T_1; } } }
168         else {
169             taux_creation_complexe = phiP*double(L0_0)/N;
170             taux_total = taux_creation_complexe;
171 
172             if (taux_total == 0) {
173                 IfContinue = 0; }
174             else {
175                 L0_1 = L0_0-1;
176                 L2_1 = L2_0+1;
177 
178                 T_1 = T_0+rand_exp(taux_total);
179 
180                 fprintf(output, "%i ", L0_1);
181                 fprintf(output, "%i ", L1_1);
182                 fprintf(output, "%i ", L2_1);
183                 fprintf(output, "%i ", M_1);
184                 fprintf(output, "%f ", T_1);
185                 fprintf(output, "\n");
186 
```

```
                L0_0 = L0_1;
                L1_0 = L1_1;
                L2_0 = L2_1;
                M_0 = M_1;
                T_0 = T_1; } } }
  fclose(output); }
```

2 - Model for a Population of Linear Actin Polymers

Fonction C++ prenant en arguments :

- std::string type_filament : le(s) type(s) de filaments à l'état inital parmi :
  - "simple" : 100 % de filaments simples (sans formine)
  - "formine" : 100 % de filaments associés à une formine
  - "both" : 50% de filaments simples et 50% de filaments associés à une formine
- int lmax : la longueur maximale des filaments
- double* K0 : un tableau (vide) de réels de taille 2*lmax+2
  (à initialiser dans main.cpp : double K0[2*lmax+2];)

demandant via la console :

- la taille des filaments initiaux
- la proportion de monomères libres à l'état initial
- la proportion de complexes actin-G/profilin à l'état initial

et modifiant le tableau K0 afin d'avoir l'état initial souhaité.

```
void GeneInit(std::string type_filament, int lmax, double* K0) {

  for(int j=0; j<2*lmax+2; j++) {
    K0[j]=0.0; }

  printf("Quelle est la taille des filaments initiaux (< %i) ? \n", lmax);

  int l0;
  scanf("%d",&l0);

  printf("Quelle est la proportion de monomères à l'état initial ? \n");

  double prop0;
  scanf("%lf",&prop0);

  printf("Quelle est la proportion de complexes actine/profiline à l'état
          initial ? \n");

  double prop1;
  scanf("%lf",&prop1);
```

```
21
22   K0[0]=prop0;
23   K0[1]=prop1;
24
25   if(type_filament=="simple") {
26     K0[l0+1]=(1-K0[0]-K0[1])/double(l0); }
27     //Départ avec prop0 monomères libres, prop1 complexes et le reste de
28     //filaments SIMPLES de taille l0
29   else if(type_filament=="formine") {
30     K0[l0+1+lmax]=(1-K0[0]-K0[1])/double(l0); }
31     //Départ avec prop0 monomères libres, prop1 complexes et le reste de
32     //filaments FORMINES de taille l0
33   else if(type_filament=="both") {
34     K0[l0+1]=(1-K0[0]-K0[1])/(2*double(l0));
35     K0[l0+lmax+1]=(1-K0[0]-K0[1])/(2*double(l0)); }
36     //Départ avec prop0 monomères libres, prop1 complexes et le
37     //reste = 50% filaments SIMPLES + 50% filaments FORMINES de taille l0
38   else {
39     printf("Le type choisi doit appartenir à la liste suivante \n");
40     printf(" * simple = départ avec des filaments simples uniquement \n");
41     printf(" * formine = départ avec des filaments associés à une formine
42              uniquement \n");
43     printf(" * both = départ avec 50% de filaments simples
44              et 50% de filaments associés à une formine \n"); } }
```

Fonction C++ prenant en arguments :

- double tmax : le temps final
- double dt : le pas de temps pour la méthode d'Euler explicite
- double lambda_plus : le taux d'élongation spontanée
- double lambda_moins : le taux de dépolymérisation
- double lambdaF_plus : le taux d'élongation avec une formine
- double phiP : le taux de création de complexes actin-G/profilin
- double phiF_plus : le taux de fixation d'une formine sur un polymère
- double phiF_moins : le taux de libération d'une formine
- int lmax : la longueur maximale des filaments
- double* K0 : le tableau de réels de taille 2*lmax+2 modifié par la fonction GeneInit

créant et remplissant un fichier contenant le vecteur

$$(t, k_t(M,1), k_t(P,1), k_t(S,1), \ldots, k_t(S,lmax), k_t(F,1), \ldots, k_t(F,lmax))$$

pour chaque temps $t \in [0, t_{max}]$.

```
1 void ModelePopFil (double tmax, double dt, double lambda_plus,
2 double lambda_moins, double lambdaF_plus, double PhiP, double PhiF_plus,
3 double PhiF_moins, int lmax, double* K0 ) {
4
5   FILE* outparametre = fopen("Parameters_pop.txt","w");
6   fprintf(outparametre,"dt = %f", dt);
7   fprintf(outparametre, "\n");
8   fprintf(outparametre,"lambda_plus = %f", lambda_plus);
```

```
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lambda_moins = %f", lambda_moins);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lambdaF_plus = %f", lambdaF_plus);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiP = %f", PhiP);
  fprintf(outparametre, "\n");
  fprintf(outparametre, "PhiF_plus = %f", PhiF_plus);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiF_moins = %f", PhiF_moins);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lmax = %i", lmax);
  fprintf(outparametre, "\n");
  fclose(outparametre);

  FILE* output = fopen("Data_pop.txt","w");

  double t=0.0;
  int cpt=0;
  int next_print=1;

  fprintf(output, "%.5e ", t);
  for (int j=0; j<2*lmax+2; j++) {
    fprintf(output, "%.5e ", K0[j]); }
  fprintf(output, "\n");

  double K1[2*lmax+2];

  while(t<tmax) {
    //Pour que la somme pondérée soit toujours égale à 1,
    //et donc éviter les erreurs
    //numérique, on normalise à chaque pas de temps.
    double sp=0.0;
    //On approche la solution du système au temps dt par une méthode
    //d'Euler explicite

    double NbPolySimple=0.0;
    for(int j=2; j<lmax+2; j++) {
      NbPolySimple=NbPolySimple+K0[j]; }

    double NbPolyFormin=0.0;
    for(int j=lmax+2; j<2*lmax+2; j++) {
      NbPolyFormin=NbPolyFormin+K0[j]; }
    K1[0]=dt*(lambda_moins*(NbPolySimple+NbPolyFormin)
          -lambda_plus*K0[0]*NbPolySimple-PhiP*K0[0])+K0[0];
    K1[1]=dt*(PhiP*K0[0]-lambdaF_plus*K0[1]*NbPolyFormin)+K0[1];
    K1[2]=dt*(lambda_moins*K0[3]+PhiF_moins*K0[lmax+2]
          -lambda_plus*K0[0]*K0[2]-lambda_moins*K0[2]-PhiF_plus*K0[2])+K0[2];
    for(int j=3; j<lmax+1; j++) {
      K1[j]=dt*(lambda_plus*K0[0]*K0[j-1]+lambda_moins*K0[j+1]
            +PhiF_moins*K0[j+lmax]-lambda_plus*K0[0]*K0[j]
            -lambda_moins*K0[j]-PhiF_plus*K0[j])+K0[j]; }
    K1[lmax+1]=dt*(lambda_plus*K0[0]*K0[lmax]+PhiF_moins*K0[2*lmax+1]
                -lambda_plus*K0[0]*K0[lmax+1]-lambda_moins*K0[lmax+1]
                -PhiF_plus*K0[lmax+1])+K0[lmax+1];
    K1[lmax+2]=dt*(lambda_moins*K0[lmax+3]+PhiF_plus*K0[2]
                -lambdaF_plus*K0[1]*K0[lmax+2]-lambda_moins*K0[lmax+2]
                -PhiF_moins*K0[lmax+2])+K0[lmax+2];
    for(int j=lmax+3; j<2*lmax+1; j++) {
      K1[j]=dt*(lambdaF_plus*K0[1]*K0[j-1]+lambda_moins*K0[j+1]
            +PhiF_plus*K0[j-lmax]-lambdaF_plus*K0[1]*K0[j]
```

```
70              -lambda_moins*K0[j]-PhiF_moins*K0[j])+K0[j]; }
71     K1[2*lmax+1]=dt*(lambdaF_plus*K0[1]*K0[2*lmax]+PhiF_plus*K0[lmax+1]
72                   -lambdaF_plus*K0[1]*K0[2*lmax+1]-lambda_moins*K0[2*lmax+1]
73                   -PhiF_moins*K0[2*lmax+1])+K0[2*lmax+1];
74
75     sp=sp+K1[0]+K1[1];
76     for(int j=2; j<lmax+2; j++) {
77       sp=sp+K1[j]*(j-1); }
78     for(int j=lmax+2; j<2*lmax+2; j++) {
79       sp=sp+K1[j]*(j-1-lmax); }
80
81     for(int j=0; j<2*lmax+2; j++) {
82       K0[j]=K1[j]/sp; }
83     t=t+dt;
84     cpt=cpt+1;
85
86     if (cpt<100000000) {
87     //On écrit le résultat dans le fichier seulement toutes les 1000
88     //itérations pour limiter le temps d'exécution
89
90       if (cpt==next_print) {
91    fprintf(output, "%.5e ", t);
92           for(int j=0; j<2*lmax+2; j++) {
93             fprintf(output, "%.5e ", K0[j]); }
94           fprintf(output,"\n");
95           next_print=next_print+1000; } }
96     else {
97       if (cpt==next_print) {
98           fprintf(output, "%.5e ", t);
99           for(int j=0; j<2*lmax+2; j++) {
100            fprintf(output, "%.5e ", K0[j]); }
101          fprintf(output,"\n");
102          next_print=next_print+1000000; } } }
103 fclose(output); }
```

Appendix B

R Code for the Simulations in Chapter II

Contents

1 - Trace and Colour Segments

Fonction R prenant en arguments :

- A : un vecteur contenant les coordonnées d'un point
- B : un vecteur contenant les coordonnées d'un point
- color : une chaine de caractère ou un entier définissant la couleur
- x : un vecteur définissant les limites de l'axe des abscisses
- y : un vecteur définissant les limites de l'axe des ordonnées

permettant de tracer le segment entre les points A et B dans la couleur donnée.

```
PlotSegPt=function(A,B,color,x,y) {
  plot(c(A[1],B[1]),c(A[2],B[2]),"o",xlim=x,ylim=y,col=color,
  ylab="",xlab="",xaxt="n",yaxt="n",cex=0.8,lwd=1.5,bg=color) }
```

Fonction R prenant en arguments :

- A : un vecteur contenant les coordonnées d'un point
- B : un vecteur contenant les coordonnées d'un point
- style : un vecteur dont la première coordonnée définie une couleur
- x : un vecteur définissant les limites de l'axe des abscisses
- y : un vecteur définissant les limites de l'axe des ordonnées

permettant de tracer le segment entre les points A et B dans la couleur donnée et avec une des extrémités cappés.

```
PlotSegPtCap=function(A,B,style,x,y) {
  plot(c(A[1],B[1]),c(A[2],B[2]),"o",xlim=x,ylim=y,col=style[1],
  ylab="",xlab="",xaxt="n",yaxt="n",cex=0.8,lwd=1.5,pch=style[2],
  bg=style[1],main="Tree representation") }
```

Fonction R prenant en arguments :

- S : une matrice 2x2 correspondant aux coordonnées de 2 points (en ligne)
- color : une chaine de caractère ou un entier définissant la couleur
- x : un vecteur définissant les limites de l'axe des abscisses
- y : un vecteur définissant les limites de l'axe des ordonnées

permettant de tracer le segment entre les deux points dans la couleur donnée.

```
PlotSeg=function(S,color,x,y) {
  PlotSegPt(S[1,],S[2,],color,x,y) }
```

Fonction R prenant en arguments :

- S : une matrice 2x2 correspondant aux coordonnées de 2 points (en ligne)
- style : un vecteur dont la première coordonnée définie une couleur
- x : un vecteur définissant les limites de l'axe des abscisses
- y : un vecteur définissant les limites de l'axe des ordonnées

permettant de tracer le segment entre les deux points dans la couleur donnée et avec une des extrémités cappés.

```
PlotSegCap=function(S,style,x,y) {
  PlotSegPtCap(S[1,],S[2,],style,x,y) }
```

Fonction R prenant en arguments :

* A : un vecteur contenant les coordonnées d'un point
* B : un vecteur contenant les coordonnées d'un point
* color : une chaine de caractère ou un entier définissant la couleur

permettant de tracer le segment entre les points A et B dans la couleur donnée sur un graphique existant.

```
LineSegPt=function(A,B,color) {
  lines(c(A[1],B[1]),c(A[2],B[2]),"o",col=color,cex=0.8,lwd=1.5,bg=color) }
```

Fonction R prenant en arguments :

* A : un vecteur contenant les coordonnées d'un point
* B : un vecteur contenant les coordonnées d'un point
* style : un vecteur dont la première coordonnée définie une couleur

permettant de tracer le segment entre les points A et B dans la couleur donnée et avec une des extrémités cappés sur un graphique existant.

```
LineSegPtCap=function(A,B,style) {
  lines(c(A[1],B[1]),c(A[2],B[2]),"o",col=style[1],cex=0.8,lwd=1.5,
  pch=style[2],bg=style[1]) }
```

Fonction R prenant en arguments :

* S : une matrice 2x2 correspondant aux coordonnées de 2 points (en ligne)
* color : une chaine de caractère ou un entier définissant la couleur

permettant de tracer le segment entre les deux points dans la couleur donnée sur un graphique existant.

```
LineSeg=function(S,color) {
  LineSegPt(S[1,],S[2,],color) }
```

Fonction R prenant en arguments :

* S : une matrice 2x2 correspondant aux coordonnées de 2 points (en ligne)
* style : un vecteur dont la première coordonnée définie une couleur

permettant de tracer le segment entre les deux points dans la couleur donnée et avec une des extrémités cappés sur un graphique existant.

```
LineSegCap=function(S,style) {
  LineSegPtCap(S[1,],S[2,],style) }
```

Fonction R prenant en arguments :

* L : une liste de matrice 2x2 correspondante aux coordonnées de 2 points
* color : une chaine de caractère ou un entier définissant la couleur

permettant de tracer les segments entre les paires de points définies par les matrices dans la couleur donnée.

```
PlotListeSeg=function(L,color) {
  maxX=L[[1]][1,1]
  maxY=L[[1]][1,2]
  minX=L[[1]][1,1]
  minY=L[[1]][1,2]
  for (i in 1:length(L)) {
    maxX=max(maxX,L[[i]][,1])
    maxY=max(maxY,L[[i]][,2])
    minX=min(minX,L[[i]][,1])
    minY=min(minY,L[[i]][,2]) }
  x=c(minX,maxX)
  y=c(minY,maxY)
  PlotSeg(L[[1]],color,x,y)
  if (length(L)>1) {
    for (i in 2:length(L)) {
      LineSeg(L[[i]],color) } } }
```

Fonction R prenant en arguments :

- L : une liste de matrice 2x2 correspondante aux coordonnées de 2 points

permettant de tracer les segments entre les paires de points définies par les matrices avec un choix de couleur automatique.

```
PlotListeSegColor=function(L) {
  color=c('red','darkorange1','goldenrod1','yellowgreen','green4',
  'darkturquoise', 'deepskyblue3','blue','purple3','darkorchid1','magenta')
  palette(color)
  maxX=L[[2]][1,1]
  maxY=L[[2]][1,2]
  minX=L[[2]][1,1]
  minY=L[[2]][1,2]
  for (i in 1:(length(L)/2))  {
    maxX=max(maxX,L[[2*i]][,1])
    maxY=max(maxY,L[[2*i]][,2])
    minX=min(minX,L[[2*i]][,1])
    minY=min(minY,L[[2*i]][,2]) }
  x=c(minX,maxX)
  y=c(minY,maxY)
  PlotSegCap(L[[2]],L[[1]],x,y)
  for (i in 1:(length(L)/2)) {
    LineSegCap(L[[2*i]],L[[2*i-1]]) } }
```

2 - Place Segments

Fonction R prenant en arguments :

- L : une liste de matrice 2x2 correspondante aux coordonnées de 2 points
- v : un vecteur de translation (de taille 2)

permettant de translater une liste de segments et renvoyant une liste de matrices 2x2 contenant les nouvelles coordonnées.

```
TranslateListeSeg=function(L,v) {
  L2=list()
  if(length(L)>0) {
    for (i in 1:length(L)) {
      L2[[i]]=matrix(c(L[[i]][,1]+v[1],L[[i]][,2]+v[2]),ncol=2) } }
  return(L2) }
```

Fonction R prenant en arguments :

- L : une liste de matrice 2x2 correspondante aux coordonnées de 2 points
- v : un vecteur de translation (de taille 2)

permettant de translater une liste de segments et renvoyant une liste de matrices 2x2 contenant les nouvelles coordonnées avec gestion automatique des couleurs.

```
TranslateListeSegColor=function(L,v) {
  L2=list()
  if(length(L)>0) {
    for (i in 1:(length(L)/2)) {
      L2[[2*i]]=matrix(c(L[[2*i]][,1]+v[1],L[[2*i]][,2]+v[2]),ncol=2)
      L2[[2*i-1]]=L[[2*i-1]] } }
  return(L2) }
```

3 - Trace and Colour a Tree

Pour coder un arbre, on utilise une liste de listes. Afin d'avoir toutes les informations nécessaires à notre modèle, un arbre correspond à une liste de longueur 5 contenant (dans l'ordre) :

1. un entier correspondant à la longueur de la racine
2. 0, 1 ou 2 : 0 si il s'agit d'un nœud interne ou d'une extrémité cappées, 1 si il s'agit d'une extrémité libre, 2 si il s'agit de l'extrémité de la branche principale
3. 0 ou 1 : 1 si il s'agit de la branche principale (partant de la racine) ; 0 sinon
4. une liste de taille 5 codant le sous arbre de gauche
5. une liste de taille 5 codant le sous arbre de droite

Par exemple, l'arbre initial utilisé pour les simulations du chapitre II, tracé à la figure 2.2 est codé de la manière suivante :

```
A011=list(2,1,0,list(),list())
A0101=list(1,1,0,list(),list())
A0100=list(1,1,0,list(),list())
A010=list(2,0,0,A0100,A0101)
A01=list(3,0,0,A010,A011)

A001=list(1,1,0,list(),list())
A000=list(3,2,1,list(),list())
A00=list(4,0,1,A000,A001)
A0=list(8,0,1,A00,A01)
```

```
11
12 A11=list(3,1,0,list(),list())
13 A10=list(2,1,0,list(),list())
14 A1=list(4,0,0,A10,A11)
15
16 A=list(5,0,1,A0,A1)
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre
- a : l'angle de la branche partant de la racine
- da : l'angle du sous arbre de gauche
- ia : l'angle entre le sous arbre de gauche et le sous arbre de droite
- r $\in [0,1]$: coefficient permettant de réduire l'angle entre la branche de gauche et la branche de droite à chaque génération

créant et renvoyant une liste de matrices 2x2 correspondant aux différents segments de l'arbre.

```
1  CreationArbre=function(L,a,da,ia,r) {
2    S=list()
3    if(length(L)==0) { }
4    else {
5      A4=CreationArbre(L[[4]],da,(da-r*(ia/2)),(ia-r*ia),r)
6      A5=CreationArbre(L[[5]],da-ia,(da-r*(ia/2)),(ia-r*ia),r)
7      l=L[[1]]
8      v=c(l*cos(pi*a/180),l*sin(pi*a/180))
9      S5=TranslateListeSeg(A5,v)
10     S4=TranslateListeSeg(A4,v)
11     if(length(S4)>0) {
12       for(i in 1:length(S4)) {
13         S[[length(S)+1]]=S4[[i]] } }
14     if(length(S5)>0) {
15       for(i in 1:length(S5)) {
16         S[[length(S)+1]]=S5[[i]] } }
17     S[[length(S)+1]]=matrix(c(c(0,0),v),byrow=TRUE,nrow=2) }
18   return(S) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre
- a : l'angle de la branche partant de la racine
- da : l'angle du sous arbre de gauche
- ia : l'angle entre le sous arbre de gauche et le sous arbre de droite
- r in [0,1] : coefficient permettant de réduire l'angle entre la branche de gauche et la branche de droite à chaque génération
- col : un entier définissant la couleur de la racine

créant et renvoyant une liste de matrices 2x2 correspondant aux différents segments de l'arbre avec gestion automatique des couleurs.

```
CreationArbreColor=function(L,a,da,ia,r,col) {
  S=list()
  if(length(L)>0) {
    A4=CreationArbreColor(L[[4]],da,(da-r*(ia/2)),(ia-r*ia),r,col)
    A5=CreationArbreColor(L[[5]],da-ia,(da-r*(ia/2)),(ia-r*ia),r,col+1)
    l=L[[1]]
    v=c(l*cos(pi*a/180),l*sin(pi*a/180))
    S5=TranslateListeSegColor(A5,v)
    S4=TranslateListeSegColor(A4,v)
    if(length(S4)>0) {
      for(i in 1:length(S4)) {
        S[[length(S)+1]]=S4[[i]] } }
    if(length(S5)>0) {
      for(i in 1:length(S5)) {
        S[[length(S)+1]]=S5[[i]] } }
    if(L[[2]]==0) {
      S[[length(S)+1]]=c(col,23) }
    else {
      S[[length(S)+1]]=c(col,5) }
    S[[length(S)+1]]=matrix(c(c(0,0),v),byrow=TRUE,nrow=2) }
  return(S) }
```

4 - Analyse a Tree

4.1 - Number of Extremities, Number of Branches and Length of Polymer

Fonction R prenant en arguments :

* L : une liste définissant un arbre

renvoyant le nombre d'extrémités de l'arbre.

```
NbExt=function(L) {
  E=0
  if(length(L)>0) {
    E5=NbExt(L[[5]])
    E=E+E5
    E4=NbExt(L[[4]])
    E=E+E4
    if (length(L[[4]])==0 & length(L[[5]])==0) {
      E=E+1 } }
  return(E) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

renvoyant le nombre d'extrémités libres (sans protéine de coiffe) de l'arbre.

```
NbExtLibre=function(L){
  E=0
```

```
  if(length(L)>0) {
    E5=NbExtLibre(L[[5]])
    E=E+E5
    E4=NbExtLibre(L[[4]])
    E=E+E4
    if (L[[2]]==2) {
      E=E+L[[2]]-1 }
    else {
      E=E+L[[2]] } }
  return(E) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

renvoyant le nombre d'extrémités libres dont la branche n'est pas de taille nulle de l'arbre.

```
NbExtLibreNonNulle=function(L) {
  s=SizeFeuille(L)
  cpt=0
  for (i in (1:length(s))) {
    if (s[[i]]>0) {
      cpt=cpt+1 } }
  return(cpt) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

renvoyant le nombre de branches de l'arbre.

```
NbBranche=function(L) {
  B=0
  if(length(L)>0) {
    B5=NbBranche(L[[5]])
    B=B+B5
    B4=NbBranche(L[[4]])
    B=B+B4
    B=B+1 }
  return(B) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

renvoyant le nombre de branchement (nœuds) de l'arbre.

```
NbBranchement=function(L) {
  B=NbBranche(L)
  Bt=floor(B/2)
  return(Bt) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

renvoyant le nombre total de monomères dans l'arbre.

```
TailleFil=function(L) {
  Lfil=0
  if(length(L)>0) {
    Lfil5=TailleFil(L[[5]])
    Lfil=Lfil+Lfil5
    Lfil4=TailleFil(L[[4]])
    Lfil=Lfil+Lfil4
    Lfil=Lfil+L[[1]] }
  return(Lfil) }
```

4.2 - Numbering Nodes and Leaves

Fonction R prenant en arguments :

- L : une liste définissant un arbre
- c : une liste de longueur NbBranchement(L) (complétée par la fonction)
- i : un entier fixant la génération dans l'arbre

permettant de numéroter les nœuds (internes) de l'arbre (suivant une numérotation préfixe) et renvoyant la liste c complétée de telle manière que le j-ième éléments de la liste correspond au chemin pour atteindre le j-ième nœud.

```
NumNode1=function(L,c,i) {
  v=c
  TNode=NbBranchement(L)
  LNode=NbBranchement(L[[4]])
  RNode=NbBranchement(L[[5]])

  if(LNode>0) {
    for (j in (i+1) : (i+LNode)) {
      v[[j]]=c(v[[j]],0)
      c=v }
    v=NumNode1(L[[4]],c,i+1) }

  if(RNode>0) {
    for (j in (i+LNode+1) : (i+TNode-1)) {
      v[[j]]=c(v[[j]],1)
      c=v }
    v=NumNode1(L[[5]],c,i+LNode+1) }
  return(v) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

permettant de numéroter les nœuds (internes) de l'arbre (suivant une numérotation préfixe) et renvoyant une liste telle que le j-ième éléments de la liste correspond au chemin pour atteindre le j-ième nœud.

```
NumNode=function(L) {
  c=list()
  c[[NbBranchement(L)+1]]=2
  v=NumNode1(L,c,1)
```

```
  vv=list()
  for (i in 1 : NbBranchement(L)) {
    vv[[i]]=v[[i]] }
  return(vv) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre
- c : une liste de longueur NbBranche(L) (complétée par la fonction)
- i : un entier fixant la génération dans l'arbre

permettant de numéroter les branches de l'arbre (suivant une numérotation préfixe) et renvoyant la liste c complétée de telle manière que le j-ième éléments de la liste correspond au chemin pour atteindre la j-ième branche suivi de la taille de la branche.

```
NumBranch1=function(L,c,i) {
  v=c
  TBranch=NbBranche(L)
  LBranch=NbBranche(L[[4]])
  RBranch=NbBranche(L[[5]])

  if(LBranch>0) {
    for (j in (i+1) : (i+LBranch)) {
      v[[j]]=c(v[[j]],0)
      c=v }
    v=NumBranch1(L[[4]],c,i+1) }

  if(RBranch>0) {
    for (j in (i+LBranch+1) : (i+TBranch-1)) {
      v[[j]]=c(v[[j]],1)
      c=v }
    v=NumBranch1(L[[5]],c,i+LBranch+1) }
  v[[i]]=c(v[[i]],L[[1]])
  return(v) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

permettant de numéroter les branches de l'arbre (suivant une numérotation préfixe) et renvoyant une liste telle que le j-ième éléments de la liste correspond au chemin pour atteindre la j-ième branche suivi de la taille de la branche.

```
NumBranch=function(L) {
  c=list()
  c[[NbBranche(L)+1]]=2
  v=NumBranch1(L,c,1)
  vv=list()
  for (i in 1 : NbBranche(L)) {
    vv[[i]]=v[[i]] }
  return(vv) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

- ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)
- i : un entier fixant une position sur le chemin ch

renvoyant la liste définissant le sous arbre au bout du chemin ch.

```
IsFeuille1=function(L,ch,i) {
  Lfeuille=L
  if (length(ch)>1) {
    if (i<length(ch)) {
      if (ch[i]==1) {
        Lfeuille=IsFeuille1(L[[5]],ch,i+1) }
      else {
        Lfeuille=IsFeuille1(L[[4]],ch,i+1) } } }
  return(Lfeuille) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre
- ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)
- i : un entier fixant une position sur le chemin ch

renvoyant TRUE si la branche au bout du chemin porte une feuille.

```
IsFeuille=function(L,ch,i) {
  ssArb=IsFeuille1(L,ch,i)
  if (length(ssArb[[4]])==0 & length(ssArb[[5]])==0) {
    isF=TRUE }
  else {
    isF=FALSE }
  return(isF) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

renvoyant un vecteur de longueur NbBranche(L) telle que la j-ième coordonnée vaut 1 si la j-ième branche porte une feuille, 0 sinon.

```
NumFeuille1=function(L) {
  v=NumBranch(L)
  I=c(1:length(v))
  for ( i in 1:length(v)) {
    vv=IsFeuille(L,v[[i]],1)
    I[i]=vv[[1]] }
  return(I) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

permettant de numéroter les feuilles de l'arbre (de gauche à droite) et renvoyant une liste telle que le j-ième éléments de la liste correspond au chemin pour atteindre la j-ième feuille suivi de la taille de la branche portant cette feuille.

```
NumFeuille=function(L) {
  v=NumBranch(L)
  I=NumFeuille1(L)
  f=list()
  cpt=1
  for ( i in 1:length(v)) {
    if (I[i]==1) {
      f[[cpt]]=v[[i]]
      cpt=cpt+1 } }
  return(f) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

permettant de numéroter les branches de l'arbre (suivant une numérotation préfixe) et renvoyant une liste telle que le j-ième éléments de la liste correspond à la taille de la branche.

```
SizeBranch=function(L) {
  s=list()
  v=NumBranch(L)
  for (i in 1:length(v)) {
    s[[i]]=v[[i]][length(v[[i]])] }
  return(s) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

permettant de numéroter les feuilles de l'arbre (de gauche à droite) et renvoyant une liste telle que le j-ième éléments de la liste correspond à la taille de la branche portant cette feuille.

```
SizeFeuille=function(L) {
  s=list()
  v=NumFeuille(L)
  for (i in 1:length(v)) {
    s[[i]]=v[[i]][length(v[[i]])] }
  return(s) }
```

5 - Transform a Tree

5.1 - Choose Randomly a Leaf or a Branch

Fonction R prenant en arguments :

* L : une liste définissant un arbre

* ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)

permettant de tirer une feuille uniformément au hasard et renvoyant une liste de taille 2 contenant :

1. le chemin à suivre pour atteindre la feuille choisie,
2. les informations de la branche portant la feuille choisie (liste définissant un arbre).

```
RandomLeaf=function(L,ch) {
  RandL=list()
  RandL[[1]]=ch
  RandL[[2]]=L
  if (length(L[[4]])>0 | length(L[[5]])>0) {
    TLeaf=NbExt(L)
    RLeaf=NbExt(L[[5]])
    Rp=RLeaf/TLeaf
    p=1*(runif(1,0,1)<=Rp)
    ch=c(ch,p)
    if(p==1) {
      RandL=RandomLeaf(L[[5]],ch) }
    else {
      RandL=RandomLeaf(L[[4]],ch) } }
  return(RandL) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre
* ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)

permettant de tirer une extrémité libre uniformément au hasard et renvoyant une liste de taille 2 contenant :

1. le chemin à suivre pour atteindre la feuille choisie,
2. les informations de la branche portant la feuille choisie (liste définissant un arbre).

```
RandomLeafLibre=function(L,ch) {
  RandL=list()
  RandL[[1]]=ch
  RandL[[2]]=L
  if (length(L[[4]])>0 | length(L[[5]])>0) {
    TLeaf=NbExtLibre(L)
    RLeaf=NbExtLibre(L[[5]])
    Rp=RLeaf/TLeaf
    p=1*(runif(1,0,1)<=Rp)
    ch=c(ch,p)
    if(p==1) {
      RandL=RandomLeafLibre(L[[5]],ch) }
    else {
      RandL=RandomLeafLibre(L[[4]],ch) } }
  return(RandL) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre
* ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)

permettant de tirer une extrémité libre dont la branche n'est pas de taille nulle uniformément au hasard et renvoyant une liste de taille 2 contenant :

1. le chemin à suivre pour atteindre la feuille choisie,
2. les informations de la branche portant la feuille choisie (liste définissant un arbre).

```
RandomLeafLibreNonNulle=function(L,ch) {
  l=RandomLeafLibre(L,c())
  s=SizeFeuille(L)
  sums=0
  for (i in (1:length(s))) {
    sums=sums+s[[i]] }
  if (sums == 0) {
    return(l) }
  else {
    while (l[[2]][[1]]==0) {
      l=RandomLeafLibre(L,c()) }
    return(l) } }
```

5.2 - Elongation

Fonction R prenant en arguments :

* L : une liste définissant un arbre
* ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)
* i : un entier fixant une position sur le chemin ch

permettant d'allonger l'extrémité ciblée par le chemin et renvoyant la liste L modifiée.

```
Elongation1=function(L,ch,i) {
  LElong=L
  if (length(ch)>0) {
    if (i<length(ch)) {
      if (ch[i]==1) {
        LElong[[5]]=Elongation1(L[[5]],ch,i+1) }
      else {
        LElong[[4]]=Elongation1(L[[4]],ch,i+1) } }
    else{
      if (ch[i]==1) {
        LElong[[5]][[1]]=LElong[[5]][[1]]+1 }
      else {
        LElong[[4]][[1]]=LElong[[4]][[1]]+1 } } }
  else {
    LElong[[1]]=L[[1]]+1 }
  return(LElong) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

permettant d'allonger une extrémité choisie uniformément au hasard et renvoyant la liste L modifiée.

```
Elongation=function(L) {
  l=RandomLeaf(L,c())
  EL=Elongation1(L,l[[1]],1)
  return(EL) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

permettant d'allonger une extrémité libre choisie uniformément au hasard et renvoyant la liste L modifiée.

```
ElongationLibre=function(L) {
  l=RandomLeafLibre(L,c())
  EL=Elongation1(L,l[[1]],1)
  return(EL) }
```

5.3 - Branching

Fonction R prenant en arguments :

* L : une liste définissant un arbre
* ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)
* i : un entier fixant une position sur le chemin ch
* l : un entier définissant une position sur la branche

permettant d'ajouter un branchement sur le l-ième monomère de la branche ciblée par le chemin et renvoyant la liste L modifiée.

```
Branchement1=function(L,ch,i,l) {
  LBranch=L
  if (length(ch[[1]])>1) {
    if (i<length(ch[[1]])-1) {
      if (ch[[1]][i]==1) {
        LBranch[[5]]=Branchement1(L[[5]],ch,i+1,l) }
      else {
        LBranch[[4]]=Branchement1(L[[4]],ch,i+1,l) } }
    else {
      if (ch[[1]][i]==1) {
        LBranch[[5]][[1]]=l
        LBranch[[5]][[2]]=0
        LBranch[[5]][[4]]=L[[5]]
        LBranch[[5]][[4]][[1]]=L[[5]][[1]]-l
        LBranch[[5]][[5]]=list(0,1,0,list(),list()) }
      else {
        LBranch[[4]][[1]]=l
        LBranch[[4]][[2]]=0
        LBranch[[4]][[4]]=L[[4]]
        LBranch[[4]][[4]][[1]]=L[[4]][[1]]-l
        LBranch[[4]][[5]]=list(0,1,0,list(),list()) } } }
  else {
    LBranch[[1]]=l
    LBranch[[4]]=L
    LBranch[[4]][[1]]=L[[1]]-l
    LBranch[[2]]=0
    LBranch[[5]]=list(0,1,0,list(),list()) }
  return(LBranch) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre
- ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)
- i : un entier fixant une position sur le chemin ch
- l : un entier correspondant à la longueur de la branche portant la feuille

permettant d'ajouter un branchement à la feuille ciblée par le chemin et renvoyant la liste L modifiée.

```
BranchementF1=function(L,ch,i,l) {
  LBranch=L
  if (length(ch)>1) {
    if (i<length(ch)) {
      if (ch[i]==1) {
        LBranch[[5]]=BranchementF1(L[[5]],ch,i+1,l) }
      else {
        LBranch[[4]]=BranchementF1(L[[4]],ch,i+1,l) } }
    else {
      if (ch[i]==1) {
        LBranch[[5]][[1]]=l
        LBranch[[5]][[2]]=0
        LBranch[[5]][[4]]=L[[5]]
        LBranch[[5]][[4]][[1]]=L[[5]][[1]]-l
        LBranch[[5]][[5]]=list(0,1,0,list(),list()) }
      else {
        LBranch[[4]][[1]]=l
        LBranch[[4]][[2]]=0
        LBranch[[4]][[4]]=L[[4]]
        LBranch[[4]][[4]][[1]]=L[[4]][[1]]-l
        LBranch[[4]][[5]]=list(0,1,0,list(),list()) } } }
  else {
    LBranch[[1]]=l
    LBranch[[4]]=L
    LBranch[[4]][[1]]=L[[1]]-l
    LBranch[[2]]=0
    LBranch[[5]]=list(0,1,0,list(),list()) }
  return(LBranch) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

permettant d'ajouter un branchement sur une branche et à une position choisie uniformément au hasard (n'importe où dans l'arbre) et renvoyant la liste L modifiée.

```
Branchement=function(L) {
  v=NumBranch(L)
  s=SizeBranch(L)
  des=0
  if (length(s)>1) {
    tot=0
    snon1=s[(s>1)]
    if (length(snon1)>0) {
      for (i in 1:length(snon1)) {
        tot=tot+snon1[[i]][1] }
      i=1
```

```
      while (s[[i]]==1) {
        i=i+1 }
      sp=s[[i]]
      p=runif(1,0,1)
      des=i*(p<(sp/tot))
      if (des == 0) {
        for (j in (i+1):NbBranche(L)) {
          if (s[[j]]!=1){
            des=des+j*(sp/tot<=p & p<(sp+s[[j]])/tot)
            sp=sp+s[[j]] } } }
      cheminb=v[des] }
    else {
      vv=NumFeuille(L)
      ss=SizeFeuille(L)
      tt=0
      for (i in 1:length(ss)) {
        tt=tt+ss[[i]] }
      ssp=ss[[1]]
      p=runif(1,0,1)
      des=1*(p<(ssp/tt))
      if (length(ss)>1){
        for (i in 2:NbExt(L)){
          des=des+i*(ssp/tt<=p & p<(ssp+ss[[i]])/tt)
          ssp=ssp+ss[[i]] } }
      cheminb=vv[des] } }
  else {
    cheminb=v[1] }
  lmax=cheminb[[1]][length(cheminb[[1]])]
  if(lmax==2 | lmax==1) {
    l=1 }
  else {
    l=floor((lmax-2)*runif(1,0,1)+2) }
  LB=Branchement1(L,cheminb,1,l)
  return(LB) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

permettant d'ajouter un branchement à une feuille choisie uniformément au hasard et renvoyant la liste L modifiée.

```
BranchementFeuille=function(L) {
  l=RandomLeaf(L,c())
  cheminb=l[[1]]
  long=l[[2]][[1]]
  LB=BranchementF1(L,cheminb,1,long)
  return(LB) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

permettant d'ajouter un branchement à une feuille choisie au hasard suivant une loi (non uniforme) selon laquelle plus une branche est longue, plus elle a de chance d'être choisie et renvoyant la liste L modifiée.

```
BranchementFeuilleNonUnif=function(L) {
  v=NumFeuille(L)
  s=SizeFeuille(L)
  if (length(s)>1) {
    t=0
    for (i in 1:length(s)) {
      t=t+s[[i]] }
    if (t>0) {
      sp=s[[1]]
      p=runif(1,0,1)
      des=1*(p<(sp/t))
      if (length(s)>1){
        for (i in 2:length(s)) {
          des=des+i*(sp/t<=p & p<(sp+s[[i]])/t)
          sp=sp+s[[i]] } }
      cheminb=v[des] }
    else {
      l=RandomLeaf(L,c())
      l[[1]]=c(l[[1]],0)
      cheminb=l[1] } }
  else {
    cheminb=v[1] }
  lmax=cheminb[[1]][length(cheminb[[1]])]
  LB=Branchement1(L,cheminb,1,lmax)
  return(LB) }
```

5.4 - Fragmentation

Fonction R prenant en arguments :

* L : une liste définissant un arbre
* ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)
* i : un entier fixant une position sur le chemin ch
* l : un entier définissant une position sur la branche

permettant de fragmenter l'arbre sur le l-ième monomère de la branche ciblée par le chemin et renvoyant la liste L modifiée.

```
Frag1=function(L,ch,i,l) {
  LFrag=L
  if (length(ch[[1]])>1) {
    if (i<length(ch[[1]])-1) {
      if (ch[[1]][i]==1) {
        LFrag[[5]]=Frag1(L[[5]],ch,i+1,l) }
      else {
        LFrag[[4]]=Frag1(L[[4]],ch,i+1,l) } }
    else {
      if (ch[[1]][i]==1) {
        LFrag[[5]][[1]]=l
        LFrag[[5]][[3]]=L[[5]][[3]]
        if (LFrag[[5]][[3]]==1) {
          LFrag[[5]][[2]]=2 }
        else {
```

```
          LFrag[[5]][[2]]=1 }
        LFrag[[5]][[4]]=list()
        LFrag[[5]][[5]]=list() }
      else {
        LFrag[[4]][[1]]=l
        LFrag[[4]][[3]]=L[[4]][[3]]
        if (LFrag[[4]][[3]]==1) {
          LFrag[[4]][[2]]=2 }
        else {
          LFrag[[4]][[2]]=1 }
        LFrag[[4]][[4]]=list()
        LFrag[[4]][[5]]=list() } } }
  else {
    LFrag=list(l,2,1,list(),list()) }
  return(LFrag) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre
- ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)
- i : un entier fixant une position sur le chemin ch
- l : un entier correspondant à la longueur de la branche portant la feuille

permettant de faire tomber la feuille ciblée par le chemin et renvoyant la liste L modifiée.

```
FragF1=function(L,ch,i,l) {
  LFrag=L
  if (length(ch)>1) {
    if (i<length(ch)) {
      if (ch[i]==1) {
        LFrag[[5]]=FragF1(L[[5]],ch,i+1,l) }
      else {
        LFrag[[4]]=FragF1(L[[4]],ch,i+1,l) } }
    else {
      if (ch[i]==1) {
        LFrag[[5]][[1]]=l
        LFrag[[5]][[3]]=L[[5]][[3]]
        if (LFrag[[5]][[3]]==1) {
          LFrag[[5]][[2]]=2 }
        else {
          LFrag[[5]][[2]]=1 }
        LFrag[[5]][[4]]=list()
        LFrag[[5]][[5]]=list() }
      else {
        LFrag[[4]][[1]]=l
        LFrag[[4]][[3]]=L[[4]][[3]]
        if (LFrag[[4]][[3]]==1) {
          LFrag[[4]][[2]]=2 }
        else {
          LFrag[[4]][[2]]=1 }
        LFrag[[4]][[4]]=list()
        LFrag[[4]][[5]]=list() } } }
  else {
    LFrag=list(l,2,1,list(),list()) }
  return(LFrag) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

permettant de fragmenter l'arbre sur une branche et à une position choisie uniformément au hasard (n'importe où dans l'arbre) et renvoyant la liste L modifiée.

```
Fragmentation=function(L) {
  v=NumBranch(L)
  s=SizeBranch(L)
  if (length(s)>1) {
    t=TailleFil(L)
    sp=s[[1]]
    p=runif(1,0,1)
    des=1*(p<(sp/t))
    if (length(s)>1){
      for (i in 2:NbBranche(L)) {
        des=des+i*(sp/t<=p & p<(sp+s[[i]])/t)
        sp=sp+s[[i]] } } }
  else {
    des=1 }
  cheminf=v[des]
  lmax=cheminf[[1]][length(cheminf[[1]])]
  l=floor(lmax*runif(1,0,1)+1)
  LF=Frag1(L,cheminf,1,l)
  return(LF) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

permettant de faire tomber une feuille choisie uniformément au hasard et renvoyant la liste L modifiée.

```
FragmentationFeuille=function(L) {
  l=RandomLeaf(L,c())
  cheminf=l[[1]]
  LF=FragF1(L,cheminf,1,0)
  return(LF) }
```

Fonction R prenant en arguments :

* L : une liste définissant un arbre

Permettant de faire tomber une feuille choisie au hasard suivant une loi (non uniforme) selon laquelle plus une branche est longue, plus elle a de chance d'être choisie et renvoyant la liste L modifiée.

```
FragmentationFeuilleNonUnif=function(L) {
  v=NumFeuille(L)
  s=SizeFeuille(L)
  if (length(s)>1) {
    t=0
    for (i in 1:length(s)) {
      t=t+s[[i]] }
    if (t>0) {
      sp=s[[1]]
      p=runif(1,0,1)
      des=1*(p<(sp/t))
      if (length(s)>1) {
```

```
        for (i in 2:length(s)) {
          des=des+i*(sp/t<=p & p<(sp+s[[i]])/t)
          sp=sp+s[[i]] } }
      cheminf=v[des] }
    else {
      l=RandomLeaf(L,c())
      l[[1]]=c(l[[1]],0)
      cheminf=l[1] } }
  else {
    cheminf=v[1] }
  LF=Frag1(L,cheminf,1,0)
  return(LF) }
```

5.5 - Capping

Fonction R prenant en arguments :

- L : une liste définissant un arbre
- ch : un vecteur de 0 et de 1 définissant un chemin (0=gauche, 1=droite)
- i : un entier fixant une position sur le chemin ch

permettant de capper l'extrémité ciblée par le chemin et renvoyant la liste L modifiée.

```
Capping1=function(L,ch,i) {
  LCap=L
  if (length(ch)>0) {
    if (i<length(ch)) {
      if (ch[i]==1) {
        LCap[[5]]=Capping1(L[[5]],ch,i+1) }
      else {
        LCap[[4]]=Capping1(L[[4]],ch,i+1) } }
    else {
      if (ch[i]==1) {
        LCap[[5]][[2]]=0 }
      else {
        LCap[[4]][[2]]=0 } } }
  else {
    LCap[[2]]=0 }
  return(LCap) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

permettant de capper une extrémité choisie uniformément au hasard et renvoyant la liste L modifiée.

```
Capping=function(L) {
  l=RandomLeafLibre(L,c())
  EL=Capping1(L,l[[1]],1)
  return(EL) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

permettant de capper une extrémité choisie uniformément au hasard parmi les extrémités dont la branche n'est pas de taille nulle et renvoyant la liste L modifiée.

```
CappingNonNulle=function(L) {
  l=RandomLeafLibreNonNulle(L,c())
  EL=Capping1(L,l[[1]],1)
  return(EL) }
```

5.6 - Actin Dynamics

Fonction R prenant en arguments :

- t0 : le temps initial
- A0 : une liste définissant l'arbre inital
- lambda_plus : le taux d'élongation
- beta : le taux de fixation d'une protéine (branchement ou fragmentation)
- a : avantage pour le branchement si positif, pour la fragmentation si négatif
- N : paramètre d'échelle

permettant de faire évoluer l'arbre selon le modèle du chapitre II et renvoyant la liste A0 modifiée.

```
EvolutionListeFacile=function(t0,A0,lambda_plus,beta,a,N) {
  event=0
  while(event==0) {
    Taux_Elongation=lambda_plus*N*NbExt(A0)
    Taux_Binding=beta*N*NbExt(A0)

    Taux_total=Taux_Elongation+Taux_Binding

    des2=1*(runif(1,0,1)<=0.5)

    if(des2==1) {
      p=1*(runif(1,0,1)<=Taux_Elongation/Taux_total)
      if(p==1) {
        A1=Elongation(A0)
        event=1 } }
    else {
      p=1*(runif(1,0,1)<=Taux_Binding/Taux_total)
      if(p==1) {
        p2=1*(runif(1,0,1)<=(0.5+a/N))
        if (p2==1) {
          A1=BranchementFeuille(A0) }
        else {
          A1=FragmentationFeuille(A0) }
        event=1 } } }
  t1=t0+rexp(1,Taux_total)
  L1=list(t1,A1)
  return(L1) }
```

Fonction R prenant en arguments :

- t0 : le temps initial

- A0 : une liste définissant l'arbre inital
- lambda_plus : le taux d'élongation
- beta : le taux de fixation d'une protéine (branchement ou fragmentation)
- a : avantage pour le branchement si positif, pour la fragmentation si négatif
- phiCap : le taux de fixation des protéines de coiffe
- N : paramètre d'échelle

permettant de faire évoluer l'arbre selon le modèle du chapitre II, prenant en compte les protéines de coiffe et renvoyant la liste A0 modifiée.

```
EvolutionListeFacileCapping=function(t0,A0,lambda_plus,beta,a,phiCap,N) {
  event=0
  while(event==0) {
    Taux_Elongation=lambda_plus*N*NbExtLibre(A0)
    Taux_Binding=beta*N*NbExt(A0)
    Taux_Capping=phiCap*N*NbExtLibre(A0)

    Taux_total=Taux_Elongation+Taux_Binding+Taux_Capping

    des3=floor(3*runif(1,0,1)+1)

    if(des3==1) {
      p=1*(runif(1,0,1)<=Taux_Elongation/Taux_total)
      if(p==1) {
        A1=ElongationLibre(A0)
        event=1 } }
    else {
      if(des3==2) {
        p=1*(runif(1,0,1)<=Taux_Binding/Taux_total)
        if(p==1) {
          p2=1*(runif(1,0,1)<=(0.5+a/N))
          if (p2==1) {
            A1=BranchementFeuille(A0) }
          else {
            A1=FragmentationFeuille(A0) }
          event=1 } }
      else {
        p=1*(runif(1,0,1)<=Taux_Capping/Taux_total)
        if(p==1) {
          A1=Capping(A0)
          event=1 } } } }
  t1=t0+rexp(1,Taux_total)
  L1=list(t1,A1)
  return(L1) }
```

Fonction R prenant en arguments :

- t0 : le temps initial
- A0 : une liste définissant l'arbre inital
- lambda_plus : le taux d'élongation
- beta : le taux de fixation d'une protéine (branchement ou fragmentation)

- a : avantage pour le branchement si positif, pour la fragmentation si négatif
- phiCap : le taux de fixation des protéines de coiffe
- N : paramètre d'échelle

permettant de faire évoluer l'arbre selon le modèle du chapitre II, prenant en compte les protéines de coiffe, considérant que celles-ci se fixent uniquement sur les extrémités dont la branche n'est pas de taille nulle et renvoyant la liste A0 modifiée.

```
EvolutionListeFacileCappingNonNulle=function(t0,A0,lambda_plus,beta,a,
phiCap,N) {
  event=0
  while(event==0) {
    Taux_Elongation=lambda_plus*N*NbExtLibre(A0)
    Taux_Binding=beta*N*NbExt(A0)
    Taux_Capping=phiCap*N*NbExtLibreNonNulle(A0)

    Taux_total=Taux_Elongation+Taux_Binding+Taux_Capping

    des3=floor(3*runif(1,0,1)+1)

    if(des3==1) {
      p=1*(runif(1,0,1)<=Taux_Elongation/Taux_total)
      if(p==1) {
        A1=ElongationLibre(A0)
        event=1 } }
    else {
      if(des3==2) {
        p=1*(runif(1,0,1)<=Taux_Binding/Taux_total)
        if(p==1) {
          p2=1*(runif(1,0,1)<=(0.5+a/N))
          if (p2==1) {
            A1=BranchementFeuille(A0) }
          else {
            A1=FragmentationFeuille(A0) }
          event=1 } }
      else {
        p=1*(runif(1,0,1)<=Taux_Capping/Taux_total)
        if(p==1) {
          A1=CappingNonNulle(A0)
          event=1 } } } }
  t1=t0+rexp(1,Taux_total)
  L1=list(t1,A1)
  return(L1) }
```

Fonction R prenant en arguments :

- t0 : le temps initial
- A0 : une liste définissant l'arbre inital
- lambda_plus : le taux d'élongation
- beta : le taux de fixation d'une protéine (branchement ou fragmentation)
- a : avantage pour le branchement si positif, pour la fragmentation si négatif
- phiCap : le taux de fixation des protéines de coiffe

* N : paramètre d'échelle

permettant de faire évoluer l'arbre selon le modèle du chapitre II, prenant en compte les protéines de coiffe, choisissant la branche sur laquelle va avoir lieu un branchement ou une fragmentation de manière non uniforme et renvoyant la liste A0 modifiée.

```
EvolutionListeFacileCappingNonUnif=function(t0,A0,lambda_plus,beta,a,
phiCap,N) {
  event=0
  while(event==0) {
    Taux_Elongation=lambda_plus*N*NbExtLibre(A0)
    Taux_Binding=beta*N*NbExt(A0)
    Taux_Capping=phiCap*N*NbExtLibre(A0)

    Taux_total=Taux_Elongation+Taux_Binding+Taux_Capping

    des3=floor(3*runif(1,0,1)+1)

    if(des3==1) {
      p=1*(runif(1,0,1)<=Taux_Elongation/Taux_total)
      if(p==1) {
        A1=ElongationLibre(A0)
        event=1 } }
    else {
      if(des3==2) {
        p=1*(runif(1,0,1)<=Taux_Binding/Taux_total)
        if(p==1) {
          p2=1*(runif(1,0,1)<=(0.5+a/N))
          if (p2==1) {
            A1=BranchementFeuilleNonUnif(A0) }
          else {
            A1=FragmentationFeuilleNonUnif(A0) }
          event=1 } }
      else {
        p=1*(runif(1,0,1)<=Taux_Capping/Taux_total)
        if(p==1) {
          A1=Capping(A0)
          event=1 } } } }
  t1=t0+rexp(1,Taux_total)
  L1=list(t1,A1)
  return(L1) }
```

Fonction R prenant en arguments :

* t0 : le temps initial
* A0 : une liste définissant l'arbre inital
* lambda_plus : le taux d'élongation
* beta : le taux de fixation d'une protéine (branchement ou fragmentation)
* a : avantage pour le branchement si positif, pour la fragmentation si négatif
* phiCap : le taux de fixation des protéines de coiffe
* N : paramètre d'échelle

permettant de faire évoluer l'arbre selon le modèle du chapitre II, prenant en compte les protéines de coiffe, autorisant les branchements et les fragmentations sur toutes les branches de l'arbre, choisissant la branche sur laquelle va avoir lieu un branchement ou une fragmentation de manière non uniforme et renvoyant la liste A0 modifiée.

```
EvolutionListeCapping=function(t0,A0,lambda_plus,beta,a,phiCap,N) {
  event=0
  while(event==0) {
    Taux_Elongation=lambda_plus*N*NbExtLibre(A0)
    Taux_Binding=beta*N*(TailleFil(A0)-NbBranchement(A0))
    Taux_Capping=phiCap*N*NbExtLibre(A0)

    Taux_total=Taux_Elongation+Taux_Binding+Taux_Capping

    des3=floor(3*runif(1,0,1)+1)

    if(des3==1) {
      p=1*(runif(1,0,1)<=Taux_Elongation/Taux_total)
      if(p==1) {
        A1=ElongationLibre(A0)
        event=1 } }
    else {
      if(des3==2) {
        if (Taux_Binding==Taux_total) {
          p=1 }
        else {
          p=1*(runif(1,0,1)<=Taux_Binding/Taux_total) }
        if(p==1) {
          p2=1*(runif(1,0,1)<=(0.5+a/N))
          if (p2==1) {
            A1=Branchement(A0) }
          else {
            A1=Fragmentation(A0) }
          event=1 } }
      else {
        p=1*(runif(1,0,1)<=Taux_Capping/Taux_total)
        if(p==1) {
          A1=Capping(A0)
          event=1 } } } }
  t1=t0+rexp(1,Taux_total)
  L1=list(t1,A1)
  return(L1) }
```

6 - Plot Results

6.1 - High Function

Fonction R prenant en arguments :

* L : une liste définissant un arbre

renvoyant la fonction hauteur de l'arbre L sous forme d'un vecteur.

```
HauteurGene=function(L) {
  v=NumBranch(L)
  Hg=matrix(0,1,length(v))
  for (i in 1 : length(v)) {
    Hg[i]=length(v[[i]])-1 }
  return(Hg) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

renvoyant un vecteur de couleurs permettant de colorier automatiquement la fonction hauteur de la même manière que l'arbre.

```
ColorHg=function(L) {
  Hg=HauteurGene(L)
  color=matrix(0,1,length(Hg))
  color[1]=1
  i=2
  while (i< length(Hg)+1) {
    if (Hg[i]==Hg[i-1]) {
      color[i]=color[i-1]+1
      i=i+1 }
    else if (Hg[i-1]>Hg[i]) {
      j=i-1
      while (Hg[i] != Hg[j]) {
        j=j-1 }
      color[i]=color[j]+1
      i=i+1 }
    else {
      while (Hg[i-1]<Hg[i]) {
        color[i]=color[i-1]
        i=i+1 } } }
  return(color) }
```

Fonction R prenant en arguments :

- L : une liste définissant un arbre

permettant de tracer la fonction hauteur de l'arbre L avec gestion automatique des couleurs.

```
PlotHg=function(L) {
  color=c('red','darkorange1','goldenrod1','yellowgreen','green4',
  'darkturquoise','deepskyblue3','blue', 'purple3','darkorchid1','magenta')
  palette(color)
  Hg=HauteurGene(L)
  color=ColorHg(L)
  plot(c(0:(length(Hg)-1)),Hg,"l",color='black',main='Depth',
  xlab='Index i',ylab='Depth of node i')
  for (i in 1 : length(color)) {
    lines(c(i-1),c(Hg[i]),col=color[i],'p',pch=19,cex=0.7+8/length(Hg)) } }
```

6.2 - Length of Branches

Fonction R prenant en arguments :

* L : une liste définissant un arbre

permettant de tracer la taille des branches de l'arbre L avec gestion automatique des couleurs.

```
PlotSizeB=function(L) {
  color=c('red','darkorange1','goldenrod1','yellowgreen','green4',
  'darkturquoise','deepskyblue3','blue', 'purple3','darkorchid1','magenta')
  palette(color)
  S=SizeBranch(L)
  color=ColorHg(L)
  plot(c(0:(length(S)-1)),S,"l",color='black',
  main='Length in number of monomers',
  xlab='Index i',ylab='Length of edge i')
  for (i in 1 : length(color)) {
    lines(c(i-1),c(S[i]),col=color[i],'p',pch=19,cex=0.7+8/length(S)) } }
```

6.3 - Temporal Evolution

Fonction R prenant en arguments :

* Tmax : le temps final
* t0 : le temps initial
* A0 : une liste définissant l'arbre inital
* lambda_plus : le taux d'élongation
* beta : le taux de fixation d'une protéine (branchement ou fragmentation)
* a : avantage pour le branchement si positif, pour la fragmentation si négatif
* N : paramètre d'échelle

faisant évoluer l'arbre selon le modèle du chapitre II jusqu'au temps Tmax et permettant de tracer (à chaque itération) une graphique possédant 6 vignettes représentant : l'arbre, la fonction hauteur, la taille des branches, le nombre de monomères dans l'arbre, le nombre de branchements de l'arbre et le nombre d'extrémités. Les graphiques sont automatiquement sauvegardés dans un dossier créé par la fonction.

```
EvolutionTempsFacile=function(Tmax,t0,A0,lambda_plus,beta,a,N) {
  color=c('red','darkorange1','goldenrod1','yellowgreen','green4',
  'darkturquoise','deepskyblue3','blue','purple3','darkorchid1','magenta')
  palette(color)

  T=c(t0)
  Arb0=CreationArbreColor(A0,175,175,170,0.1,1)
  L=c(TailleFil(A0))
  B=c(NbBranchement(A0))
  E=c(NbExt(A0))
  e0=E/N

  NumPlot=1
  dir.create('Chap2evolutionPNG')
  titre=paste0('Chap2evolutionPNG/', paste0(NumPlot,".png"))
  png(file=titre,width = 1024, height = 576, units = "px")
```

```
  par(mfrow=c(2,3),oma=c(1,1,3,1)+0.1 , mar=c(4,4,2,1)+0.1,
  cex.axis = 1.2, cex.lab=1.4, cex.main=1.5)
  PlotListeSegColor(Arb0)
  PlotHg(A0)
  PlotSizeB(A0)
  plot(T,L,'p',main = 'Number of monomers in the polymer', xlab = 'Time t',
  ylab='L(t)',lwd=2,col=7,pch=20,xlim=c(0,Tmax))
  plot(T,B,'p',main = 'Number of branch nodes', xlab = 'Time t',
  ylab='B(t)',lwd=2,col=9,pch=20,xlim=c(0,Tmax))
  plot(T,E/N,'p',main = 'Scaled number of extremities', xlab = 'Time t',
  ylab='X(t)',lwd=2,col=2,pch=20,xlim=c(0,Tmax))
  par(mfrow=c(1,1))
  mtext(text = substitute(paste('Evolution with : ',N == p1 , ', ',
  lambda^"+"== p2 , ', ' , beta== p3 , ", " , a== p4 , ", " , t== p6),
  list(p1=N,p2=lambda_plus,p3=beta,p4=a,p6=t0)), side = 3, line = 0,
  outer = FALSE, padj=-2.2, cex=1.5)
  dev.off()

  while(t0<Tmax) {
    Liste1=EvolutionListeFacile(t0,A0,lambda_plus,beta,a,N)
    t1=Liste1[[1]]
    A1=Liste1[[2]]

    T=c(T,t1)
    Arb1=CreationArbreColor(A1,175,175,170,0.1,1)
    L=c(L,TailleFil(A1))
    B=c(B,NbBranchement(A1))
    E=c(E,NbExt(A1))

    NumPlot=NumPlot+1

    titre=paste0('Chap2evolutionPNG/',paste0(NumPlot,".png"))
    png(file=titre,width = 1024, height = 576, units = "px")
    par(mfrow=c(2,3),oma=c(1,1,3,1)+0.1 , mar=c(4,4,2,1)+0.1 ,
    cex.axis = 1.2, cex.lab=1.4, cex.main=1.5)
    PlotListeSegColor(Arb1)
    PlotHg(A1)
    PlotSizeB(A1)
    plot(T,L,'l',main = 'Number of monomers in the polymer',
    xlab = 'Time t', ylab='L(t)',lwd=2,col=7,xlim=c(0,max(Tmax,t1)))
    plot(T,B,'l',main = 'Number of branch nodes', xlab = 'Time t',
    ylab='B(t)',lwd=2,col=9,xlim=c(0,max(Tmax,t1)))
    plot(T,E/N,'l',main = 'Scaled number of free ends', xlab = 'Time t',
    ylab='X(t)',lwd=2,col=2,xlim=c(0,max(Tmax,t1)))
    par(mfrow=c(1,1))
    mtext(text = substitute(paste('Evolution with : ',N == p1 , ', ',
    lambda^"+"== p2 , ', ' , beta== p3 , ", " , a== p4 , ", " , t== p6),
    list(p1=N,p2=lambda_plus,p3=beta,p4=a,p6=t0)), side = 3, line = 0,
    outer = FALSE, padj=-2.2, cex=1.5)
    dev.off()

    t0=t1
    A0=A1 } }
```

Fonction R prenant en arguments :

- t0 : le temps initial
- A0 : une liste définissant l'arbre inital

- lambda_plus : le taux d'élongation
- beta : le taux de fixation d'une protéine (branchement ou fragmentation)
- a : avantage pour le branchement si positif, pour la fragmentation si négatif
- phiCap : le taux de fixation des protéines de coiffe
- N : paramètre d'échelle

faisant évoluer l'arbre selon le modèle du chapitre II jusqu'au temps Tmax, prenant en compte les protéines de coiffe et permettant de tracer (à chaque itération) une graphique possédant 6 vignettes représentant : l'arbre, la fonction hauteur, la taille des branches, le nombre de monomères dans l'arbre, le nombre de branchements de l'arbre et le nombre d'extrémités. Les graphiques sont automatiquement sauvegardés dans un dossier créé par la fonction.

```
EvolutionTempsFacileCapping=function(Tmax,t0,A0,lambda_plus,beta,a,
phiCap,N) {
  color=c('red','darkorange1','goldenrod1','yellowgreen','green4',
  'darkturquoise','deepskyblue3','blue','purple3','darkorchid1','magenta')
  palette(color)

  T=c(t0)
  Arb0=CreationArbreColor(A0,175,175,170,0.1,1)
  L=c(TailleFil(A0))
  B=c(NbBranchement(A0))
  E=c(NbExt(A0))
  Elibre=c(NbExtLibre(A0))

  NumPlot=1
  dir.create('Chap2evolutionPNG')
  titre=paste0('Chap2evolutionPNG/', paste0(NumPlot,".png"))
  png(file=titre,width = 1024, height = 576, units = "px")
  par(mfrow=c(2,3),oma=c(1,1,3,1)+0.1 , mar=c(4,4,2,1)+0.1 ,
  cex.axis = 1.2, cex.lab=1.4, cex.main=1.5)
  PlotListeSegColor(Arb0)
  PlotHg(A0)
  PlotSizeB(A0)
  plot(T,L,'p',main = 'Number of monomers in the polymer', xlab = 'Time t',
  ylab='L(t)',lwd=2,col=7,pch=20,xlim=c(0,Tmax))
  plot(T,B,'p',main = 'Number of branch nodes', xlab = 'Time t',
  ylab='B(t)',lwd=2,col=9,pch=20,xlim=c(0,Tmax))
  plot(T,E/N,'p',main = 'Scaled number of extremities', xlab = 'Time t',
  ylab='X(t)',lwd=2,col=2,pch=20,xlim=c(0,Tmax),
  ylim=c(min(Elibre/N),max(E/N)))
  lines(T,Elibre/N)
  legend("topleft",c("All extremities","Free extremities"),
  col=c(2,'black'),lty=c(1,1),lwd=c(1.5,1.5),cex=1.2)
  par(mfrow=c(1,1))
  mtext(text = substitute(paste('Evolution with : ',N == p1 , ', ',
  lambda^"+"== p2 , ', ' , beta== p3 , ", " , a== p4 , ", ",
  phi[Cap]== p5 , ", " , t== p6), list(p1=N,p2=lambda_plus,p3=beta,p4=a,
  p5=phiCap,p6=t0)), side = 3, line = 0, outer = FALSE, padj=-2.2, cex=1.5)
  dev.off()

  while(t0<Tmax) {
    Liste1=EvolutionListeFacileCapping(t0,A0,lambda_plus,beta,a,phiCap,N)
    t1=Liste1[[1]]
    A1=Liste1[[2]]

```

```
45      T=c(T,t1)
46      Arb1=CreationArbreColor(A1,175,175,170,0.1,1)
47      L=c(L, TailleFil(A1))
48      B=c(B,NbBranchement(A1))
49      E=c(E,NbExt(A1))
50      Elibre=c(Elibre,NbExtLibre(A1))
51
52      NumPlot=NumPlot+1
53
54      titre=paste0('Chap2evolutionPNG/', paste0(NumPlot,".png"))
55      png(file=titre,width = 1024, height = 576, units = "px")
56      par(mfrow=c(2,3),oma=c(1,1,3,1)+0.1 , mar=c(4,4,2,1)+0.1 ,
57      cex.axis = 1.2, cex.lab=1.4, cex.main=1.5)
58      PlotListeSegColor(Arb1)
59      PlotHg(A1)
60      PlotSizeB(A1)
61      plot(T,L,'l',main = 'Number of monomers in the polymer',
62      xlab = 'Time t', ylab='L(t)',lwd=2,col=7,xlim=c(0,max(Tmax,t1)))
63      plot(T,B,'l',main = 'Number of branch nodes', xlab = 'Time t',
64      ylab='B(t)',lwd=2,col=9,xlim=c(0,max(Tmax,t1)))
65      plot(T,E/N,'l',main = 'Scaled number of free ends', xlab = 'Time t',
66      ylab='X(t)',lwd=2,col=2,xlim=c(0,max(Tmax,t1)),
67      ylim=c(min(Elibre/N),max(E/N)))
68      lines(T,Elibre/N)
69      legend("topleft",c("All extremities","Free extremities"),
70      col=c(2,'black'),lty=c(1,1),lwd=c(1.5,1.5),cex=1.2)
71      par(mfrow=c(1,1))
72      mtext(text = substitute(paste('Evolution with : ',N == p1 , ', ',
73      lambda^"+"== p2 , ', ' , beta== p3 , ", " , a== p4 , ", ",
74      phi[Cap]== p5 , ", " , t== p6),list(p1=N,p2=lambda_plus,p3=beta,p4=a,
75      p5=phiCap,p6=t0)),side = 3, line = 0, outer = FALSE, padj=-2.2, cex=1.5)
76      dev.off()
77      t0=t1
78      A0=A1 } }
```

Fonction R prenant en arguments :

- t0 : le temps initial
- A0 : une liste définissant l'arbre inital
- lambda_plus : le taux d'élongation
- beta : le taux de fixation d'une protéine (branchement ou fragmentation)
- a : avantage pour le branchement si positif, pour la fragmentation si négatif
- phiCap : le taux de fixation des protéines de coiffe
- N : paramètre d'échelle

faisant évoluer l'arbre selon le modèle du chapitre II jusqu'au temps Tmax, prenant en compte les protéines de coiffe, considérant que celles-ci se fixent uniquement sur les extrémités dont la branche n'est pas de taille nulle et permettant de tracer (à chaque itération) une graphique possédant 6 vignettes représentant : l'arbre, la fonction hauteur, la taille des branches, le nombre de monomères dans l'arbre, le nombre de branchements de l'arbre et le nombre d'extrémités. Les graphiques sont automatiquement sauvegardés dans un dossier créé par la fonction.

```
EvolutionTempsFacileCappingNonNulle=function(Tmax,t0,A0,lambda_plus,beta,a,
phiCap,N) {
  color=c('red','darkorange1','goldenrod1','yellowgreen','green4',
  'darkturquoise','deepskyblue3','blue','purple3','darkorchid1','magenta')
  palette(color)

  T=c(t0)
  Arb0=CreationArbreColor(A0,175,175,170,0.1,1)
  L=c(TailleFil(A0))
  B=c(NbBranchement(A0))
  E=c(NbExt(A0))
  Elibre=c(NbExtLibre(A0))

  NumPlot=1
  dir.create('Chap2evolutionPNG')
  titre=paste0('Chap2evolutionPNG/', paste0(NumPlot,".png"))
  png(file=titre,width = 1024, height = 576, units = "px")
  par(mfrow=c(2,3),oma=c(1,1,3,1)+0.1 , mar=c(4,4,2,1)+0.1 ,
  cex.axis = 1.2, cex.lab=1.4, cex.main=1.5)
  PlotListeSegColor(Arb0)
  PlotHg(A0)
  PlotSizeB(A0)
  plot(T,L,'p',main = 'Number of monomers in the polymer', xlab = 'Time t',
  ylab='L(t)',lwd=2,col=7,pch=20,xlim=c(0,Tmax))
  plot(T,B,'p',main = 'Number of branch nodes', xlab = 'Time t',
  ylab='B(t)',lwd=2,col=9,pch=20,xlim=c(0,Tmax))
  plot(T,E/N,'p',main = 'Scaled number of extremities', xlab = 'Time t',
  ylab='X(t)',lwd=2,col=2,pch=20,xlim=c(0,Tmax),
  ylim=c(min(Elibre/N),max(E/N)))
  lines(T,Elibre/N)
  legend("topleft",c("All extremities","Free extremities"),
  col=c(2,'black'),lty=c(1,1),lwd=c(1.5,1.5),cex=1.2)
  par(mfrow=c(1,1))
  mtext(text = substitute(paste('Evolution with : ',N == p1 , ', ',
  lambda^"+"== p2 , ', ' , beta== p3 , ", " , a== p4 , ", ",
  phi[Cap]== p5 , ", " , t== p6),list(p1=N,p2=lambda_plus,p3=beta,p4=a,
  p5=phiCap,p6=t0)), side = 3, line = 0, outer = FALSE, padj=-2.2, cex=1.5)
  dev.off()

  while(t0<Tmax) {
    Liste1=EvolutionListeFacileCappingNonNulle(t0,A0,lambda_plus,beta,a,
    phiCap,N)
    t1=Liste1[[1]]
    A1=Liste1[[2]]

    T=c(T,t1)
    Arb1=CreationArbreColor(A1,175,175,170,0.1,1)
    L=c(L,TailleFil(A1))
    B=c(B,NbBranchement(A1))
    E=c(E,NbExt(A1))
    Elibre=c(Elibre,NbExtLibre(A1))

    NumPlot=NumPlot+1

    titre=paste0('Chap2evolutionPNG/', paste0(NumPlot,".png"))
    png(file=titre,width = 1024, height = 576, units = "px")
    par(mfrow=c(2,3),oma=c(1,1,3,1)+0.1 , mar=c(4,4,2,1)+0.1 ,
    cex.axis = 1.2, cex.lab=1.4, cex.main=1.5)
    PlotListeSegColor(Arb1)
    PlotHg(A1)
```

```
    PlotSizeB(A1)
    plot(T,L,'l',main = 'Number of monomers in the polymer',
    xlab = 'Time t', ylab='L(t)',lwd=2,col=7,xlim=c(0,max(Tmax,t1)))
    plot(T,B,'l',main = 'Number of branch nodes', xlab = 'Time t',
    ylab='B(t)',lwd=2,col=9,xlim=c(0,max(Tmax,t1)))
    plot(T,E/N,'l',main = 'Scaled number of free ends', xlab = 'Time t',
    ylab='X(t)',lwd=2,col=2,xlim=c(0,max(Tmax,t1)),
    ylim=c(min(Elibre/N),max(E/N)))
    lines(T,Elibre/N)
    legend("topleft",c("All extremities","Free extremities"),
    col=c(2,'black'),lty=c(1,1),lwd=c(1.5,1.5),cex=1.2)
    par(mfrow=c(1,1))
    mtext(text = substitute(paste('Evolution with : ',N == p1 , ', ',
    lambda^"+"== p2 , ', ' , beta== p3 , ", " , a== p4 , ", ",
    phi[Cap]== p5 , ", " , t== p6),list(p1=N,p2=lambda_plus,p3=beta,p4=a,
    p5=phiCap,p6=t0)), side = 3, line = 0, outer = FALSE, padj=-2.2, cex=1.5)
    dev.off()
    t0=t1
    A0=A1 } }
```

Fonction R prenant en arguments :

- t0 : le temps initial
- A0 : une liste définissant l'arbre inital
- lambda_plus : le taux d'élongation
- beta : le taux de fixation d'une protéine (branchement ou fragmentation)
- a : avantage pour le branchement si positif, pour la fragmentation si négatif
- phiCap : le taux de fixation des protéines de coiffe
- N : paramètre d'échelle

faisant évoluer l'arbre selon le modèle du chapitre II jusqu'au temps Tmax, prenant en compte les protéines de coiffe, choisissant la branche sur laquelle va avoir lieu un branchement ou une fragmentation de manière non uniforme et permettant de tracer (à chaque itération) une graphique possédant 6 vignettes représentant : l'arbre, la fonction hauteur, la taille des branches, le nombre de monomères dans l'arbre, le nombre de branchements de l'arbre et le nombre d'extrémités. Les graphiques sont automatiquement sauvegardés dans un dossier créé par la fonction.

```
EvolutionTempsFacileCappingNonUnif=function(Tmax,t0,A0,lambda_plus,beta,a,
phiCap,N) {
  color=c('red','darkorange1','goldenrod1','yellowgreen','green4',
  'darkturquoise','deepskyblue3','blue','purple3','darkorchid1','magenta')
  palette(color)

  T=c(t0)
  Arb0=CreationArbreColor(A0,175,175,170,0.1,1)
  L=c(TailleFil(A0))
  B=c(NbBranchement(A0))
  E=c(NbExt(A0))
  Elibre=c(NbExtLibre(A0))

  NumPlot=1
  dir.create('Chap2evolutionPNG')
```

```
  titre=paste0('Chap2evolutionPNG/', paste0(NumPlot,".png"))
  png(file=titre,width = 1024, height = 576, units = "px")
  par(mfrow=c(2,3),oma=c(1,1,3,1)+0.1 , mar=c(4,4,2,1)+0.1 ,
  cex.axis = 1.2, cex.lab=1.4, cex.main=1.5)
  PlotListeSegColor(Arb0)
  PlotHg(A0)
  PlotSizeB(A0)
  plot(T,L,'p',main = 'Number of monomers in the polymer', xlab = 'Time t',
  ylab='L(t)',lwd=2,col=7,pch=20,xlim=c(0,Tmax))
  plot(T,B,'p',main = 'Number of branch nodes', xlab = 'Time t',
  ylab='B(t)',lwd=2,col=9,pch=20,xlim=c(0,Tmax))
  plot(T,E/N,'p',main = 'Scaled number of extremities', xlab = 'Time t',
  ylab='X(t)',lwd=2,col=2,pch=20,xlim=c(0,Tmax),
  ylim=c(min(Elibre/N),max(E/N)))
  lines(T,Elibre/N)
  legend("topleft",c("All extremities","Free extremities"),
  col=c(2,'black'),lty=c(1,1),lwd=c(1.5,1.5),cex=1.2)
  par(mfrow=c(1,1))
  mtext(text = substitute(paste('Evolution with : ',N == p1 , ', ',
  lambda^"+"== p2 , ', ' , beta== p3 , ", " , a== p4 , ", ",
  phi[Cap]== p5 , ", " , t== p6),list(p1=N,p2=lambda_plus,p3=beta,p4=a,
  p5=phiCap,p6=t0)), side = 3, line = 0, outer = FALSE, padj=-2.2, cex=1.5)
  dev.off()

  while(t0<Tmax) {
    Liste1=EvolutionListeFacileCappingNonUnif(t0,A0,lambda_plus,beta,a,
    phiCap,N)
    t1=Liste1[[1]]
    A1=Liste1[[2]]

    T=c(T,t1)
    Arb1=CreationArbreColor(A1,175,175,170,0.1,1)
    L=c(L,TailleFil(A1))
    B=c(B,NbBranchement(A1))
    E=c(E,NbExt(A1))
    Elibre=c(Elibre,NbExtLibre(A1))

    NumPlot=NumPlot+1

    titre=paste0('Chap2evolutionPNG/', paste0(NumPlot,".png"))
    png(file=titre,width = 1024, height = 576, units = "px")
    par(mfrow=c(2,3),oma=c(1,1,3,1)+0.1 , mar=c(4,4,2,1)+0.1 ,
    cex.axis = 1.2, cex.lab=1.4, cex.main=1.5)
    PlotListeSegColor(Arb1)
    PlotHg(A1)
    PlotSizeB(A1)
    plot(T,L,'l',main = 'Number of monomers in the polymer',
    xlab = 'Time t', ylab='L(t)',lwd=2,col=7,xlim=c(0,max(Tmax,t1)))
    plot(T,B,'l',main = 'Number of branch nodes', xlab = 'Time t',
    ylab='B(t)',lwd=2,col=9,xlim=c(0,max(Tmax,t1)))
    plot(T,E/N,'l',main = 'Scaled number of free ends', xlab = 'Time t',
    ylab='X(t)',lwd=2,col=2,xlim=c(0,max(Tmax,t1)),
    ylim=c(min(Elibre/N),max(E/N)))
    lines(T,Elibre/N)
    legend("topleft",c("All extremities","Free extremities"),
    col=c(2,'black'),lty=c(1,1),lwd=c(1.5,1.5),cex=1.2)
    par(mfrow=c(1,1))
    mtext(text = substitute(paste('Evolution with : ',N == p1 , ', ',
    lambda^"+"== p2 , ', ' , beta== p3 , ", " , a== p4 , ", ",
    phi[Cap]== p5 , ", " , t== p6),list(p1=N,p2=lambda_plus,p3=beta,p4=a,
    p5=phiCap,p6=t0)), side = 3, line = 0, outer = FALSE, padj=-2.2, cex=1.5)
```

```
    dev.off()
    t0=t1
    A0=A1  }  }
```

Fonction R prenant en arguments :

- t0 : le temps initial
- A0 : une liste définissant l'arbre inital
- lambda_plus : le taux d'élongation
- beta : le taux de fixation d'une protéine (branchement ou fragmentation)
- a : avantage pour le branchement si positif, pour la fragmentation si négatif
- phiCap : le taux de fixation des protéines de coiffe
- N : paramètre d'échelle

faisant évoluer l'arbre selon le modèle du chapitre II jusqu'au temps Tmax, prenant en compte les protéines de coiffe, autorisant les branchements et les fragmentations sur toutes les branches de l'arbre, choisissant la branche sur laquelle va avoir lieu un branchement ou une fragmentation de manière non uniforme et permettant de tracer (à chaque itération) une graphique possédant 6 vignettes représentant : l'arbre, la fonction hauteur, la taille des branches, le nombre de monomères dans l'arbre, le nombre de branchements de l'arbre et le nombre d'extrémités. Les graphiques sont automatiquement sauvegardés dans un dossier créé par la fonction.

```
EvolutionTempsCapping=function(Tmax,t0,A0,lambda_plus,beta,a,phiCap,N) {
  color=c('red','darkorange1','goldenrod1','yellowgreen','green4',
  'darkturquoise','deepskyblue3','blue','purple3','darkorchid1','magenta')
  palette(color)

  T=c(t0)
  Arb0=CreationArbreColor(A0,175,175,170,0.1,1)
  L=c(TailleFil(A0))
  B=c(NbBranchement(A0))
  E=c(NbExt(A0))
  Elibre=c(NbExtLibre(A0))

  NumPlot=1
  dir.create('Chap2evolutionPNG')
  titre=paste0('Chap2evolutionPNG/', paste0(NumPlot,".png"))
  png(file=titre,width = 1024, height = 576, units = "px")
  par(mfrow=c(2,3),oma=c(1,1,3,1)+0.1 , mar=c(4,4,2,1)+0.1 ,
  cex.axis = 1.2, cex.lab=1.4, cex.main=1.5)
  PlotListeSegColor(Arb0)
  PlotHg(A0)
  PlotSizeB(A0)
  plot(T,L,'p',main = 'Number of monomers in the polymer',xlab = 'Time t',
  ylab='L(t)',lwd=2,col=7,pch=20,xlim=c(0,Tmax))
  plot(T,B,'p',main = 'Number of branch nodes', xlab = 'Time t',
  ylab='B(t)',lwd=2,col=9,pch=20,xlim=c(0,Tmax))
  plot(T,E/N,'p',main = 'Scaled number of extremities', xlab = 'Time t',
  ylab='X(t)',lwd=2,col=2,pch=20,xlim=c(0,Tmax),
  ylim=c(min(Elibre/N),max(E/N)))
  lines(T,Elibre/N)
  legend("topleft",c("All extremities","Free extremities"),
```

```
col=c(2,'black'),lty=c(1,1),lwd=c(1.5,1.5),cex=1.2)
par(mfrow=c(1,1))
mtext(text = substitute(paste('Evolution with : ',N == p1 , ', ',
lambda^"+"== p2 , ', ' , beta== p3 , ", " , a== p4 , ", ",
phi[Cap]== p5 , ", " , t== p6),list(p1=N,p2=lambda_plus,p3=beta,p4=a,
p5=phiCap,p6=t0)), side = 3, line = 0, outer = FALSE, padj=-2.2, cex=1.5)
dev.off()

while(t0<Tmax) {
  Liste1=EvolutionListeCapping(t0,A0,lambda_plus,beta,a,phiCap,N)
  t1=Liste1[[1]]
  A1=Liste1[[2]]

  T=c(T,t1)
  Arb1=CreationArbreColor(A1,175,175,170,0.1,1)
  L=c(L,TailleFil(A1))
  B=c(B,NbBranchement(A1))
  E=c(E,NbExt(A1))
  Elibre=c(Elibre,NbExtLibre(A1))

  NumPlot=NumPlot+1

  titre=paste0('Chap2evolutionPNG/',paste0(NumPlot,".png"))
  png(file=titre,width = 1024, height = 576, units = "px")
  par(mfrow=c(2,3),oma=c(1,1,3,1)+0.1 , mar=c(4,4,2,1)+0.1 ,
  cex.axis = 1.2, cex.lab=1.4, cex.main=1.5)
  PlotListeSegColor(Arb1)
  PlotHg(A1)
  PlotSizeB(A1)
  plot(T,L,'l',main = 'Number of monomers in the polymer',
  xlab = 'Time t',ylab='L(t)',lwd=2,col=7,xlim=c(0,max(Tmax,t1)))
  plot(T,B,'l',main = 'Number of branch nodes', xlab = 'Time t',
  ylab='B(t)',lwd=2,col=9,xlim=c(0,max(Tmax,t1)))
  plot(T,E/N,'l',main = 'Scaled number of free ends', xlab = 'Time t',
  ylab='X(t)',lwd=2,col=2,xlim=c(0,max(Tmax,t1)),
  ylim=c(min(Elibre/N),max(E/N)))
  lines(T,Elibre/N)
  legend("topleft",c("All extremities","Free extremities"),
  col=c(2,'black'),lty=c(1,1),lwd=c(1.5,1.5),cex=1.2)
  par(mfrow=c(1,1))
  mtext(text = substitute(paste('Evolution with : ',N == p1 , ', ',
  lambda^"+"== p2 , ', ' , beta== p3 , ", " , a== p4 , ", ",
  phi[Cap]== p5 , ", " , t== p6),list(p1=N,p2=lambda_plus,p3=beta,p4=a,
  p5=phiCap,p6=t0)), side = 3, line = 0, outer = FALSE, padj=-2.2, cex=1.5)
  dev.off()
  t0=t1
  A0=A1 } }
```

Appendix C

C++ Code for the Model Considering all Accessory Proteins

Contents

1 - Main Functions

1.1 - Stochastic Model

Fonction C++ prenant en arguments :

- int N : le nombre total de monomères dans le système
- int n1 : le nombre de monomères inclus dans les filaments simples
- int n2 : le nombre de complexes actine-G/profiline
- int n3 : le nombre de monomères inclus dans les filaments associés à une formine
- int n4 : le nombre de monomères inclus dans les filaments linéaires cappés
- int n5 : le nombre de monomères inclus dans les filaments branchés
- int l0 : la taille initiale des filaments
- int b0 : le nombre initial de branchements (pour les filaments branchés)
- int e0 : le nombre intial d'extrémités libres (pour les filaments branchés)
- int vi : entier permettant de numéroter le fichier résultat

et créant une map contenant l'état initial souhaité.

```
std::map<std::string,int> GeneInitMapParam(int N,int n1,int n2,int n3,
int n4,int n5,int l0,int b0,int e0, int vi) {
  std::map<std::string,int> K0;

  std::string sl0 = std::to_string(l0);
  std::string sb0 = std::to_string(b0);
  std::string se0 = std::to_string(e0);

  std::string svi = std::to_string(vi);

  if (N-n1-n2-n3-n4-n5>0) {
    K0["M;0;1;0;"]=N-n1-n2-n3-n4-n5; }
  if (n1>0) {
    K0["S;0;"+sl0+";1;"]=n1/l0; }
  if (n2>0) {
    K0["P;0;1;0;"]=n2; }
  if (n3>0) {
    K0["F;0;"+sl0+";1;"]=n3/l0; }
  if (n4>0) {
    K0["C;0;"+sl0+";0;"]=n4/l0; }
  if (n5>0) {
    K0["B;"+sb0+";"+sl0+";"+se0+";"]=n5/l0; }

  std::string NomFichier = "ValeursInitiales"+svi+".txt";

  FILE* outInit = fopen(NomFichier.c_str(),"w");

  fprintf(outInit, "Libres = %d \n", (N-n1-n2-n3-n4-n5));
  fprintf(outInit, "Simples = %d \n", n1);
  fprintf(outInit, "Complexes = %d \n", n2);
  fprintf(outInit, "Formine = %d \n", n3);
  fprintf(outInit, "Cappes = %d \n", n4);
  fprintf(outInit, "Branchants = %d \n", n5);
  fprintf(outInit, "Longueur = %d \n", l0);
  if (n5>0) {
    fprintf(outInit, "Branchements = %d \n", b0);
    fprintf(outInit, "Extremites = %d \n", e0); }
  fclose(outInit);

  return K0; }
```

Fonction C++ prenant en arguments :

- std::map<std::string,int> K0 : map initiale obtenue grâce à la fonction GeneInitMapParam
- double t0 : le temps initial
- int N : le nombre total de monomères dans le système
- double tmax : le temps final
- double lambda_plus : le taux d'élongation spontanée
- double lambda_moins : le taux de dépolymérisation
- double lambdaF_plus : le taux d'élongation avec une formine
- double lambdaN : le taux de nucléation
- double phiP : le taux de création de complexes actin-G/profilin
- double phiF_plus : le taux de fixation d'une formine sur un polymère

- double phiF_moins : le taux de libération d'une formine
- double phiA_plus : le taux de fixation d'un Arp2/3 sur un polymère
- double phiA_moins : le taux de libération d'un Arp2/3
- double phiC : le taux de capping
- double phiT : le taux de fragmentation
- int vi : entier permettant de numéroter le fichier résultat

et modifiant la map selon le modèle décrit dans la Section 4.2 du Chapitre III.

```
std::map<std::string,int> PopulationTotaleMap(std::map<std::string,int> K0,
  double t0, int N, double tmax, double lambda_plus, double lambda_moins,
  double lambdaF_plus, double lambdaN, double phiP, double phiF_plus,
  double phiF_moins, double phiA_plus, double phiA_moins, double phiC,
  double phiT, int vi) {
  std::string svi = std::to_string(vi);

  std::string NomFichier = "ValeursParametres"+svi+".txt";
  //On crée un fichier pour récupérer la valeur des paramètres
  FILE* outparametre = fopen(NomFichier.c_str(),"w");

  //On écrit la valeur des paramètres
  fprintf(outparametre,"N = %i", N);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lambda_plus = %e", lambda_plus);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lambdaF_plus = %e", lambdaF_plus);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lambda_moins = %e", lambda_moins);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lambdaN = %e", lambdaN);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiP = %e", phiP);
  fprintf(outparametre, "\n");
  fprintf(outparametre, "PhiF_plus = %e", phiF_plus);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiF_moins = %e", phiF_moins);
  fprintf(outparametre, "\n");
  fprintf(outparametre, "PhiA_plus = %e", phiA_plus);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiA_moins = %e", phiA_moins);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiC = %e", phiC);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiT = %e", phiT);
  fprintf(outparametre, "\n");

  //On ferme le fichier pour les paramètres
  fclose(outparametre);

  //On crée un fichier pour l'évolution du système
  NomFichier = "DistributionS"+svi+".txt";
  FILE* outputS = fopen(NomFichier.c_str(),"w");
  NomFichier = "DistributionF"+svi+".txt";
  FILE* outputF = fopen(NomFichier.c_str(),"w");
  NomFichier = "DistributionC"+svi+".txt";
  FILE* outputC = fopen(NomFichier.c_str(),"w");
```

```
48  NomFichier = "DistributionB"+svi+".txt";
49  FILE* outputB = fopen(NomFichier.c_str(),"w");
50
51  NomFichier = "MoyennesBE"+svi+".txt";
52  FILE* outputMoy = fopen(NomFichier.c_str(),"w");
53
54  NomFichier = "DistributionBranche"+svi+".txt";
55  FILE* outputBranche = fopen(NomFichier.c_str(),"w");
56  NomFichier = "DistributionExtremite"+svi+".txt";
57  FILE* outputExtremite = fopen(NomFichier.c_str(),"w");
58
59  //et un fichier pour les proportions de filaments dans chaque
60  //sous-population
61  NomFichier = "Proportions"+svi+".txt";
62  FILE* outputprop = fopen(NomFichier.c_str(),"w");
63
64  //On initialise le temps
65  double T0 = t0;
66
67  //On écrit l'état initial dans le fichier résultat sous la forme
68  //temps;type;#branchements;longueur;#extremites;quantité
69  std::map<std::string, int>::iterator it;
70  //On initialise les nombres de monomères presents dans chaque
71  //sous-population
72
73  int N0=K0["M;0;1;0;"]; //monomères libres
74  int N1=0; //filaments simples
75  int N2=K0["P;0;1;0;"];    //complexes G-actin/proilin
76  int N3=0;    //filaments+formin
77  int N4=0;    //filaments linéaires cappés
78  int N5=0;    //filaments branchés
79
80  int NL; //Longueur en cours
81  int BL; //Branchement en cours
82  int EL; //Extremite en cours
83
84  int NbFilS=0; //Nb de filaments simples
85  int NbFilF=0; //Nb de filaments formine
86  int NbFilC=0; //Nb de filaments cappé
87  int NbFilB=0; //Nb de filaments branchants
88
89  int DistS[N+1]; //Dans case i, nombre de filaments simples de taille i
90  int DistF[N+1]; //Dans case i, nombre de filaments formines de taille i
91  int DistC[N+1]; //Dans case i, nombre de filaments cappés de taille i
92  int DistB[N+1]; //Dans case i, nombre de filaments branchants de taille i
93
94  int Bran[N+1]; //Dans case i, nombre de filaments avec i branchements
95  int Ext[N+1]; //Dans case i, nombre de filaments avec i extremites libres
96
97  int ExtB[N+1]; //Dans case i, nombre de filaments branchants avec i
98  //extremites libres
99
100 for (int i=0; i<N+1; i++) {
101   DistS[i]=0;
102   DistF[i]=0;
103   DistC[i]=0;
104   DistB[i]=0;
105   Bran[i]=0;
106   Ext[i]=0;
107   ExtB[i]=0; }
108
```

```
109  double MoyBranch=0.0; //Nb moyen de branchements parmi filaments
110  //branchants =somme[b*nb fil. avec b branchements/nb fil. branchants]
111  double MoyExt=0.0; //Nb moyen d'extremite libres parmi filaments
112  //branchants
113
114  for(it=K0.begin(); it!=K0.end(); ++it){
115     NL=findL(it->first.c_str());
116     BL=findB(it->first.c_str());
117     EL=findE(it->first.c_str());
118     if (it->first.c_str()[0]!='M' && it->first.c_str()[0]!='P'
119     && it->first.c_str()[0]!='e') {
120        Ext[EL]=Ext[EL]+it->second;
121        Bran[BL]=Bran[BL]+it->second; }
122     if(it->first.c_str()[0]=='S') {
123        N1=N1+it->second*NL;
124        NbFilS=NbFilS+it->second;
125        DistS[NL]=it->second; }
126     else if(it->first.c_str()[0]=='F') {
127        N3=N3+it->second*NL;
128        NbFilF=NbFilF+it->second;
129        DistF[NL]=it->second; }
130     else if(it->first.c_str()[0]=='C') {
131        N4=N4+it->second*NL;
132        NbFilC=NbFilC+it->second;
133        DistC[NL]=it->second; }
134     else if(it->first.c_str()[0]=='B') {
135        N5=N5+it->second*NL;
136        NbFilB=NbFilB+it->second;
137        DistB[NL]=it->second;
138        ExtB[EL]=ExtB[EL]+it->second; } }
139
140   if (NbFilB == 0) {
141     MoyBranch=0.0;
142     MoyExt=0.0; }
143   else {
144     for (int j=1; j<N+1; j++) {
145        MoyBranch=MoyBranch+j*Bran[j];
146        MoyExt=MoyExt+j*ExtB[j]; }
147
148     MoyBranch=MoyBranch/double(NbFilB);
149     MoyExt=MoyExt/double(NbFilB); }
150
151   fprintf(outputprop, "%f ; ", T0);
152   fprintf(outputprop, "%d ; ", N0);
153   fprintf(outputprop, "%d ; ", N1);
154   fprintf(outputprop, "%d ; ", N2);
155   fprintf(outputprop, "%d ; ", N3);
156   fprintf(outputprop, "%d ; ", N4);
157   fprintf(outputprop, "%d ; ", N5);
158   fprintf(outputprop, "%d ;",N0+N1+N2+N3+N4+N5);
159
160   fprintf(outputprop, "%d ; ", NbFilS);
161   fprintf(outputprop, "%d ; ", NbFilF);
162   fprintf(outputprop, "%d ; ", NbFilC);
163   fprintf(outputprop, "%d ;\n ", NbFilB);
164
165   fprintf(outputS, "%f ; ", T0);
166   fprintf(outputF, "%f ; ", T0);
167   fprintf(outputC, "%f ; ", T0);
168   fprintf(outputB, "%f ; ", T0);
169
```

```
170     fprintf(outputBranche, "%f ; ", T0);
171     fprintf(outputExtremite, "%f ; ", T0);
172 
173     for (int i=0; i<N+1; i++) {
174       fprintf(outputS, "%d ; ", DistS[i]);
175       fprintf(outputF, "%d ; ", DistF[i]);
176       fprintf(outputC, "%d ; ", DistC[i]);
177       fprintf(outputB, "%d ; ", DistB[i]);
178       fprintf(outputBranche, "%d ; ", Bran[i]);
179       fprintf(outputExtremite, "%d ; ", Ext[i]); }
180 
181     fprintf(outputS, "\n");
182     fprintf(outputF, "\n");
183     fprintf(outputC, "\n");
184     fprintf(outputB, "\n");
185 
186     fprintf(outputBranche, "\n");
187     fprintf(outputExtremite, "\n");
188 
189     fprintf(outputMoy, "%f ; ", T0);
190     fprintf(outputMoy, "%f ; ", MoyBranch);
191     fprintf(outputMoy, "%f ;\n ", MoyExt);
192 
193     //On initialise l'état suivant
194     double T1=t0;
195     printf("%f\n", T1);
196 
197     std::map<std::string,int> K1;
198     std::map<std::string, int>::iterator it1;
199 
200     //On initialise le taux de chaque évenement possible
201     double taux_elongation;
202     double taux_depolymerisation;
203     double taux_nucleation;
204     double taux_creation_complexe;
205     double taux_fixation_formine;
206     double taux_liberation_formine;
207     double taux_fixation_arp;
208     double taux_liberation_arp;
209     double taux_capping;
210     double taux_fragmentation;
211     double taux_total_filament;
212     double taux_total_pop;
213 
214     bool event = 0;
215     bool evolution = 1;
216     double alea;
217     int filaments;
218     int somme;
219     char population;
220     int branchement;
221     int longueur;
222     int extremite;
223 
224     int positionSep[4];
225     int pos;
226     int pospos;
227 
228     int LastPrint=0;
229     int PrintT;
230 
```

```
231   int LastPrint2=0;
232   int PrintT2;
233
234   //Tant que le tmax choisi n'est pas atteint, on laisse évoluer le
235   //système
236   while (T0 < tmax && evolution == 1) {
237     //On calcule le taux total d'évenement dans la population
238     //i.e. la somme de tous les taux
239     taux_total_pop = TauxTotalPop(K0,N,lambda_plus, lambda_moins,
240       lambdaF_plus, lambdaN, phiP, phiF_plus, phiF_moins,phiA_plus,
241       phiA_moins, phiC, phiT);
242     //On vérifie qu'il peut se passer quelque chose
243     //(e.g. si il n'y plus aucun filament et que tous les monomères sont
244     //en complexe avec de la profiline alors il ne peut rien se passer)
245     if (taux_total_pop == 0) {
246       evolution = 0; }
247     else {
248       //Tant qu'aucun évenement n'a eu lieu
249       while (event == 0) {
250         //On tire le type de filament qui va tenter sa chance
251         //(en jeu : simple,formine,cappé,branchant)
252         filaments=0;
253         for(it=K0.begin(); it!=K0.end(); ++it) {
254           filaments=filaments+it->second; }
255         filaments=filaments-K0["M;0;1;0;"]-K0["P;0;1;0;"];
256
257         alea=rand01();
258         somme=0;
259         longueur=0;
260
261         if (filaments == 0) {
262           K1=EvolM(K0,N,lambdaN,phiP);
263           event=K1["event"]; }
264         else {
265           std::map<std::string, int>::iterator it2;
266           it2=K0.begin();
267           while(it2!=K0.end() && longueur==0) { //Tirage du filaments
268             if(it2->first.c_str()[0]!='M' && it2->first.c_str()[0]!='P'
269             && alea > (double(somme)/double(filaments))
270             && alea <= (double(somme+it2->second)/double(filaments))) {
271               pospos=0;
272               pos=0;
273               while (pospos<4) {
274                 if(it2->first.c_str()[pos]==';') {
275                   positionSep[pospos]=pos;
276                   pospos=pospos+1;
277                   pos=pos+1; }
278                 else {
279                   pos=pos+1; } }
280
281               population=it2->first.c_str()[0];
282
283               branchement=0;
284               for (int j=positionSep[0]+1; j<positionSep[1]; j++) {
285                 branchement=branchement*10+(it2->first.c_str()[j]-'0'); }
286
287               longueur=0;
288               for (int j=positionSep[1]+1; j<positionSep[2]; j++) {
289                 longueur=longueur*10+(it2->first.c_str()[j]-'0'); }
290
291               extremite=0;
```

```
292             for (int j=positionSep[2]+1; j<positionSep[3]; j++) {
293               extremite=extremite*10+(it2->first.c_str()[j]-'0'); } }
294           else {
295             if (it2->first.c_str()[0]=='M'||it2->first.c_str()[0]=='P') {
296               ++it2; }
297             else {
298               somme=somme+it2->second;
299               ++it2; } } }
300
301         //On tire l'évenement qui va potentiellement se produire
302         if (population == 'S') {
303           K1=EvolS(K0,N,lambda_plus,lambda_moins,lambdaN,phiP,phiF_plus,
304             phiA_plus,phiC,phiT,longueur);
305           event=K1["event"]; }
306         else if (population == 'C') {
307           K1=EvolC(K0,N,lambda_moins,lambdaN,phiP,phiA_plus,phiT,
308             longueur);
309           event=K1["event"]; }
310         else if (population == 'F') {
311           K1=EvolF(K0,N,lambda_moins,lambdaF_plus,lambdaN,phiP,
312             phiF_moins,phiT,longueur);
313           event=K1["event"]; }
314         else if (population == 'B') {
315           K1=EvolB(K0,N,lambda_plus,lambda_moins,lambdaN,phiP,phiA_plus,
316             phiA_moins,phiC,phiT,longueur,extremite,branchement);
317           event=K1["event"]; } } }
318
319       //On tire un temps exponentiel correspondant au temps auquel
320       //l'évenement a eu lieu
321       T1 = T0+rand_exp(taux_total_pop);
322       printf("%f\n", T1);
323       PrintT=floor(T1);
324       if( PrintT%10==0 && LastPrint!=PrintT) {
325         printf("%f \n",T1);
326         LastPrint=PrintT; }
327
328       K0 = K1;
329
330       for(it1=K1.begin(); it1!=K1.end(); ++it1) {
331         if (it1->second == 0 && it1->first.c_str()[0]!='M'
332         && it1->first.c_str()[0]!='P' ) {
333           K0.erase(it1->first.c_str()); } }
334
335       //On écrit les proportions initiales dans le fichier proportion
336       fprintf(outputprop, "%f ; ", T1);
337       N0=K1["M;0;1;0;"];//N0 = nb mono libres
338       N2=K1["P;0;1;0;"];//N2 = nb complexes
339
340       N1=0; //N1 = nb de mono dans filaments simples
341       N3=0; //N3 = nb de mono dans filaments formine
342       N4=0; //N4 = nb de mono dans filaments lineaire cappe
343       N5=0; //N5 = nb de mono dans filaments branchants
344
345       NbFilS=0;
346       NbFilF=0;
347       NbFilC=0;
348       NbFilB=0;
349
350       for (int i=0; i<N+1; i++) {
351         DistS[i]=0;
352         DistF[i]=0;
```

```
353           DistC[i]=0;
354           DistB[i]=0;
355           Bran[i]=0;
356           Ext[i]=0;
357           ExtB[i]=0; }
358
359         MoyBranch=0.0;
360         MoyExt=0.0;
361
362       for(it=K0.begin(); it!=K0.end(); ++it){
363          NL=findL(it->first.c_str());
364          BL=findB(it->first.c_str());
365          EL=findE(it->first.c_str());
366          if (it->first.c_str()[0]!='M' && it->first.c_str()[0]!='P'
367          && it->first.c_str()[0]!='e') {
368             Ext[EL]=Ext[EL]+it->second;
369             Bran[BL]=Bran[BL]+it->second; }
370          if(it->first.c_str()[0]=='S') {
371             N1=N1+it->second*NL;
372             NbFilS=NbFilS+it->second;
373             DistS[NL]=it->second; }
374          else if(it->first.c_str()[0]=='F') {
375             N3=N3+it->second*NL;
376             NbFilF=NbFilF+it->second;
377             DistF[NL]=it->second; }
378          else if(it->first.c_str()[0]=='C') {
379             N4=N4+it->second*NL;
380             NbFilC=NbFilC+it->second;
381             DistC[NL]=it->second; }
382          else if(it->first.c_str()[0]=='B') {
383             N5=N5+it->second*NL;
384             NbFilB=NbFilB+it->second;
385             DistB[NL]=it->second;
386             ExtB[EL]=ExtB[EL]+it->second; } }
387
388         if (NbFilB == 0) {
389           MoyBranch=0.0;
390           MoyExt=0.0; }
391         else {
392           for (int j=1; j<N+1; j++) {
393             MoyBranch=MoyBranch+j*Bran[j];
394             MoyExt=MoyExt+j*ExtB[j]; }
395
396           MoyBranch=MoyBranch/double(NbFilB);
397           MoyExt=MoyExt/double(NbFilB); }
398
399          fprintf(outputprop, "%d ; ", N0);
400          fprintf(outputprop, "%d ; ", N1);
401          fprintf(outputprop, "%d ; ", N2);
402          fprintf(outputprop, "%d ; ", N3);
403          fprintf(outputprop, "%d ; ", N4);
404          fprintf(outputprop, "%d ; ", N5);
405          fprintf(outputprop, "%d ; ",N0+N1+N2+N3+N4+N5);
406
407          fprintf(outputprop, "%d ; ", NbFilS);
408          fprintf(outputprop, "%d ; ", NbFilF);
409          fprintf(outputprop, "%d ; ", NbFilC);
410          fprintf(outputprop, "%d ;\n ", NbFilB);
411
412          PrintT2=floor(T1);
413
```

```
414         if( (PrintT2%100==0 && LastPrint2!=PrintT2) || T1>=tmax) {
415           fprintf(outputS, "%f ; ", T1);
416           fprintf(outputF, "%f ; ", T1);
417           fprintf(outputC, "%f ; ", T1);
418           fprintf(outputB, "%f ; ", T1);
419
420           fprintf(outputBranche, "%f ; ", T1);
421           fprintf(outputExtremite, "%f ; ", T1);
422
423           for (int i=0; i<N+1; i++) {
424             fprintf(outputS, "%d ; ", DistS[i]);
425             fprintf(outputF, "%d ; ", DistF[i]);
426             fprintf(outputC, "%d ; ", DistC[i]);
427             fprintf(outputB, "%d ; ", DistB[i]);
428             fprintf(outputBranche, "%d ; ", Bran[i]);
429             fprintf(outputExtremite, "%d ; ", Ext[i]); }
430
431           fprintf(outputS, "\n");
432           fprintf(outputF, "\n");
433           fprintf(outputC, "\n");
434           fprintf(outputB, "\n");
435
436           fprintf(outputBranche, "\n");
437           fprintf(outputExtremite, "\n");
438
439           LastPrint2=PrintT2; }
440
441          fprintf(outputMoy, "%f ; ", T1);
442          fprintf(outputMoy, "%f ; ", MoyBranch);
443          fprintf(outputMoy, "%f ;\n ", MoyExt);
444
445         //On réinitialise le tableau
446         T0 = T1;
447
448         //On repart pour un nouvel évenement
449         event = 0;
450         K0["event"]=0; } }
451     //On ferme le fichier résultat
452     fclose(outputS);
453     fclose(outputF);
454     fclose(outputC);
455     fclose(outputB);
456     fclose(outputBranche);
457     fclose(outputExtremite);
458     fclose(outputMoy);
459     fclose(outputprop);
460
461     return K1; }
```

1.2 - Biological Applications

Fonction C++ prenant en arguments :

* int N : le nombre total de monomères dans le système
* double tmax : le temps final
* double dt1 : la durée pendant laquelle le système est soumis au premier jeu de paramètres

- double dt2 : la durée pendant laquelle le système est soumis au second jeu de paramètres
- int n1 : le nombre de monomères inclus dans les filaments simples
- int n2 : le nombre de complexes actine-G/profiline
- int n3 : le nombre de monomères inclus dans les filaments associés à une formine
- int n4 : le nombre de monomères inclus dans les filaments linéaires cappés
- int n5 : le nombre de monomères inclus dans les filaments branchés
- int l0 : la taille initiale des filaments
- int b0 : le nombre initial de branchements (pour les filaments branchés)
- int e0 : le nombre intial d'extrémités libres (pour les filaments branchés)
- double lambda_plus1 : le taux d'élongation spontanée du premier jeu de paramètres
- double lambda_moins1 : le taux de dépolymérisation du premier jeu de paramètres
- double lambdaF_plus1 : le taux d'élongation avec une formine du premier jeu de paramètres
- double lambdaN1 : le taux de nucléation du premier jeu de paramètres
- double phiP1 : le taux de création de complexes actin-G/profilin du premier jeu de paramètres
- double phiF_plus1 : le taux de fixation d'une formine sur un polymère du premier jeu de paramètres
- double phiF_moins1 : le taux de libération d'une formine du premier jeu de paramètres
- double phiA_plus1 : le taux de fixation d'un Arp2/3 sur un polymère du premier jeu de paramètres
- double phiA_moins1 : le taux de libération d'un Arp2/3 du premier jeu de paramètres
- double phiC1 : le taux de capping du premier jeu de paramètres
- double phiT1 : le taux de fragmentation du premier jeu de paramètres
- double lambda_plus2 : le taux d'élongation spontanée du second jeu de paramètres
- double lambda_moins2 : le taux de dépolymérisation du second jeu de paramètres
- double lambdaF_plus2 : le taux d'élongation avec une formine du second jeu de paramètres
- double lambdaN2 : le taux de nucléation du second jeu de paramètres
- double phiP2 : le taux de création de complexes actin-G/profilin du second jeu de paramètres
- double phiF_plus2 : le taux de fixation d'une formine sur un polymère du second jeu de paramètres
- double phiF_moins2 : le taux de libération d'une formine du second jeu de paramètres
- double phiA_plus2 : le taux de fixation d'un Arp2/3 sur un polymère du second jeu de paramètres
- double phiA_moins2 : le taux de libération d'un Arp2/3 du second jeu de paramètres
- double phiC2 : le taux de capping du second jeu de paramètres
- double phiT2 : le taux de fragmentation du second jeu de paramètres

créant une map contenant l'état initial souhaité et modifiant cette map selon le modèle décrit dans la Section 4.2 du Chapitre III jusqu'au temps tmax en oscillant entre le premier et le second jeu de paramètres.

```
void SwitchParametres(
  int N, double tmax, double dt1, double dt2,
  int n1,int n2,int n3,int n4,int n5,int l0,int b0,int e0,
  double lambda_plus1, double lambda_moins1, double lambdaF_plus1,
  double lambdaN1,double phiP1, double phiF_plus1, double phiF_moins1,
  double phiA_plus1, double phiA_moins1, double phiC1, double phiT1,
  double lambda_plus2, double lambda_moins2, double lambdaF_plus2,
  double lambdaN2, double phiP2, double phiF_plus2, double phiF_moins2,
  double phiA_plus2, double phiA_moins2, double phiC2, double phiT2) {
    int cpt=0;
    double t0=0.0;
    double T=0.0;

    std::map<std::string,int> K0 =
    GeneInitMapParam(N,n1,n2,n3,n4,n5,l0,b0,e0,cpt);
    std::map<std::string,int> K1 = PopulationTotaleMap(K0,t0,N,t0+dt1,
      lambda_plus1,lambda_moins1,lambdaF_plus1,lambdaN1,
      phiP1, phiF_plus1,phiF_moins1,phiA_plus1,phiA_moins1,phiC1,phiT1,cpt);

    T=t0+dt1;
    t0=T;
    cpt=cpt+1;
    std::map<std::string,int> K2;
    while (T<tmax) {
        K2 = PopulationTotaleMap(K1,t0,N,t0+dt2,
        lambda_plus2,lambda_moins2,lambdaF_plus2,lambdaN2,
        phiP2, phiF_plus2,phiF_moins2,phiA_plus2,phiA_moins2,phiC2,phiT2,cpt);

        T=t0+dt2;
        t0=T;
        cpt=cpt+1;

      if (T < tmax) {
          K1 = PopulationTotaleMap(K2,t0,N,t0+dt1,
          lambda_plus1,lambda_moins1,lambdaF_plus1,lambdaN1,
          phiP1, phiF_plus1,phiF_moins1,phiA_plus1,phiA_moins1,phiC1,phiT1,cpt);

          T=t0+dt1;
          t0=T;
          cpt=cpt+1; } } }
```

Fonction C++ prenant en arguments :

- int N : le nombre total de monomères dans le système
- double tmax : le temps final
- double dt1 : la durée pendant laquelle le système est soumis au taux de fixation de formines et au taux de capping normaux
- double dt2 : la durée pendant laquelle le système est soumis au taux de fixation de formines et au taux de capping modifiés
- double retard : la durée pendant laquelle le taux de fixation de formines est déjà modifié alors que le taux de capping est normal
- int n1 : le nombre de monomères inclus dans les filaments simples
- int n2 : le nombre de complexes actine-G/profiline
- int n3 : le nombre de monomères inclus dans les filaments associés à une formine

- int n4 : le nombre de monomères inclus dans les filaments linéaires cappés
- int n5 : le nombre de monomères inclus dans les filaments branchés
- int l0 : la taille initiale des filaments
- int b0 : le nombre initial de branchements (pour les filaments branchés)
- int e0 : le nombre intial d'extrémités libres (pour les filaments branchés)
- double lambda_plus : le taux d'élongation spontanée
- double lambda_moins : le taux de dépolymérisation
- double lambdaF_plus : le taux d'élongation avec une formine
- double lambdaN : le taux de nucléation
- double phiP : le taux de création de complexes actin-G/profilin du premier jeu de paramètres
- double phiF_plus1 : le taux de fixation d'une formine normal
- double phiF_plus2 : le taux de fixation d'une formine modifié
- double phiF_moins : le taux de libération d'une formine
- double phiA_plus : le taux de fixation d'un Arp2/3 sur un polymère
- double phiA_moins : le taux de libération d'un Arp2/3
- double phiC1 : le taux de capping normal
- double phiC2 : le taux de capping modifié
- double phiT : le taux de fragmentation

créant une map contenant l'état initial souhaité et modifiant cette map selon le modèle décrit dans la Section 4.2 du Chapitre III jusqu'au temps tmax en oscillant entre deux valeurs de Φ_F^+ et deux valeurs de Φ_C.

```
void SwitchParametres2(
  int N, double tmax, double dt1, double dt2, double retard,
  int n1,int n2,int n3,int n4,int n5,int l0,int b0,int e0,
  double lambda_plus, double lambda_moins, double lambdaF_plus,
  double lambdaN, double phiP, double phiF_plus1, double phiF_plus2,
  double phiF_moins,double phiA_plus, double phiA_moins, double phiC1,
  double phiC2, double phiT) {
    int cpt=0;
    double t0=0.0;
    double T=0.0;

    std::map<std::string,int> K0 = GeneInitMapParam(N,n1,n2,n3,n4,n5,
      l0,b0,e0,cpt);
    std::map<std::string,int> K1 = PopulationTotaleMap(K0,t0,N,t0+dt1,
      lambda_plus,lambda_moins,lambdaF_plus,lambdaN,
      phiP, phiF_plus1,phiF_moins,phiA_plus,phiA_moins,phiC1,phiT,cpt);

    T=t0+dt1;
    t0=T;
    cpt=cpt+1;
    std::map<std::string,int> K2;
    std::map<std::string,int> K3;
    std::map<std::string,int> K4;
```

```
24      while (T<tmax) {
25          K2 = PopulationTotaleMap(K1,t0,N,t0+retard,
26          lambda_plus,lambda_moins,lambdaF_plus,lambdaN,
27          phiP, phiF_plus2,phiF_moins,phiA_plus,phiA_moins,phiC1,phiT,cpt);
28
29          T=t0+retard;
30          t0=T;
31          cpt=cpt+1;
32
33          K3 = PopulationTotaleMap(K2,t0,N,t0+dt2-retard,
34          lambda_plus,lambda_moins,lambdaF_plus,lambdaN,
35          phiP, phiF_plus2,phiF_moins,phiA_plus,phiA_moins,phiC2,phiT,cpt);
36
37          T=t0+dt2-retard;
38          t0=T;
39          cpt=cpt+1;
40
41          K4 = PopulationTotaleMap(K3,t0,N,t0+retard,
42          lambda_plus,lambda_moins,lambdaF_plus,lambdaN,
43          phiP, phiF_plus1,phiF_moins,phiA_plus,phiA_moins,phiC2,phiT,cpt);
44
45          T=t0+retard;
46          t0=T;
47          cpt=cpt+1;
48
49        if (T < tmax) {
50            K1 = PopulationTotaleMap(K4,t0,N,t0+dt1,
51            lambda_plus,lambda_moins,lambdaF_plus,lambdaN,
52            phiP, phiF_plus1,phiF_moins,phiA_plus,phiA_moins,phiC1,phiT,cpt);
53
54            T=t0+dt1;
55            t0=T;
56            cpt=cpt+1; } } }
```

1.3 - Large Population Limit

Fonction C++ prenant en arguments :

- int lmax : la longueur maximale des polymères à considérer
- double p1 : la proportion de monomères inclus dans les filaments simples
- double p2 : la proportion de complexes actine-G/profiline
- double p3 : la proportion de monomères inclus dans les filaments associés à une formine
- double p4 : la proportion de monomères inclus dans les filaments linéaires cappés
- double p5 : la proportion de monomères inclus dans les filaments branchés
- int l0 : la taille initiale des filaments
- int b0 : le nombre initial de branchements (pour les filaments branchés)
- int e0 : le nombre intial d'extrémités libres (pour les filaments branchés)
- int vi : entier permettant de numéroter le fichier résultat

et créant une map contenant l'état initial souhaité.

```cpp
std::map<std::string,double> GeneInitMapLimiteParam(int lmax,
  double p1,double p2,double p3,double p4,double p5,
  int l0,int b0,int e0,int vi) {
  std::map<std::string,double> Klim0;

  std::string sl0 = std::to_string(l0);
  std::string sb0 = std::to_string(b0);
  std::string se0 = std::to_string(e0);

  std::string svi = std::to_string(vi);

  if (1-p1-p2-p3-p4-p5>0) {
    Klim0["M;0;1;0;"]=1-p1-p2-p3-p4-p5; }
  if (p1>0) {
    Klim0["S;0;"+sl0+";1;"]=p1/double(l0); }
  if (p2>0) {
    Klim0["P;0;1;0;"]=p2; }
  if (p3>0) {
    Klim0["F;0;"+sl0+";1;"]=p3/double(l0); }
  if (p4>0) {
    Klim0["C;0;"+sl0+";0;"]=p4/double(l0); }
  if (p5>0) {
    Klim0["B;"+sb0+";"+sl0+";"+se0+";"]=p5/double(l0); }

  std::string NomFichier = "ValeursInitialesLimite"+svi+".txt";

  FILE* outInit = fopen(NomFichier.c_str(),"w");

  fprintf(outInit, "Libres = %f \n", (1-p1-p2-p3-p4-p5));
  fprintf(outInit, "Simples = %f \n", p1);
  fprintf(outInit, "Complexes = %f \n", p2);
  fprintf(outInit, "Formine = %f \n", p3);
  fprintf(outInit, "Cappes = %f \n", p4);
  fprintf(outInit, "Branchants = %f \n", p5);
  fprintf(outInit, "Longueur = %d \n", l0);
  if (p5>0) {
    fprintf(outInit, "Branchements = %d \n", b0);
    fprintf(outInit, "Extremites = %d \n", e0); }
  fclose(outInit);

  return Klim0; }
```

Fonction C++ prenant en arguments :

- std::map<std::string, int> K0 : map initiale obtenue grâce à la fonction GeneInitMapLimiteParam
- double t0 : le temps initial
- int lmax : la longueur maximale des polymères à considérer
- double tmax : le temps final
- double dt : le pas de temps pour la méthode d'Euler explicite
- double lambda_plus : le taux d'élongation spontanée
- double lambda_moins : le taux de dépolymérisation
- double lambdaF_plus : le taux d'élongation avec une formine
- double lambdaN : le taux de nucléation

- double phiP : le taux de création de complexes actin-G/profilin
- double phiF_plus : le taux de fixation d'une formine sur un polymère
- double phiF_moins : le taux de libération d'une formine
- double phiA_plus : le taux de fixation d'un Arp2/3 sur un polymère
- double phiA_moins : le taux de libération d'un Arp2/3
- double phiC : le taux de capping
- double phiT : le taux de fragmentation
- int vi : entier permettant de numéroter le fichier résultat

et modifiant la map selon le système d'équations différentielles ordinaires de la Conjecture 3.4.1.

```
std::map<std::string,double> PopulationTotaleMapLimite
(std::map<std::string,double> Klim0,double t0,int lmax, double tmax, double dt,
  double lambda_plus, double lambda_moins, double lambdaF_plus,
  double lambdaN, double phiP, double phiF_plus, double phiF_moins,
  double phiA_plus, double phiA_moins, double phiC, double phiT,int vi) {
  std::string svi = std::to_string(vi);

  std::string NomFichier = "ValeursParametresLimite"+svi+".txt";
  //On crée un fichier pour récupérer la valeur des paramètres
  FILE* outparametre = fopen(NomFichier.c_str(),"w");

  //On écrit la valeur des paramètres
  fprintf(outparametre,"dt = %f", dt);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lambda_plus = %f", lambda_plus);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lambdaF_plus = %f", lambdaF_plus);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lambda_moins = %f", lambda_moins);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lambdaN = %f", lambdaN);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiP = %f", phiP);
  fprintf(outparametre, "\n");
  fprintf(outparametre, "PhiF_plus = %f", phiF_plus);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiF_moins = %f", phiF_moins);
  fprintf(outparametre, "\n");
  fprintf(outparametre, "PhiA_plus = %f", phiA_plus);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiA_moins = %f", phiA_moins);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiC = %f", phiC);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"PhiT = %f", phiT);
  fprintf(outparametre, "\n");
  fprintf(outparametre,"lmax = %i", lmax);
  fprintf(outparametre, "\n");

  //On ferme le fichier pour les paramètres
  fclose(outparametre);

  //On crée un fichier pour l'évolution du système
```

```
44  NomFichier = "DistributionSlim"+svi+".txt";
45  FILE* outputS = fopen(NomFichier.c_str(),"w");
46  NomFichier = "DistributionFlim"+svi+".txt";
47  FILE* outputF = fopen(NomFichier.c_str(),"w");
48  NomFichier = "DistributionClim"+svi+".txt";
49  FILE* outputC = fopen(NomFichier.c_str(),"w");
50  NomFichier = "DistributionBlim"+svi+".txt";
51  FILE* outputB = fopen(NomFichier.c_str(),"w");
52
53  NomFichier = "MoyennesBElim"+svi+".txt";
54  FILE* outputMoy = fopen(NomFichier.c_str(),"w");
55
56  NomFichier = "DistributionBranchelim"+svi+".txt";
57  FILE* outputBranche = fopen(NomFichier.c_str(),"w");
58  NomFichier = "DistributionExtremitelim"+svi+".txt";
59  FILE* outputExtremite = fopen(NomFichier.c_str(),"w");
60  //et un fichier pour les prortions de filaments dans
61  //chaque sous-population
62  NomFichier = "Proportionslim"+svi+".txt";
63  FILE* outputprop = fopen(NomFichier.c_str(),"w");
64
65  //On initialise le temps
66  double t=t0;
67  printf("%f\n", t);
68  int cpt=0;
69  int next_print=1;
70
71  std::map<std::string, double>::iterator it;
72  //On initialise les proportions de monomeres presents
73  //dans chaque sous-population
74  double p0=Klim0["M;0;1;0;"];    //monomères libres
75  double p1=0;    //filaments simples
76  double p2=Klim0["P;0;1;0;"];    //complexes G-actin/proilin
77  double p3=0;    //filaments+formin
78  double p4=0;    //filaments linéaires cappés
79  double p5=0;    //filaments branchés
80
81  int NL; //Longueur en cours
82  int BL; //Branchement en cours
83  int EL; //Extremite en cours
84
85  double pFilS=0; //Prop de filaments simples
86  double pFilF=0; //Prop de filaments formine
87  double pFilC=0; //Prop de filaments cappé
88  double pFilB=0; //Prop de filaments branchants
89
90  double DistSlim[lmax+1]; //Dans case i,
91  //prop de filaments simples de taille i
92  double DistFlim[lmax+1]; //Dans case i,
93  //prop de filaments formine de taille i
94  double DistClim[lmax+1]; //Dans case i,
95  //prop de filaments cappés de taille i
96  double DistBlim[lmax+1]; //Dans case i,
97  //prop de filaments branchants de taille i
98
99  double pBran[lmax+1]; //Dans case i,
100 //prop de filaments avec i branchements
101 double pExt[lmax+1]; //Dans case i,
102 //prop de filaments avec i extremites libres
103 double pExtB[lmax+1]; //Dans case i,
104 //prop de filaments branchants avec i extremites libres
```

```
105
106   for (int i=0; i<lmax+1; i++) {
107     DistSlim[i]=0;
108     DistFlim[i]=0;
109     DistClim[i]=0;
110     DistBlim[i]=0;
111     pBran[i]=0;
112     pExt[i]=0;
113     pExtB[i]=0; }
114
115   double MoyBranchLim=0.0;
116
117   for(it=Klim0.begin(); it!=Klim0.end(); ++it) {
118      NL=findL(it->first.c_str());
119      BL=findB(it->first.c_str());
120      EL=findE(it->first.c_str());
121      if (it->first.c_str()[0]!='M' && it->first.c_str()[0]!='P'
122      && it->first.c_str()[0]!='e') {
123        pExt[EL]=pExt[EL]+it->second;
124        pBran[BL]=pBran[BL]+it->second; }
125      if(it->first.c_str()[0]=='S') {
126        p1=p1+it->second*NL;
127        pFilS=pFilS+it->second;
128        DistSlim[NL]=it->second; }
129      else if(it->first.c_str()[0]=='F') {
130        p3=p3+it->second*NL;
131        pFilF=pFilF+it->second;
132        DistFlim[NL]=it->second; }
133      else if(it->first.c_str()[0]=='C') {
134        p4=p4+it->second*NL;
135        pFilC=pFilC+it->second;
136        DistClim[NL]=it->second; }
137      else if(it->first.c_str()[0]=='B') {
138        p5=p5+it->second*NL;
139        pFilB=pFilB+it->second;
140        DistBlim[NL]=it->second;
141        pExtB[EL]=pExtB[EL]+it->second; } }
142
143    if (pFilB == 0) {
144      MoyBranchLim=0.0;
145      MoyExtLim=0.0; }
146    else {
147      for (int j=1; j<lmax+1; j++) {
148        MoyBranchLim=MoyBranchLim+j*pBran[j];
149        MoyExtLim=MoyExtLim+j*pExtB[j]; }
150
151      MoyBranchLim=MoyBranchLim/double(pFilB);
152      MoyExtLim=MoyExtLim/double(pFilB); }
153
154    fprintf(outputprop, "%f ; ", t);
155    fprintf(outputprop, "%f ; ", p0);
156    fprintf(outputprop, "%f ; ", p1);
157    fprintf(outputprop, "%f ; ", p2);
158    fprintf(outputprop, "%f ; ", p3);
159    fprintf(outputprop, "%f ; ", p4);
160    fprintf(outputprop, "%f ; ", p5);
161    fprintf(outputprop, "%f ;",p0+p1+p2+p3+p4+p5);
162    fflush(outputprop);
163
164    fprintf(outputprop, "%f ; ", pFilS);
165    fprintf(outputprop, "%f ; ", pFilF);
```

```
166     fprintf(outputprop, "%f ; ", pFilC);
167     fprintf(outputprop, "%f ;\n ", pFilB);
168 
169     fprintf(outputS, "%f ; ", t);
170     fprintf(outputF, "%f ; ", t);
171     fprintf(outputC, "%f ; ", t);
172     fprintf(outputB, "%f ; ", t);
173 
174     fprintf(outputBranche, "%f ; ", t);
175     fprintf(outputExtremite, "%f ; ", t);
176 
177     for (int i=0; i<lmax+1; i++) {
178       fprintf(outputS, "%f ; ", DistSlim[i]);
179       fprintf(outputF, "%f ; ", DistFlim[i]);
180       fprintf(outputC, "%f ; ", DistClim[i]);
181       fprintf(outputB, "%f ; ", DistBlim[i]);
182       fprintf(outputBranche, "%f ; ", pBran[i]);
183       fprintf(outputExtremite, "%f ; ", pExt[i]); }
184 
185     fprintf(outputS, "\n");
186     fprintf(outputF, "\n");
187     fprintf(outputC, "\n");
188     fprintf(outputB, "\n");
189 
190     fprintf(outputBranche, "\n");
191     fprintf(outputExtremite, "\n");
192 
193     fprintf(outputMoy, "%f ; ", t);
194     fprintf(outputMoy, "%f ; ", MoyBranchLim);
195     fprintf(outputMoy, "%f ;\n ", MoyExtLim);
196 
197      //On initialise l'etat suivant du systeme
198     std::map<std::string,double> Klim1;
199     std::map<std::string, double>::iterator it1;
200 
201     double creation_complexe;
202     double nucleation;
203     double mapP=0;
204     double mapM=0;
205     std::map<int,int> mapSLong; //Longueurs existantes
206     //pour les filaments simples
207     std::map<int,int>::iterator itSlong;
208     std::map<int,int> mapFLong; //Longueurs existantes
209     //pour les filaments formine
210     std::map<int,int>::iterator itFlong;
211     std::map<int,int> mapCLong; //Longueurs existantes
212     //pour les filaments cappes
213     std::map<int,int>::iterator itClong;
214     std::map<std::string,std::string> mapAType; //Types existants
215     //pour les filaments branché
216     std::map<std::string,std::string>::iterator itAtype;
217     std::map<int,double> mapS; //taux total de changement
218     //pour les filaments simples
219     std::map<int, double>::iterator itS;
220     std::map<int,double> mapF; //taux total de changement
221     //pour les filaments formine
222     std::map<int, double>::iterator itF;
223     std::map<int,double> mapC; //taux total de changement
224     //pour les filaments cappés
225     std::map<int, double>::iterator itC;
226     std::map<std::string,double> mapA; //taux total de changement
```

```
227   //pour les filaments arp
228   std::map<std::string, double>::iterator itA;
229   std::string sL;
230   std::string sLm;
231   std::string sLp;
232   std::string sB;
233   std::string sE;
234   std::string sBp;
235   std::string sEp;
236   std::string sBm;
237   std::string sEm;
238
239   std::string sl;
240   std::string sb;
241   std::string se;
242
243   double sp;
244
245   //Tant qu'on a pas atteint le tmax,
246   //on continue a faire evoluer le systeme
247   while(t<tmax) {
248     sp=0.0;
249
250     //On approche la solution du systeme au temps dt par
251     //une methode d'Euler explicite
252     //Le systme d'EDO s'ecrit :
253     //Klim1[]=dt*()+Klim0[];
254
255      std::map<std::string, double>::iterator ite;
256
257     //on remplit Klim1
258     creation_complexe = phiP*Klim0["M;0;1;0;"];
259     nucleation = 3*lambdaN*Klim0["M;0;1;0;"]
260     *Klim0["M;0;1;0;"]*Klim0["M;0;1;0;"];
261
262    mapM=mapM-creation_complexe-nucleation;
263    mapP=mapP+creation_complexe;
264    mapS[3]=mapS[3]+nucleation;
265
266    for(it=Klim0.begin(); it!=Klim0.end(); ++it) {
267      if (it->first.c_str()[0]=='S') {
268        NL=findL(it->first.c_str());
269        mapSLong[NL]=NL; }
270      if (it->first.c_str()[0]=='F') {
271        NL=findL(it->first.c_str());
272        mapFLong[NL]=NL; }
273      if (it->first.c_str()[0]=='B') {
274        mapAType[it->first.c_str()]=it->first.c_str(); }
275      if (it->first.c_str()[0]=='C') {
276        NL=findL(it->first.c_str());
277        mapCLong[NL]=NL; } }
278
279    for(itSlong=mapSLong.begin();itSlong!=mapSLong.end();++itSlong) {
280      sL = std::to_string(itSlong->second);
281      sLm = std::to_string(itSlong->second-1);
282      sLp = std::to_string(itSlong->second+1);
283      if (itSlong->second == 3){
284        mapS[itSlong->second]=mapS[itSlong->second]
285        +lambda_moins*Klim0["S;0;"+sLp+";1;"]
286        -lambda_plus*Klim0["M;0;1;0;"]*Klim0["S;0;"+sL+";1;"]
287        -lambda_moins*Klim0["S;0;"+sL+";1;"]
```

```
288           -phiF_plus*Klim0["S;0;"+sL+";1;"]
289           -phiA_plus*Klim0["S;0;"+sL+";1;"]*3
290           -phiC*Klim0["S;0;"+sL+";1;"]
291           -phiT*Klim0["S;0;"+sL+";1;"]*itSlong->second*itSlong->second;
292           mapM=mapM-lambda_plus*Klim0["M;0;1;0;"]*Klim0["S;0;"+sL+";1;"]
293           +3*lambda_moins*Klim0["S;0;"+sL+";1;"]
294           +3*phiT*Klim0["S;0;"+sL+";1;"]*itSlong->second*itSlong->second; }
295         else {
296           mapS[itSlong->second]=mapS[itSlong->second]
297           +lambda_plus*Klim0["M;0;1;0;"]*Klim0["S;0;"+sLm+";1;"]
298           +lambda_moins*Klim0["S;0;"+sLp+";1;"]
299           -lambda_plus*Klim0["M;0;1;0;"]*Klim0["S;0;"+sL+";1;"]
300           -lambda_moins*Klim0["S;0;"+sL+";1;"]
301           -phiF_plus*Klim0["S;0;"+sL+";1;"]
302           -phiA_plus*Klim0["S;0;"+sL+";1;"]*itSlong->second
303           -phiC*Klim0["S;0;"+sL+";1;"]
304           -phiT*Klim0["S;0;"+sL+";1;"]*itSlong->second*itSlong->second;
305           mapM=mapM-lambda_plus*Klim0["M;0;1;0;"]*Klim0["S;0;"+sL+";1;"]
306           +lambda_moins*Klim0["S;0;"+sL+";1;"];
307           mapS[itSlong->second-1]=mapS[itSlong->second-1]
308           +lambda_moins*Klim0["S;0;"+sL+";1;"];
309
310           for(int l=3; l<itSlong->second; l++){
311             mapS[l]=mapS[l]
312             +phiT*Klim0["S;0;"+sL+";1;"]*itSlong->second*itSlong->second
313             *ProbaGeom(itSlong->second-4,fmax(0,
314               1-(lambda_moins/(lambda_plus*Klim0["M;0;1;0;"]))),l-3);
315             mapM=mapM
316             +phiT*Klim0["S;0;"+sL+";1;"]*itSlong->second*itSlong->second
317             *ProbaGeom(itSlong->second-4,fmax(0,
318               1-(lambda_moins/(lambda_plus*Klim0["M;0;1;0;"]))),l-3)
319               *(itSlong->second-l); } }
320         mapS[itSlong->second+1]=mapS[itSlong->second+1]
321         +lambda_plus*Klim0["M;0;1;0;"]*Klim0["S;0;"+sL+";1;"];
322         mapF[itSlong->second]=mapF[itSlong->second]
323         +phiF_plus*Klim0["S;0;"+sL+";1;"];
324         mapC[itSlong->second]=mapC[itSlong->second]
325         +phiC*Klim0["S;0;"+sL+";1;"];
326         mapA["B;1;"+sL+";2;"]=mapA["B;1;"+sL+";2;"]
327         +phiA_plus*Klim0["S;0;"+sL+";1;"]*itSlong->second; }
328
329       for(itFlong=mapFLong.begin(); itFlong!=mapFLong.end(); ++itFlong) {
330         sL = std::to_string(itFlong->second);
331         sLm = std::to_string(itFlong->second-1);
332         sLp = std::to_string(itFlong->second+1);
333         if (itFlong->second == 3){
334           mapF[itFlong->second]=mapF[itFlong->second]
335           +lambda_moins*Klim0["F;0;"+sLp+";1;"]
336           -lambdaF_plus*Klim0["P;0;1;0;"]*Klim0["F;0;"+sL+";1;"]
337           -lambda_moins*Klim0["F;0;"+sL+";1;"]
338           -phiF_moins*Klim0["F;0;"+sL+";1;"]
339           -phiT*Klim0["F;0;"+sL+";1;"]*itFlong->second*itFlong->second;
340           mapP=mapP-lambdaF_plus*Klim0["P;0;1;0;"]*Klim0["F;0;"+sL+";1;"];
341           mapM=mapM+3*lambda_moins*Klim0["F;0;"+sL+";1;"]
342           +3*phiT*Klim0["F;0;"+sL+";1;"]*itFlong->second*itFlong->second; }
343         else {
344           mapF[itFlong->second]=mapF[itFlong->second]
345           +lambdaF_plus*Klim0["P;0;1;0;"]*Klim0["F;0;"+sLm+";1;"]
346           +lambda_moins*Klim0["F;0;"+sLp+";1;"]
347           -lambdaF_plus*Klim0["P;0;1;0;"]*Klim0["F;0;"+sL+";1;"]
348           -lambda_moins*Klim0["F;0;"+sL+";1;"]
```

```
349         -phiF_moins*Klim0["F;0;"+sL+";1;"]
350         -phiT*Klim0["F;0;"+sL+";1;"]*itFlong->second*itFlong->second;
351         mapP=mapP-lambdaF_plus*Klim0["P;0;1;0;"]*Klim0["F;0;"+sL+";1;"];
352         mapM=mapM+lambda_moins*Klim0["F;0;"+sL+";1;"];
353         mapF[itFlong->second-1]=mapF[itFlong->second-1]
354         +lambda_moins*Klim0["F;0;"+sL+";1;"];
355
356         for(int l=3; l<itFlong->second; l++) {
357           mapF[l]=mapF[l]
358           +phiT*Klim0["F;0;"+sL+";1;"]*itFlong->second*itFlong->second
359           *ProbaGeom(itFlong->second-4,fmax(0,
360             1-(lambda_moins/(lambdaF_plus*Klim0["P;0;1;0;"]))),l-3);
361           mapM=mapM
362           +phiT*Klim0["F;0;"+sL+";1;"]*itFlong->second*itFlong->second
363           *ProbaGeom(itFlong->second-4,fmax(0,
364             1-(lambda_moins/(lambdaF_plus*Klim0["P;0;1;0;"]))),l-3)
365             *(itFlong->second-l); } }
366       mapF[itFlong->second+1]=mapF[itFlong->second+1]
367       +lambdaF_plus*Klim0["P;0;1;0;"]*Klim0["F;0;"+sL+";1;"];
368       mapS[itFlong->second]=mapS[itFlong->second]
369       +phiF_moins*Klim0["F;0;"+sL+";1;"]; }
370
371     for(itClong=mapCLong.begin(); itClong!=mapCLong.end(); ++itClong) {
372       sL = std::to_string(itClong->second);
373       sLm = std::to_string(itClong->second-1);
374       sLp = std::to_string(itClong->second+1);
375       if (itClong->second == 3){
376         mapC[itClong->second]=mapC[itClong->second]
377         +lambda_moins*Klim0["C;0;"+sLp+";0;"]
378         -lambda_moins*Klim0["C;0;"+sL+";0;"]
379         -phiA_plus*Klim0["C;0;"+sL+";0;"]*itClong->second
380         -phiT*Klim0["C;0;"+sL+";0;"]*itClong->second*itClong->second;
381         mapM=mapM+3*lambda_moins*Klim0["C;0;"+sL+";0;"]
382         +3*phiT*Klim0["C;0;"+sL+";0;"]*itClong->second*itClong->second; }
383       else {
384         mapC[itClong->second]=mapC[itClong->second]
385         +lambda_moins*Klim0["C;0;"+sLp+";0;"]
386         -lambda_moins*Klim0["C;0;"+sL+";0;"]
387         -phiA_plus*Klim0["C;0;"+sL+";0;"]*itClong->second
388         -phiT*Klim0["C;0;"+sL+";0;"]*itClong->second*itClong->second;
389         mapM=mapM+lambda_moins*Klim0["C;0;"+sL+";0;"]
390         +itClong->second*phiT*Klim0["C;0;"+sL+";0;"]
391         *itClong->second*itClong->second;
392         mapC[itClong->second-1]=mapC[itClong->second-1]
393         +lambda_moins*Klim0["C;0;"+sL+";0;"]; }
394       mapA["B;1;"+sL+";1;"]=mapA["B;1;"+sL+";1;"]
395       +phiA_plus*Klim0["C;0;"+sL+";0;"]*itClong->second; }
396
397     for(itAtype=mapAType.begin(); itAtype!=mapAType.end(); ++itAtype) {
398       NL=findL(itAtype->first.c_str());
399       BL=findB(itAtype->first.c_str());
400       EL=findE(itAtype->first.c_str());
401       sL = std::to_string(NL);
402       sLm = std::to_string(NL-1);
403       sLp = std::to_string(NL+1);
404       sB = std::to_string(BL);
405       sBp = std::to_string(BL+1);
406       sBm = std::to_string(BL-1);
407       sE = std::to_string(EL);
408       sEp = std::to_string(EL+1);
409       sEm = std::to_string(EL-1);
```

```
410         if (NL == 3) {
411           if (BL>1 && EL>0) {
412             mapA[itAtype->first.c_str()]=mapA[itAtype->first.c_str()]
413             +lambda_moins*Klim0["B;"+sB+";"+sLp+";"+sE+";"]
414             +phiA_plus*Klim0["B;"+sBm+";"+sL+";"+sEm+";"]*(NL-(BL-1))
415             +phiC*Klim0["B;"+sB+";"+sL+";"+sEp+";"]*(EL+1)
416             -lambda_plus*Klim0["M;0;1;0;"]*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL
417             -lambda_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
418             -phiA_plus*Klim0["B;"+sB+";"+sL+";"+sE+";"]*(NL-BL)
419             -phiC*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL
420             -phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]*BL
421             -phiT*Klim0["B;"+sB+";"+sL+";"+sE+";"]*NL*NL; }
422           else {
423             mapA[itAtype->first.c_str()]=mapA[itAtype->first.c_str()]
424             +lambda_moins*Klim0["B;"+sB+";"+sLp+";"+sE+";"]
425             +phiC*Klim0["B;"+sB+";"+sL+";"+sEp+";"]*(EL+1)
426             -lambda_plus*Klim0["M;0;1;0;"]*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL
427             -lambda_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
428             -phiA_plus*Klim0["B;"+sB+";"+sL+";"+sE+";"]*(NL-BL)
429             -phiC*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL
430             -phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]*BL
431             -phiT*Klim0["B;"+sB+";"+sL+";"+sE+";"]*NL*NL; }
432           mapM=mapM
433           -lambda_plus*Klim0["M;0;1;0;"]*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL
434           +3*lambda_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]; }
435         else {
436           if (BL>1 && EL>0) {
437             mapA[itAtype->first.c_str()]=mapA[itAtype->first.c_str()]
438             +lambda_plus
439             *Klim0["M;0;1;0;"]*Klim0["B;"+sB+";"+sLm+";"+sE+";"]*EL
440             +lambda_moins*Klim0["B;"+sB+";"+sLp+";"+sE+";"]
441             +phiC*Klim0["B;"+sB+";"+sL+";"+sEp+";"]*(EL+1)
442             +phiA_plus*Klim0["B;"+sBm+";"+sL+";"+sEm+";"]*(NL-(BL-1))
443             -lambda_plus
444             *Klim0["M;0;1;0;"]*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL
445             -lambda_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
446             -phiA_plus*Klim0["B;"+sB+";"+sL+";"+sE+";"]*(NL-BL)
447             -phiC*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL
448             -phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]*BL
449             -phiT*Klim0["B;"+sB+";"+sL+";"+sE+";"]*NL*NL; }
450           else {
451             mapA[itAtype->first.c_str()]=mapA[itAtype->first.c_str()]
452             +lambda_plus
453             *Klim0["M;0;1;0;"]*Klim0["B;"+sB+";"+sLm+";"+sE+";"]*EL
454             +lambda_moins*Klim0["B;"+sB+";"+sLp+";"+sE+";"]
455             +phiC*Klim0["B;"+sB+";"+sL+";"+sEp+";"]*(EL+1)
456             -lambda_plus
457             *Klim0["M;0;1;0;"]*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL
458             -lambda_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
459             -phiA_plus*Klim0["B;"+sB+";"+sL+";"+sE+";"]*(NL-BL)
460             -phiC*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL
461             -phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]*BL
462             -phiT*Klim0["B;"+sB+";"+sL+";"+sE+";"]*NL*NL; }
463           mapM=mapM-lambda_plus
464           *Klim0["M;0;1;0;"]*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL
465           +lambda_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"];
466           mapA["B;"+sB+";"+sLm+";"+sE+";"]=mapA["B;"+sB+";"+sLm+";"+sE+";"]
467           +lambda_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
468           *(1-(BL/double(NL)));
469
470           for(int l=1; l<NL; l++) {
```

```
471             for (int b=0; b<=min_int(BL-1,l); b++) {
472               for (int e=0; e<=min_int(EL,b+1); e++) {
473                 sl = std::to_string(l);
474                 sb = std::to_string(b);
475                 se = std::to_string(e);
476                 if (l==1) {
477                   mapM=mapM+phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
478                   *BL*ProbaDepoSplit(l,NL,b,BL,e,EL); }
479                 else if (l==2) {
480                   mapM=mapM+2*phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
481                   *BL*ProbaDepoSplit(l,NL,b,BL,e,EL); }
482                 else {
483                   if (b==0) {
484                     if (e==0) {
485                       mapC[l]=mapC[l]
486                       +phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
487                       *BL*ProbaDepoSplit(l,NL,b,BL,e,EL); }
488                     else if (e==1) {
489                       mapS[l]=mapS[l]
490                       +phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
491                       *BL*ProbaDepoSplit(l,NL,b,BL,e,EL); } }
492                   else {
493                     mapA["B;"+sb+";"+sl+";"+se+";"]=
494                     mapA["B;"+sb+";"+sl+";"+se+";"]
495                     +phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
496                     *BL*ProbaDepoSplit(l,NL,b,BL,e,EL); } } } } } }
497           mapA["B;"+sB+";"+sLp+";"+sE+";"]=mapA["B;"+sB+";"+sLp+";"+sE+";"]
498           +lambda_plus*Klim0["M;0;1;0;"]*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL;
499           mapA["B;"+sBp+";"+sL+";"+sEp+";"]=mapA["B;"+sBp+";"+sL+";"+sEp+";"]
500           +phiA_plus*Klim0["B;"+sB+";"+sL+";"+sE+";"]*(NL-BL);
501           mapA["B;"+sB+";"+sL+";"+sEm+";"]=mapA["B;"+sB+";"+sL+";"+sEm+";"]
502           +phiC*Klim0["B;"+sB+";"+sL+";"+sE+";"]*EL;
503
504           //on cree un (b,l,e) a partir de notre (BL,NL,EL)
505           //si l=0 rien ne se passe
506           //si l=1 on gagne un monomère (qq soit b et e)
507           //si l=2 on gagne 2 monomère (qq soit b et e)
508           //si l=3 on gagne un filament dont le type dépend de b et e
509
510           for(int l=1; l<NL+1; l++) {
511             for (int b=0; b<=min_int(BL-1,l); b++) {
512               for (int e=0; e<=min_int(EL,b+1); e++) {
513                 sl = std::to_string(l);
514                 sb = std::to_string(b);
515                 se = std::to_string(e);
516                 if (l==1) {
517                   mapM=mapM+phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
518                   *BL*ProbaLibA(l,NL,b,BL,e,EL); }
519                 else if (l==2) {
520                   mapM=mapM+2*phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
521                   *BL*ProbaLibA(l,NL,b,BL,e,EL); }
522                 else {
523                   if (b==0) {
524                     if (e==0) {
525                       mapC[l]=mapC[l]+phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
526                       *BL*ProbaLibA(l,NL,b,BL,e,EL); }
527                     else if (e==1) {
528                       mapS[l]=mapS[l]+phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
529                       *BL*ProbaLibA(l,NL,b,BL,e,EL); } }
530                   else {
531                     mapA["B;"+sb+";"+sl+";"+se+";"]=mapA["B;"+sb+";"+sl+";"+se+";"]
```

```
532                 +phiA_moins*Klim0["B;"+sB+";"+sL+";"+sE+";"]
533                 *BL*ProbaLibA(l,NL,b,BL,e,EL);} } } } }
534
535       mapM=mapM+phiT*Klim0["B;"+sB+";"+sL+";"+sE+";"]*NL*NL
536       *ProbaFragA(1,NL,0,BL,0,EL,
537         lambda_moins,lambda_plus*Klim0["M;0;1;0;"]*EL);
538       for(int l=1; l<NL; l++) {
539         for(int b=0; b<=min_int(BL,l);b++) {
540           sl = std::to_string(l);
541           sb = std::to_string(b);
542           se = std::to_string(b+1);
543           if (b==0) {
544             mapS[l]=mapS[l]+phiT*Klim0["B;"+sB+";"+sL+";"+sE+";"]*NL*NL
545             *ProbaFragA(l,NL,b,BL,b+1,EL,
546               lambda_moins,lambda_plus*Klim0["M;0;1;0;"]*EL); }
547           else {
548             mapA["B;"+sb+";"+sl+";"+se+";"]=mapA["B;"+sb+";"+sl+";"+se+";"]
549             +phiT*Klim0["B;"+sB+";"+sL+";"+sE+";"]*NL*NL
550             *ProbaFragA(l,NL,b,BL,b+1,EL,
551               lambda_moins,lambda_plus*Klim0["M;0;1;0;"]*EL); } } } }
552
553
554     Klim1["M;0;1;0;"]=dt*mapM+Klim0["M;0;1;0;"];
555     Klim1["P;0;1;0;"]=dt*mapP+Klim0["P;0;1;0;"];
556
557     for(itS=mapS.begin(); itS!=mapS.end(); ++itS) {
558       sL = std::to_string(itS->first);
559       Klim1["S;0;"+sL+";1;"]=dt*mapS[itS->first]+Klim0["S;0;"+sL+";1;"]; }
560
561     for(itF=mapF.begin(); itF!=mapF.end(); ++itF) {
562       sL = std::to_string(itF->first);
563       Klim1["F;0;"+sL+";1;"]=dt*mapF[itF->first]+Klim0["F;0;"+sL+";1;"]; }
564
565     for(itC=mapC.begin(); itC!=mapC.end(); ++itC) {
566       sL = std::to_string(itC->first);
567       Klim1["C;0;"+sL+";0;"]=dt*mapC[itC->first]+Klim0["C;0;"+sL+";0;"]; }
568
569     for(itA=mapA.begin(); itA!=mapA.end(); ++itA) {
570       Klim1[itA->first.c_str()]=dt*mapA[itA->first.c_str()]
571       +Klim0[itA->first.c_str()]; }
572
573     //On calcule la somme ponderee de Klim1
574     for(it1=Klim1.begin(); it1!=Klim1.end(); ++it1) {
575        NL=findL(it1->first.c_str());
576        if (it1->second > 0.000000000001||it1->first.c_str()[0]=='M'
577          || it1->first.c_str()[0]=='P' ) {
578          sp=sp+(it1->second)*NL; } }
579      printf("sp %f\n", sp);
580
581
582      Klim0.clear();
583      for(it1=Klim1.begin(); it1!=Klim1.end(); ++it1) {
584         if (it1->second >= 0.000000000001||it1->first.c_str()[0]=='M'
585           || it1->first.c_str()[0]=='P') {
586             Klim0[it1->first.c_str()]=it1->second/sp; } }
587
588       //On vide les maps
589      mapM=0;
590      mapP=0;
591      mapS.clear();
592      mapF.clear();
```

```
593         mapC.clear();
594         mapA.clear();
595         mapSLong.clear();
596         mapFLong.clear();
597         mapCLong.clear();
598         mapAType.clear();
599         Klim1.clear();
600 
601        //On avance le temps
602        t=t+dt;
603        printf("%f\n", t);
604        cpt=cpt+1;
605 
606        //Tant qu'on a pas fait 100 millions d'iterations,
607       //on ecrit dans le fichier l'evolution toutes les 1000 iterations
608        if (cpt<100000000) {
609          if (cpt==next_print) {
610              p0=Klim0["M;0;1;0;"];
611              p2=Klim0["P;0;1;0;"];
612 
613              p1=0;
614              p3=0;
615              p4=0;
616              p5=0;
617 
618              pFilS=0;
619              pFilF=0;
620              pFilC=0;
621              pFilB=0;
622 
623           for (int i=0; i<lmax+1; i++) {
624             DistSlim[i]=0;
625             DistFlim[i]=0;
626             DistClim[i]=0;
627             DistBlim[i]=0;
628             pBran[i]=0;
629             pExt[i]=0;
630             pExtB[i]=0; }
631 
632           double MoyBranchLim=0.0;
633           double MoyExtLim=0.0;
634 
635           for(it=Klim0.begin(); it!=Klim0.end(); ++it){
636              NL=findL(it->first.c_str());
637              BL=findB(it->first.c_str());
638              EL=findE(it->first.c_str());
639              if (it->first.c_str()[0]!='M'
640              && it->first.c_str()[0]!='P' && it->first.c_str()[0]!='e') {
641                pExt[EL]=pExt[EL]+it->second;
642                pBran[BL]=pBran[BL]+it->second; }
643              if(it->first.c_str()[0]=='S') {
644                p1=p1+it->second*NL;
645                pFilS=pFilS+it->second;
646                DistSlim[NL]=it->second; }
647              else if(it->first.c_str()[0]=='F') {
648                p3=p3+it->second*NL;
649                pFilF=pFilF+it->second;
650                DistFlim[NL]=it->second; }
651              else if(it->first.c_str()[0]=='C') {
652                p4=p4+it->second*NL;
653                pFilC=pFilC+it->second;
```

```
654                 DistClim[NL]=it->second; }
655               else if(it->first.c_str()[0]=='B') {
656                 p5=p5+it->second*NL;
657                 pFilB=pFilB+it->second;
658                 DistBlim[NL]=it->second;
659                 pExtB[EL]=pExtB[EL]+it->second; } }
660
661           if (pFilB == 0) {
662             MoyBranchLim=0.0;
663             MoyExtLim=0.0; }
664           else {
665             for (int j=1; j<lmax+1; j++) {
666               MoyBranchLim=MoyBranchLim+j*pBran[j];
667               MoyExtLim=MoyExtLim+j*pExtB[j]; }
668
669             MoyBranchLim=MoyBranchLim/double(pFilB);
670             MoyExtLim=MoyExtLim/double(pFilB); }
671
672           fprintf(outputprop, "%f ; ", t);
673           fprintf(outputprop, "%f ; ", p0);
674           fprintf(outputprop, "%f ; ", p1);
675           fprintf(outputprop, "%f ; ", p2);
676           fprintf(outputprop, "%f ; ", p3);
677           fprintf(outputprop, "%f ; ", p4);
678           fprintf(outputprop, "%f ; ", p5);
679           fprintf(outputprop, "%f ;",p0+p1+p2+p3+p4+p5);
680           fflush(outputprop);
681
682           fprintf(outputprop, "%f ; ", pFilS);
683           fprintf(outputprop, "%f ; ", pFilF);
684           fprintf(outputprop, "%f ; ", pFilC);
685           fprintf(outputprop, "%f ;\n ", pFilB);
686
687           fprintf(outputS, "%f ; ", t);
688           fprintf(outputF, "%f ; ", t);
689           fprintf(outputC, "%f ; ", t);
690           fprintf(outputB, "%f ; ", t);
691
692           fprintf(outputBranche, "%f ; ", t);
693           fprintf(outputExtremite, "%f ; ", t);
694
695           for (int i=0; i<lmax+1; i++) {
696             fprintf(outputS, "%f ; ", DistSlim[i]);
697             fprintf(outputF, "%f ; ", DistFlim[i]);
698             fprintf(outputC, "%f ; ", DistClim[i]);
699             fprintf(outputB, "%f ; ", DistBlim[i]);
700             fprintf(outputBranche, "%f ; ", pBran[i]);
701             fprintf(outputExtremite, "%f ; ", pExt[i]); }
702
703           fprintf(outputS, "\n");
704           fprintf(outputF, "\n");
705           fprintf(outputC, "\n");
706           fprintf(outputB, "\n");
707
708           fprintf(outputBranche, "\n");
709           fprintf(outputExtremite, "\n");
710
711           fprintf(outputMoy, "%f ; ", t);
712           fprintf(outputMoy, "%f ; ", MoyBranchLim);
713           fprintf(outputMoy, "%f ;\n ", MoyExtLim);
714           next_print=next_print+1; } }
```

```
715       else {
716       //Ensuite on ecrit uniquement toutes les 1 000 000 iterations
717          if (cpt==next_print) {
718            p0=Klim0["M;0;1;0;"];
719            p2=Klim0["P;0;1;0;"];
720
721            p1=0;
722            p3=0;
723            p4=0;
724            p5=0;
725
726            pFilS=0;
727            pFilF=0;
728            pFilC=0;
729            pFilB=0;
730
731            for (int i=0; i<lmax+1; i++) {
732              DistSlim[i]=0;
733              DistFlim[i]=0;
734              DistClim[i]=0;
735              DistBlim[i]=0;
736              pBran[i]=0;
737              pExt[i]=0;
738              pExtB[i]=0; }
739
740            double MoyBranchLim=0.0;
741            double MoyExtLim=0.0;
742
743            for(it=Klim0.begin(); it!=Klim0.end(); ++it) {
744               NL=findL(it->first.c_str());
745               BL=findB(it->first.c_str());
746               EL=findE(it->first.c_str());
747               if (it->first.c_str()[0]!='M'&& it->first.c_str()[0]!='P'
748               && it->first.c_str()[0]!='e') {
749                  pExt[EL]=pExt[EL]+it->second;
750                  pBran[BL]=pBran[BL]+it->second; }
751               if(it->first.c_str()[0]=='S') {
752                  p1=p1+it->second*NL;
753                  pFilS=pFilS+it->second;
754                  DistSlim[NL]=it->second; }
755               else if(it->first.c_str()[0]=='F') {
756                  p3=p3+it->second*NL;
757                  pFilF=pFilF+it->second;
758                  DistFlim[NL]=it->second; }
759               else if(it->first.c_str()[0]=='C') {
760                  p4=p4+it->second*NL;
761                  pFilC=pFilC+it->second;
762                  DistClim[NL]=it->second; }
763               else if(it->first.c_str()[0]=='B') {
764                  p5=p5+it->second*NL;
765                  pFilB=pFilB+it->second;
766                  DistBlim[NL]=it->second;
767                  pExtB[EL]=pExtB[EL]+it->second; } }
768
769            if (pFilB == 0) {
770              MoyBranchLim=0.0;
771              MoyExtLim=0.0; }
772            else {
773              for (int j=1; j<lmax+1; j++) {
774                MoyBranchLim=MoyBranchLim+j*pBran[j];
775                MoyExtLim=MoyExtLim+j*pExtB[j]; }
```

```
776
777               MoyBranchLim=MoyBranchLim/double(pFilB);
778               MoyExtLim=MoyExtLim/double(pFilB); }
779
780             fprintf(outputprop, "%f ; ", t);
781             fprintf(outputprop, "%f ; ", p0);
782             fprintf(outputprop, "%f ; ", p1);
783             fprintf(outputprop, "%f ; ", p2);
784             fprintf(outputprop, "%f ; ", p3);
785             fprintf(outputprop, "%f ; ", p4);
786             fprintf(outputprop, "%f ; ", p5);
787             fprintf(outputprop, "%f ;",p0+p1+p2+p3+p4+p5);
788             fflush(outputprop);
789
790             fprintf(outputprop, "%f ; ", pFilS);
791             fprintf(outputprop, "%f ; ", pFilF);
792             fprintf(outputprop, "%f ; ", pFilC);
793             fprintf(outputprop, "%f ;\n ", pFilB);
794
795             fprintf(outputS, "%f ; ", t);
796             fprintf(outputF, "%f ; ", t);
797             fprintf(outputC, "%f ; ", t);
798             fprintf(outputB, "%f ; ", t);
799
800             fprintf(outputBranche, "%f ; ", t);
801             fprintf(outputExtremite, "%f ; ", t);
802
803             for (int i=0; i<lmax+1; i++) {
804               fprintf(outputS, "%f ; ", DistSlim[i]);
805               fprintf(outputF, "%f ; ", DistFlim[i]);
806               fprintf(outputC, "%f ; ", DistClim[i]);
807               fprintf(outputB, "%f ; ", DistBlim[i]);
808               fprintf(outputBranche, "%f ; ", pBran[i]);
809               fprintf(outputExtremite, "%f ; ", pExt[i]); }
810
811             fprintf(outputS, "\n");
812             fprintf(outputF, "\n");
813             fprintf(outputC, "\n");
814             fprintf(outputB, "\n");
815
816             fprintf(outputBranche, "\n");
817             fprintf(outputExtremite, "\n");
818
819             fprintf(outputMoy, "%f ; ", t);
820             fprintf(outputMoy, "%f ; ", MoyBranchLim);
821             fprintf(outputMoy, "%f ;\n ", MoyExtLim);
822             next_print=next_print+1000000; } } }
823
824   //On ferme le fichier résultat
825   fclose(outputS);
826   fclose(outputF);
827   fclose(outputC);
828   fclose(outputB);
829   fclose(outputBranche);
830   fclose(outputExtremite);
831   fclose(outputMoy);
832   fclose(outputprop);
833
834   return Klim0; }
```

2 - Auxiliary Functions

❉ Fonction pour calculer le taux total d'événement dans la population.

```
double TauxTotalPop(std::map<std::string,int> K, int N,
  double lambda_plus, double lambda_moins, double lambdaF_plus,
  double lambdaN, double phiP, double phiF_plus, double phiF_moins,
  double phiA_plus, double phiA_moins, double phiC, double phiT) {
  double taux_total_pop = 0;
  std::map<std::string, int>::iterator it;

  int longu;
  int branch;
  int extre;

  for(it=K.begin(); it!=K.end(); ++it){
      longu=findL(it->first.c_str());
      branch=findB(it->first.c_str());
      extre=findE(it->first.c_str());
      //a - Simple
     if(it->first.c_str()[0]=='S') {
       taux_total_pop=taux_total_pop
       +lambda_plus*K["M;0;1;0;"]*it->second/double(N)
       +lambda_moins*it->second+phiF_plus*it->second+phiA_plus*it->second*longu
       +phiC*it->second+phiT*it->second*longu*longu; }
       //b - Formine
     else if(it->first.c_str()[0]=='F') {
       taux_total_pop=taux_total_pop
       +lambdaF_plus*K["P;0;1;0;"]*it->second/double(N)
       +lambda_moins*it->second+phiF_moins*it->second
       +phiT*it->second*longu*longu; }
       //c - Branchant
     else if(it->first.c_str()[0]=='B') {
       taux_total_pop=taux_total_pop
       +lambda_plus*K["M;0;1;0;"]*it->second*extre/double(N)
       +lambda_moins*it->second+phiA_plus*it->second*(longu-branch)
       +phiA_moins*it->second*branch+phiC*it->second*extre
       +phiT*it->second*(longu-branch)*(longu-branch); }
       //d - Cappé
     else if(it->first.c_str()[0]=='C') {
       taux_total_pop=taux_total_pop+lambda_moins*it->second
       +phiA_plus*it->second*longu+phiT*it->second*longu*longu; }
       //e - Profiline
     else if(it->first.c_str()[0]=='P') {
       taux_total_pop=taux_total_pop+phiP*K["M;0;1;0;"]; } }
   taux_total_pop=taux_total_pop
   +lambdaN*K["M;0;1;0;"]*(K["M;0;1;0;"]-1)*(K["M;0;1;0;"]-2)/double(N*N);

   return taux_total_pop; }
```

❉ Fonction permettant de faire évoluer un filament simple.

```
std::map<std::string,int> EvolS(std::map<std::string,int> K0, int N,
  double lambda_plus, double lambda_moins, double lambdaN,
  double phiP,double phiF_plus,double phiA_plus, double phiC,
  double phiT,int longueur) {
  std::map<std::string,int> K1=K0;
  K1["event"]=0;

```

```
8   double taux_elongation=lambda_plus*K0["M;0;1;0;"]/N;
9   double taux_depolymerisation=lambda_moins;
10  double taux_creation_complexe=phiP*K0["M;0;1;0;"];
11  double taux_fixation_formine=phiF_plus;
12  double taux_fixation_arp=phiA_plus*longueur;
13  double taux_capping=phiC;
14  double taux_fragmentation=phiT*longueur*longueur;
15  double taux_nucleation=lambdaN*K0["M;0;1;0;"]*(K0["M;0;1;0;"]-1.0)
16  *(K0["M;0;1;0;"]-2)/double(N*N);
17
18  double taux_total_filament=taux_elongation+taux_depolymerisation+
19  taux_creation_complexe+taux_fixation_formine+taux_fixation_arp+
20  taux_capping+taux_fragmentation+taux_nucleation;
21
22  int alea1;
23  bool alea2;
24  int T;
25  int G;
26
27  std::string slongeur = std::to_string(longueur);
28  std::string slongeur_plus1 = std::to_string(longueur+1);
29  std::string slongeur_moins1 = std::to_string(longueur-1);
30
31  alea1=floor(8*rand01()+1);
32  if(alea1== 1) {
33    alea2 = (rand01() <= (taux_elongation/taux_total_filament));
34    if (alea2 == 1 && longueur<N) {
35      K1["S;0;"+slongeur+";1;"]=K0["S;0;"+slongeur+";1;"]-1;
36      K1["S;0;"+slongeur_plus1+";1;"]=K0["S;0;"+slongeur_plus1+";1;"]+1;
37      K1["M;0;1;0;"]=K0["M;0;1;0;"]-1;
38      K1["event"]=1; } }
39  else if (alea1== 2) {
40    alea2 = (rand01() <= (taux_depolymerisation/taux_total_filament));
41    if (alea2 == 1) {
42      if (longueur>3) {
43        K1["S;0;"+slongeur+";1;"]=K0["S;0;"+slongeur+";1;"]-1;
44        K1["S;0;"+slongeur_moins1+";1;"]=K0["S;0;"+slongeur_moins1+";1;"]+1;
45        K1["M;0;1;0;"]=K0["M;0;1;0;"]+1;
46        K1["event"]=1; }
47      else {
48        K1["S;0;"+slongeur+";1;"]=K0["S;0;"+slongeur+";1;"]-1;
49        K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur;
50        K1["event"]=1; } } }
51  else if (alea1== 3) {
52    alea2 = (rand01() <= (taux_creation_complexe/taux_total_filament));
53    if (alea2 == 1) {
54      K1["M;0;1;0;"]=K0["M;0;1;0;"]-1;
55      K1["P;0;1;0;"]=K0["P;0;1;0;"]+1;
56      K1["event"]=1; } }
57  else if (alea1== 4) {
58    alea2 = (rand01() <= (taux_fixation_formine/taux_total_filament));
59    if (alea2 == 1) {
60      K1["S;0;"+slongeur+";1;"]=K0["S;0;"+slongeur+";1;"]-1;
61      K1["F;0;"+slongeur+";1;"]=K0["F;0;"+slongeur+";1;"]+1;
62      K1["event"]=1; } }
63  else if (alea1== 5) {
64    alea2 = (rand01() <= (taux_fixation_arp/taux_total_filament));
65    if (alea2 == 1) {
66      K1["S;0;"+slongeur+";1;"]=K0["S;0;"+slongeur+";1;"]-1;
67      K1["B;1;"+slongeur+";2;"]=K0["B;1;"+slongeur+";2;"]+1;
68      K1["event"]=1; } }
```

```
  else if (alea1== 6) {
    alea2 = (rand01() <= (taux_capping/taux_total_filament));
    if (alea2 == 1) {
      K1["S;0;"+slongeur+";1;"]=K0["S;0;"+slongeur+";1;"]-1;
      K1["C;0;"+slongeur+";0;"]=K0["C;0;"+slongeur+";0;"]+1;
      K1["event"]=1; } }
  else if (alea1== 7) {
    alea2 = (rand01() <= (taux_fragmentation/taux_total_filament));
    if (alea2 == 1) {
      K1["S;0;"+slongeur+";1;"]=K0["S;0;"+slongeur+";1;"]-1;
      if (longueur >= 4) {
        G=rand_geom(longueur-4,
          fmax(0,1-(taux_depolymerisation/taux_elongation)));
        T=G+3;
        std::string sT = std::to_string(T);
        K1["S;0;"+sT+";1;"]=K0["S;0;"+sT+";1;"]+1;
        K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur-T; }
      else {
        K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur; }
      K1["event"]=1; } }
  else if (alea1== 8) {
    alea2 = (rand01() <= (taux_nucleation/taux_total_filament));
    if (alea2 == 1) {
      K1["M;0;1;0;"]=K0["M;0;1;0;"]-3;
      K1["S;0;3;1;"]=K0["S;0;3;1;"]+1;
      K1["event"]=1; } }
  return K1; }
```

* Fonction permettant de faire évoluer un filament associé à une formine.

```
std::map<std::string,int> EvolF(std::map<std::string,int> K0, int N,
  double lambda_moins, double lambdaF_plus, double lambdaN,
  double phiP, double phiF_moins, double phiT, int longueur) {
  std::map<std::string,int> K1=K0;
  K1["event"]=0;

  double taux_elongation=lambdaF_plus*K0["P;0;1;0;"]/N;
  double taux_depolymerisation=lambda_moins;
  double taux_creation_complexe=phiP*K0["M;0;1;0;"];
  double taux_liberation_formine=phiF_moins;
  double taux_fragmentation=phiT*longueur*longueur;
  double taux_nucleation=lambdaN*K0["M;0;1;0;"]*(K0["M;0;1;0;"]-1)
  *(K0["M;0;1;0;"]-2)/double(N*N);

  double taux_total_filament=taux_elongation+taux_depolymerisation+
  taux_creation_complexe+taux_liberation_formine+
  taux_fragmentation+taux_nucleation;

  int alea;
  bool alea2;
  int T;
  int G;

  std::string slongeur = std::to_string(longueur);
  std::string slongeur_plus1 = std::to_string(longueur+1);
  std::string slongeur_moins1 = std::to_string(longueur-1);

  alea=floor(6*rand01()+1);
  if(alea == 1) {
    alea2 = (rand01() <= (taux_elongation/taux_total_filament));
```

```
    if (alea2 == 1 && longueur<N) {
      K1["F;0;"+slongeur+";1;"]=K0["F;0;"+slongeur+";1;"]-1;
      K1["F;0;"+slongeur_plus1+";1;"]=K0["F;0;"+slongeur_plus1+";1;"]+1;
      K1["P;0;1;0;"]=K0["P;0;1;0;"]-1;
      K1["event"]=1; } }
  else if (alea == 2) {
    alea2 = (rand01() <= (taux_depolymerisation/taux_total_filament));
    if (alea2 == 1) {
      if (longueur>3) {
        K1["F;0;"+slongeur+";1;"]=K0["F;0;"+slongeur+";1;"]-1;
        K1["F;0;"+slongeur_moins1+";1;"]=K0["F;0;"+slongeur_moins1+";1;"]+1;
        K1["M;0;1;0;"]=K0["M;0;1;0;"]+1;
        K1["event"]=1; }
      else {
        K1["F;0;"+slongeur+";1;"]=K0["F;0;"+slongeur+";1;"]-1;
        K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur;
        K1["event"]=1; } } }
  else if (alea == 3) {
    alea2 = (rand01() <= (taux_creation_complexe/taux_total_filament));
    if (alea2 == 1) {
      K1["M;0;1;0;"]=K0["M;0;1;0;"]-1;
      K1["P;0;1;0;"]=K0["P;0;1;0;"]+1;
      K1["event"]=1; } }
  else if (alea == 4) {
    alea2 = (rand01()<=(taux_liberation_formine/taux_total_filament));
    if (alea2 == 1) {
      K1["F;0;"+slongeur+";1;"]=K0["F;0;"+slongeur+";1;"]-1;
      K1["S;0;"+slongeur+";1;"]=K0["S;0;"+slongeur+";1;"]+1;
      K1["event"]=1; } }
  else if (alea == 5) {
    alea2 = (rand01() <= (taux_fragmentation/taux_total_filament));
    if (alea2 == 1) {
      K1["F;0;"+slongeur+";1;"]=K0["F;0;"+slongeur+";1;"]-1;
      if (longueur >= 4) {
        G=rand_geom(longueur-4,
          fmax(0,1-(taux_depolymerisation/taux_elongation)));
        T=G+3;
        std::string sT = std::to_string(T);
        K1["F;0;"+sT+";1;"]=K0["F;0;"+sT+";1;"]+1;
        K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur-T; }
      else {
        K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur; }
      K1["event"]=1; } }
  else if (alea == 6) {
    alea2 = (rand01() <= (taux_nucleation/taux_total_filament));
    if (alea2 == 1) {
      K1["M;0;1;0;"]=K0["M;0;1;0;"]-3;
      K1["S;0;3;1;"]=K0["S;0;3;1;"]+1;
      K1["event"]=1; } }
  return K1; }
```

❄ Fonction permettant de faire évoluer un filament linéaire cappé.

```
std::map<std::string,int> EvolC(std::map<std::string,int> K0, int N,
   double lambda_moins, double lambdaN,double phiP,double phiA_plus,
   double phiT, int longueur) {
  std::map<std::string,int> K1=K0;
  K1["event"]=0;
  double taux_depolymerisation=lambda_moins;
  double taux_creation_complexe=phiP*K0["M;0;1;0;"];
```

```
  double taux_fixation_arp=phiA_plus*longueur;
  double taux_fragmentation=phiT*longueur*longueur;
  double taux_nucleation=lambdaN*K0["M;0;1;0;"]*(K0["M;0;1;0;"]-1)
  *(K0["M;0;1;0;"]-2)/double(N*N);

  double taux_total_filament=taux_depolymerisation+
  taux_creation_complexe+taux_fixation_arp+taux_fragmentation+taux_nucleation;
  int alea;
  bool alea2;
  std::string slongeur = std::to_string(longueur);
  std::string slongeur_moins1 = std::to_string(longueur-1);
  alea=floor(5*rand01()+1);

  if (alea == 1) {
    alea2 = (rand01() <= (taux_depolymerisation/taux_total_filament));
    if (alea2 == 1) {
      if (longueur>3) {
        K1["C;0;"+slongeur+";0;"]=K0["C;0;"+slongeur+";0;"]-1;
        K1["C;0;"+slongeur_moins1+";0;"]=K0["C;0;"+slongeur_moins1+";0;"]+1;
        K1["M;0;1;0;"]=K0["M;0;1;0;"]+1;
        K1["event"]=1; }
      else {
        K1["C;0;"+slongeur+";0;"]=K0["C;0;"+slongeur+";0;"]-1;
        K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur;
        K1["event"]=1; } } }
  else if (alea == 2) {
    alea2 = (rand01() <= (taux_creation_complexe/taux_total_filament));
    if (alea2 == 1) {
      K1["M;0;1;0;"]=K0["M;0;1;0;"]-1;
      K1["P;0;1;0;"]=K0["P;0;1;0;"]+1;
      K1["event"]=1; } }
  else if (alea == 3) {
    alea2 = (rand01() <= (taux_fixation_arp/taux_total_filament));
    if (alea2 == 1) {
      K1["C;0;"+slongeur+";0;"]=K0["C;0;"+slongeur+";0;"]-1;
      K1["B;1;"+slongeur+";1;"]=K0["B;1;"+slongeur+";1;"]+1;
      K1["event"]=1; } }
  else if (alea == 4) {
    alea2 = (rand01() <= (taux_fragmentation/taux_total_filament));
    if (alea2 == 1) {
      K1["C;0;"+slongeur+";0;"]=K0["C;0;"+slongeur+";0;"]-1;
      K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur;
      K1["event"]=1; } }
  else if (alea == 5) {
    alea2 = (rand01() <= (taux_nucleation/taux_total_filament));
    if (alea2 == 1) {
      K1["M;0;1;0;"]=K0["M;0;1;0;"]-3;
      K1["S;0;3;1;"]=K0["S;0;3;1;"]+1;
      K1["event"]=1; } }
  return K1; }
```

❋ Fonction permettant de faire évoluer un filament branché.

```
#include "Frag1.cpp"
#include "Frag2.cpp"

std::map<std::string,int> EvolB(std::map<std::string,int> K0, int N,
  double lambda_plus, double lambda_moins, double lambdaN,
  double phiP, double phiA_plus, double phiA_moins, double phiC,
  double phiT,int longueur, int extremite, int branchement) {
```

```
8   std::map<std::string,int> K1=K0;
9   K1["event"]=0;
10
11  double taux_elongation=lambda_plus*K0["M;0;1;0;"]*extremite/N;
12  double taux_depolymerisation=lambda_moins;
13  double taux_creation_complexe=phiP*K0["M;0;1;0;"];
14  double taux_fixation_arp=phiA_plus*(longueur-branchement);
15  double taux_liberation_arp=phiA_moins*branchement;
16  double taux_capping=phiC*extremite;
17  double taux_fragmentation=phiT*(longueur-branchement)*(longueur-branchement);
18  double taux_nucleation=lambdaN*K0["M;0;1;0;"]*(K0["M;0;1;0;"]-1)
19  *(K0["M;0;1;0;"]-2)/double(N*N);
20
21  double taux_total_filament=taux_elongation+taux_depolymerisation+
22  taux_creation_complexe+taux_fixation_arp+taux_liberation_arp+
23  taux_capping+taux_fragmentation+taux_nucleation;
24
25  int alea;
26  bool alea2;
27  double BsurL;
28  double EsurB;
29  bool bernoulli;
30  int L1;
31  int L2;
32  int B1;
33  int B2;
34  int E1;
35  int E2;
36
37  alea=floor(8*rand01()+1);
38
39  std::string slongeur = std::to_string(longueur);
40  std::string slongeur_plus1 = std::to_string(longueur+1);
41  std::string slongeur_moins1 = std::to_string(longueur-1);
42
43  std::string sextremite = std::to_string(extremite);
44  std::string sextremite_plus1 = std::to_string(extremite+1);
45  std::string sextremite_moins1 = std::to_string(extremite-1);
46
47  std::string sbranchement = std::to_string(branchement);
48  std::string sbranchement_plus1 = std::to_string(branchement+1);
49  std::string sbranchement_moins1 = std::to_string(branchement-1);
50
51  if(alea == 1) {
52    alea2 = (rand01() <= (taux_elongation/taux_total_filament));
53    if (alea2 == 1 && longueur<N) {
54      K1["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]
55      =K0["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]-1;
56      K1["B;"+sbranchement+";"+slongeur_plus1+";"+sextremite+";"]
57      =K0["B;"+sbranchement+";"+slongeur_plus1+";"+sextremite+";"]+1;
58      K1["M;0;1;0;"]=K0["M;0;1;0;"]-1;
59      K1["event"]=1; } }
60  else if (alea == 2) {
61    alea2 = (rand01() <= (taux_depolymerisation/taux_total_filament));
62    if (alea2 == 1) {
63      K1["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]
64      =K0["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]-1;
65      if (longueur == 3) {
66        K1["M;0;1;0;"]=K0["M;0;1;0;"]+3;
67        K1["event"]=1; }
68      else {
```

```
69          K1["M;0;1;0;"]=K0["M;0;1;0;"]+1;
70          BsurL = double(branchement)/double(longueur);
71          EsurB = double(extremite)/double(branchement+1);
72          bernoulli = (rand01() <= BsurL);
73          if (bernoulli == 0) {
74            K1["B;"+sbranchement+";"+slongeur_moins1+";"+sextremite+";"]
75           =K0["B;"+sbranchement+";"+slongeur_moins1+";"+sextremite+";"]+1; }
76          else {
77            L1=rand_unif(0,(longueur-1));
78            L2=longueur-1-L1;
79            B1=rand_unif(max_int(0,branchement-L2-1),
80            min_int(branchement-1,L1));
81            B2=branchement-1-B1;
82
83            if (max_int(0,extremite-B2-1)==0) {
84              if (extremite == branchement + 1) {
85                E1=rand_unif(1,min_int(extremite,B1+1)); }
86              else {
87                E1=rand_unif(0,min_int(extremite,B1+1)); } }
88            else {
89              if (extremite-B2-1 > B1+1) {
90                if (extremite == branchement + 1) {
91                  E1=rand_unif(1,min_int(extremite,B1+1)); }
92                else {
93                  E1=rand_unif(0,min_int(extremite,B1+1)); } }
94              else {
95                E1=rand_unif(extremite-B2-1,min_int(extremite,B1+1)); } }
96
97            E2=extremite-E1;
98
99            std::string sL1 = std::to_string(L1);
100           std::string sL2 = std::to_string(L2);
101           std::string sB1 = std::to_string(B1);
102           std::string sB2 = std::to_string(B2);
103           std::string sE1 = std::to_string(E1);
104           std::string sE2 = std::to_string(E2);
105
106           if (L1 == L2 && B1==B2 && E1==E2) {
107             if (L1 >= 3) {
108               if (B1 > 0) {
109                 K1["B;"+sB1+";"+sL1+";"+sE1+";"]
110                 =K0["B;"+sB1+";"+sL1+";"+sE1+";"]+2; }
111               else {
112                 if (E1==1) {
113                   K1["S;0;"+sL1+";1;"]=K0["S;0;"+sL1+";1;"]+2; }
114                 else {
115                   K1["C;0;"+sL1+";0;"]=K0["C;0;"+sL1+";0;"]+2; } } }
116             else {
117               K1["M;0;1;0;"]=K1["M;0;1;0;"]+longueur-1; } }
118           else {
119             if (L1 >= 3) {
120               if (L2 >= 3) {
121                 if (B1 > 0) {
122                   K1["B;"+sB1+";"+sL1+";"+sE1+";"]
123                   =K0["B;"+sB1+";"+sL1+";"+sE1+";"]+1; }
124                 else {
125                   if (E1==1) {
126                     K1["S;0;"+sL1+";1;"]=K0["S;0;"+sL1+";1;"]+1; }
127                   else {
128                     K1["C;0;"+sL1+";0;"]=K0["C;0;"+sL1+";0;"]+1; } }
129
```

```
                if (B2 > 0) {
                  K1["B;"+sB2+";"+sL2+";"+sE2+";"]
                  =K0["B;"+sB2+";"+sL2+";"+sE2+";"]+1; }
                else {
                  if (E2==1) {
                    K1["S;0;"+sL2+";1;"]=K0["S;0;"+sL2+";1;"]+1; }
                  else {
                    K1["C;0;"+sL2+";0;"]=K0["C;0;"+sL2+";0;"]+1; } } }
              else {
                if (B1 > 0) {
                  K1["B;"+sB1+";"+sL1+";"+sE1+";"]
                  =K0["B;"+sB1+";"+sL1+";"+sE1+";"]+1; }
                else {
                  if (E1==1) {
                    K1["S;0;"+sL1+";1;"]=K0["S;0;"+sL1+";1;"]+1; }
                  else {
                    K1["C;0;"+sL1+";0;"]=K0["C;0;"+sL2+";0;"]+1; } }

                K1["M;0;1;0;"]=K1["M;0;1;0;"]+L2; } }
            else {
              if (L2 >= 3) {
                K1["M;0;1;0;"]=K1["M;0;1;0;"]+L1;

                if (B2 > 0) {
                  K1["B;"+sB2+";"+sL2+";"+sE2+";"]
                  =K0["B;"+sB2+";"+sL2+";"+sE2+";"]+1; }
                else {
                  if (E2==1) {
                    K1["S;0;"+sL2+";1;"]=K0["S;0;"+sL2+";1;"]+1; }
                  else {
                    K1["C;0;"+sL2+";0;"]=K0["C;0;"+sL2+";0;"]+1; } } }
              else {
                K1["M;0;1;0;"]=K1["M;0;1;0;"]+longueur-1; } } } }
        K1["event"]=1; } } }
  else if (alea == 3) {
    alea2 = (rand01() <= (taux_creation_complexe/taux_total_filament));
    if (alea2 == 1) {
      K1["M;0;1;0;"]=K0["M;0;1;0;"]-1;
      K1["P;0;1;0;"]=K0["P;0;1;0;"]+1;
      K1["event"]=1; } }
  else if (alea == 4) {
    alea2 = (rand01() <= (taux_fixation_arp/taux_total_filament));
    if (alea2 == 1 && branchement<longueur) {
      K1["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]
      =K0["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]-1;
      K1["B;"+sbranchement_plus1+";"+slongeur+";"+sextremite_plus1+";"]
     =K0["B;"+sbranchement_plus1+";"+slongeur+";"+sextremite_plus1+";"]+1;
      K1["event"]=1; } }
  else if (alea == 5) {
    alea2 = (rand01() <= (taux_liberation_arp/taux_total_filament));
    if (alea2 == 1) {
      K1["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]
     =K0["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]-1;
      L1=rand_unif(0,longueur);
      L2=longueur-L1;
      B1=rand_unif(max_int(0,branchement-L2-1),min_int(L1,branchement-1));
      B2=branchement-1-B1;

      if (max_int(0,extremite-B2-1)==0) {
        if (extremite == branchement + 1) {
          E1=rand_unif(1,min_int(extremite,B1+1)); }
```

```
191         else {
192           E1=rand_unif(0,min_int(extremite,B1+1)); } }
193       else {
194         if (extremite-B2-1 > B1+1) {
195           if (extremite == branchement + 1) {
196             E1=rand_unif(1,min_int(extremite,B1+1)); }
197           else {
198             E1=rand_unif(0,min_int(extremite,B1+1)); } }
199         else {
200           E1=rand_unif(extremite-B2-1,min_int(extremite,B1+1)); } }
201
202       E2=extremite-E1;
203
204       std::string sL1 = std::to_string(L1);
205       std::string sL2 = std::to_string(L2);
206       std::string sB1 = std::to_string(B1);
207       std::string sB2 = std::to_string(B2);
208       std::string sE1 = std::to_string(E1);
209       std::string sE2 = std::to_string(E2);
210
211       if(L1==L2 && B1==B2 && E1==E2) {
212         if (L1 >=3) {
213           if (B1>0) {
214             K1["B;"+sB1+";"+sL1+";"+sE1+";"]
215             =K0["B;"+sB1+";"+sL1+";"+sE1+";"]+2; }
216           else {
217             if (E1==1) {
218               K1["S;0;"+sL1+";1;"]=K0["S;0;"+sL1+";1;"]+2; }
219             else {
220               K1["C;0;"+sL1+";0;"]=K0["C;0;"+sL1+";0;"]+2; } } }
221         else {
222           K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur; } }
223       else {
224         if (L1 >= 3) {
225           if (L2 >= 3) {
226             if (B1>0) {
227               K1["B;"+sB1+";"+sL1+";"+sE1+";"]
228               =K0["B;"+sB1+";"+sL1+";"+sE1+";"]+1; }
229             else {
230               if (E1==1) {
231                 K1["S;0;"+sL1+";1;"]=K0["S;0;"+sL1+";1;"]+1; }
232               else {
233                 K1["C;0;"+sL1+";0;"]=K0["C;0;"+sL1+";0;"]+1; } }
234
235             if (B2>0) {
236               K1["B;"+sB2+";"+sL2+";"+sE2+";"]
237               =K0["B;"+sB2+";"+sL2+";"+sE2+";"]+1; }
238             else {
239               if (E2==1) {
240                 K1["S;0;"+sL2+";1;"]=K0["S;0;"+sL2+";1;"]+1; }
241               else {
242                 K1["C;0;"+sL2+";0;"]=K0["C;0;"+sL2+";0;"]+1; } } }
243           else {
244             if (B1>0) {
245               K1["B;"+sB1+";"+sL1+";"+sE1+";"]
246               =K0["B;"+sB1+";"+sL1+";"+sE1+";"]+1; }
247             else {
248               if (E1==1) {
249                 K1["S;0;"+sL1+";1;"]=K0["S;0;"+sL1+";1;"]+1; }
250               else {
251                 K1["C;0;"+sL1+";0;"]=K0["C;0;"+sL1+";0;"]+1; } }
```

```
252
253             K1["M;0;1;0;"]=K0["M;0;1;0;"]+L2; } }
254         else {
255           if (L2 >= 3) {
256             K1["M;0;1;0;"]=K0["M;0;1;0;"]+L1;
257
258             if (B2 >0) {
259               K1["B;"+sB2+";"+sL2+";"+sE2+";"]
260               =K0["B;"+sB2+";"+sL2+";"+sE2+";"]+1; }
261             else {
262               if (E2==1) {
263                 K1["S;0;"+sL2+";1;"]=K0["S;0;"+sL2+";1;"]+1; }
264               else {
265                 K1["C;0;"+sL2+";0;"]=K0["C;0;"+sL2+";0;"]+1; } } }
266           else {
267             K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur; } } }
268       K1["event"]=1; } }
269   else if (alea == 6) {
270     alea2 = (rand01() <= (taux_capping/taux_total_filament));
271     if (alea2 == 1 && extremite>0) {
272       K1["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]
273       =K0["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]-1;
274       K1["B;"+sbranchement+";"+slongeur+";"+sextremite_moins1+";"]
275       =K0["B;"+sbranchement+";"+slongeur+";"+sextremite_moins1+";"]+1;
276       K1["event"]=1; } }
277   else if (alea == 7) {
278     alea2 = (rand01() <= (taux_fragmentation/taux_total_filament));
279     if (alea2 == 1) {
280       K1=Frag2(K0,K1,longueur,extremite,branchement,
281         taux_depolymerisation, taux_elongation); } }
282   else if (alea == 8) {
283     alea2 = (rand01() <= (taux_nucleation/taux_total_filament));
284     if (alea2 == 1) {
285       K1["M;0;1;0;"]=K0["M;0;1;0;"]-3;
286       K1["S;0;3;1;"]=K0["S;0;3;1;"]+1;
287       K1["event"]=1; } }
288
289   return K1; }
```

❋ Fonction permettant de faire évoluer le nombre de monomères libres.

```
1  std::map<std::string,int> EvolM(std::map<std::string,int> K0, int N,
2    double lambdaN,double phiP) {
3    std::map<std::string,int> K1=K0;
4    K1["event"]=0;
5
6    double taux_creation_complexe=phiP*K0["M;0;1;0;"];
7    double taux_nucleation=lambdaN*K0["M;0;1;0;"]*(K0["M;0;1;0;"]-1.0)
8    *(K0["M;0;1;0;"]-2)/double(N*N);
9
10   double taux_total_filament=taux_creation_complexe+taux_nucleation;
11
12   int alea1;
13   bool alea2;
14
15   alea1=floor(2*rand01()+1);
16
17   if (alea1== 1) {
18     alea2 = (rand01() <= (taux_creation_complexe/taux_total_filament));
19     if (alea2 == 1) {
```

```
      K1["M;0;1;0;"]=K0["M;0;1;0;"]-1;
      K1["P;0;1;0;"]=K0["P;0;1;0;"]+1;
      K1["event"]=1; } }
  else if (alea1== 2) {
    alea2 = (rand01() <= (taux_nucleation/taux_total_filament));
    if (alea2 == 1) {
      K1["M;0;1;0;"]=K0["M;0;1;0;"]-3;
      K1["S;0;3;1;"]=K0["S;0;3;1;"]+1;
      K1["event"]=1; } }

  return K1; }
```

* Fonction permettant d'appliquer une fragmentation à un filament branché (comme décrit dans le modèle Section 4.2).

```
std::map<std::string,int> Frag2(std::map<std::string,int> K0,
  std::map<std::string,int> K1,int longueur, int extremite,
  int branchement,double taux_depolymerisation, double taux_elongation) {
  std::string slongeur = std::to_string(longueur);
  std::string sextremite = std::to_string(extremite);
  std::string sbranchement = std::to_string(branchement);

  K1["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]
  =K0["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]-1;

  int nf;
  int cap; //Nombre d'extremités cappées a l'origine
  cap = branchement+1-extremite;
  if (min_int(extremite,(longueur-max_int(1,cap))/3)>1) {
    nf=rand_unif(1,min_int(extremite,(longueur-max_int(1,cap))/3)); }
  else {
    nf=0; }

  if(nf==0) {
    K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur; }
  else {
    if (extremite-nf==0) {
    //Nombre de branchement qu'il reste a distribuer
      int G=rand_geom(longueur-max_int(1,cap)-3*nf,
      fmax(0,1-(taux_depolymerisation/taux_elongation)));
      int T=G+3*nf;

      int L[nf];
      int SumL=0;

      for(int j=0; j<(nf-1); j++) {
        L[j]=T/nf;
        SumL=SumL+L[j]; }
      L[nf-1]=T-SumL;
      SumL=SumL+L[nf-1];

      std::string sNewL;

      K1["S;0;"+sNewL+";1;"]=K0["S;0;"+sNewL+";1;"];

      for (int j=0; j<nf; j++) {
        sNewL = std::to_string(L[j]);
        K1["S;0;"+sNewL+";1;"]=K1["S;0;"+sNewL+";1;"]+1; }
      K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur-T; }
    else {
```

```
46        int B[nf];
47        int SumB=0;
48
49        for (int j=0; j<(nf-1); j++) {
50          B[j]=(extremite-nf)/nf;
51          SumB=SumB+B[j]; }
52        B[nf-1]=extremite-nf-SumB;
53        SumB=SumB+B[nf-1];
54
55        int G=rand_geom((longueur-max_int(1,cap)-max_int(3*nf,extremite-nf)),
56        fmax(0,1-(taux_depolymerisation/taux_elongation)));
57        int T=G+max_int(3*nf,extremite-nf);
58
59        int L[nf];
60        int SumL=0;
61
62        for(int j=0; j<(nf-1); j++) {
63          L[j]=T/nf;
64          SumL=SumL+L[j]; }
65        L[nf-1]=T-SumL;
66        SumL=SumL+L[nf-1];
67
68        std::string sNewL;
69        std::string sNewB;
70        std::string sNewE;
71
72        K1["B;"+sNewB+";"+sNewL+";"+sNewE+";"]
73        =K0["B;"+sNewB+";"+sNewL+";"+sNewE+";"];
74        K1["S;0;"+sNewL+";1;"]=K0["S;0;"+sNewL+";1;"];
75
76        for (int j=0; j<nf; j++) {
77          sNewB = std::to_string(B[j]);
78          sNewL = std::to_string(L[j]);
79          sNewE = std::to_string(B[j]+1);
80
81          if (B[j] > 0) {
82            K1["B;"+sNewB+";"+sNewL+";"+sNewE+";"]
83            =K1["B;"+sNewB+";"+sNewL+";"+sNewE+";"]+1; }
84          else {
85            K1["S;0;"+sNewL+";1;"]=K1["S;0;"+sNewL+";1;"]+1; } }
86        K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur-T; } }
87
88    K1["event"]=1;
89    return K1; }
```

* Fonction permettant d'appliquer une fragmentation à un filament branché (version où la longueur, le nombre de branchements et le nombre d'extrémités libre de chaque filament créé après la fragmentation est choisi aléatoirement selon une loi uniforme).

```
1  std::map<std::string,int> Frag1(std::map<std::string,int> K0,
2    std::map<std::string,int> K1,int longueur, int extremite,
3    int branchement,double taux_depolymerisation, double taux_elongation) {
4    std::string slongeur = std::to_string(longueur);
5    std::string sextremite = std::to_string(extremite);
6    std::string sbranchement = std::to_string(branchement);
7
8    K1["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]
9    =K0["B;"+sbranchement+";"+slongeur+";"+sextremite+";"]-1;
10
11   int nf;
```

```
12  int cap;
13  cap = branchement+1-extremite;
14  if (min_int(extremite,(longueur-max_int(1,cap))/3)>1) {
15    nf=rand_unif(1,min_int(extremite,(longueur-max_int(1,cap))/3)); }
16  else {
17    nf=0; }
18
19  if(nf==0) {
20    K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur; }
21  else {
22    if (extremite-nf==0) {
23      int G=rand_geom(longueur-max_int(1,cap)-3*nf,
24      fmax(0,1-(taux_depolymerisation/taux_elongation)));
25      int T=G+3*nf;
26
27      int L[nf];
28      int SumL=0;
29
30      for(int j=0; j<(nf-1); j++) {
31        L[j]=rand_unif(3,T-3*(nf-(j+1))-SumL);
32        SumL=SumL+L[j]; }
33      L[nf-1]=T-SumL;
34      SumL=SumL+L[nf-1];
35
36      std::string sNewL;
37
38      K1["S;0;"+sNewL+";1;"]=K0["S;0;"+sNewL+";1;"];
39
40      for (int j=0; j<nf; j++) {
41        sNewL = std::to_string(L[j]);
42        K1["S;0;"+sNewL+";1;"]=K1["S;0;"+sNewL+";1;"]+1; }
43      K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur-T; }
44    else {
45      int B[nf];
46      int SumB=0;
47
48      for (int j=0; j<(nf-1); j++) {
49        B[j]=rand_unif(0,(extremite-nf-SumB));
50        SumB=SumB+B[j]; }
51      B[nf-1]=extremite-nf-SumB;
52      SumB=SumB+B[nf-1];
53
54      int G=rand_geom((longueur-max_int(1,cap)-max_int(3*nf,extremite-nf)),
55      fmax(0,1-(taux_depolymerisation/taux_elongation)));
56      int T=G+max_int(3*nf,extremite-nf);
57
58      int L[nf];
59      int SumL=0;
60
61      for(int j=0; j<(nf-1); j++) {
62        L[j]=rand_unif(max_int(3,B[j]),
63        T-max_int(3*(nf-(j+1)),SumB-B[j])-SumL);
64        SumL=SumL+L[j];
65        SumB=SumB-B[j]; }
66      L[nf-1]=T-SumL;
67      SumL=SumL+L[nf-1];
68
69      std::string sNewL;
70      std::string sNewB;
71      std::string sNewE;
72
```

```
      K1["B;"+sNewB+";"+sNewL+";"+sNewE+";"]
      =K0["B;"+sNewB+";"+sNewL+";"+sNewE+";"];
      K1["S;0;"+sNewL+";1;"]=K0["S;0;"+sNewL+";1;"];

      for (int j=0; j<nf; j++) {
        sNewB = std::to_string(B[j]);
        sNewL = std::to_string(L[j]);
        sNewE = std::to_string(B[j]+1);

        if (B[j] > 0) {
          K1["B;"+sNewB+";"+sNewL+";"+sNewE+";"]
          =K1["B;"+sNewB+";"+sNewL+";"+sNewE+";"]+1; }
        else {
          K1["S;0;"+sNewL+";1;"]=K1["S;0;"+sNewL+";1;"]+1; } }
      K1["M;0;1;0;"]=K0["M;0;1;0;"]+longueur-T; } }

  K1["event"]=1;
  return K1; }
```

3 - Helpers Functions

* Fonction qui renvoie la longueur à partir d'une chaine de caractères de la forme "T;b;l;e;".

```
int findL(std::string s) {
  int positionSep[4];
  int pospos=0;
  int pos=0;
  int L=0;

  while (pospos<4) {
    if(s[pos]==';') {
      positionSep[pospos]=pos;
      pospos=pospos+1;
      pos=pos+1; }
    else {
      pos=pos+1; } }
  for (int j=positionSep[1]+1; j<positionSep[2]; j++) {
    L=L*10+(s[j]-'0'); }
  return L; }
```

* Fonction qui renvoie le nombre de branchements à partir d'une chaine de caractères de la forme "T;b;l;e;".

```
int findB(std::string s) {
  int positionSep[4];
  int pospos=0;
  int pos=0;
  int B=0;

  while (pospos<4) {
    if(s[pos]==';') {
      positionSep[pospos]=pos;
      pospos=pospos+1;
      pos=pos+1; }
    else {
      pos=pos+1; } }
```

```
14   for (int j=positionSep[0]+1; j<positionSep[1]; j++) {
15     B=B*10+(s[j]-'0'); }
16   return B; }
```

❉ Fonction qui renvoie le nombre d'extrémités libres à partir d'une chaine de caractère de la forme "T;b;l;e;".

```
1  int findE(std::string s) {
2    int positionSep[4];
3    int pospos=0;
4    int pos=0;
5    int E=0;
6
7    while (pospos<4) {
8      if(s[pos]==';') {
9        positionSep[pospos]=pos;
10       pospos=pospos+1;
11       pos=pos+1; }
12     else {
13       pos=pos+1; } }
14   for (int j=positionSep[2]+1; j<positionSep[3]; j++) {
15     E=E*10+(s[j]-'0'); }
16   return E; }
```

Index

This index recalls the general notations used throughout the manuscript as well as the notations used in each chapter.

General notations

Chapter I

Parameters

Single Actin Polymer

Population of Actin Polymers

Chapter II

Ancestors

Distances

Distance Matrix

Generators

Metric Measure Spaces

Parameters

Polynomials

Tree-Valued Process

Chapter III

Parameters

Population of Actin Polymers

Bibliography

[mec] https://www.mechanobio.info/cytoskeleton-dynamics/what-is-the-cytoskeleton/what-are-actin-filaments.

[Par21] *The Paris Book of Scottish Country Dances. La Ville Lumière*, **2**, RSCDS Paris Branch, 2021.

[AJL+15] B. Alberts, A. Johnson, J. Lewis, D. Morgan, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell*, 6th ed., Garland science, 2015.

[Ald78] D. Aldous, Stopping times and tightness, *The Annals of Probability* **6** no. 2 (1978), 335–340.

[AE07] R. Ananthakrishnan and A. Ehrlicher, The forces behind cell movement, *International journal of biological sciences* **3** no. 5 (2007), 303–317.

[BV17] V. Bansaye and V. Vatutin, On the survival probability for a class of subcritical branching processes in random environment, *Bernoulli* **23** no. 1 (2017), 58–88.

[BSK+02] J. Bear, T. Svitkina, M. Krause, D. Schafer, J. Loureiro, G. Strasser, I. Maly, O. Chaga, J. Cooper, and G. e. a. Borisy, Antagonism between Ena/VASP proteins and actin filament capping regulates fibroblast motility, *Cell* **109** (2002), 509–521.

[BL16] M. Benaïm and C. Lobry, Lotka–volterra with randomly fluctuating environments or "How switching between beneficial environments can make survival harder", *The Annals of Applied Probability* **26** no. 6 (2016), 3754–3785.

[BMB+07] J. Berro, A. Michelot, L. Blanchoin, D. R. Kovar, and J.-L. Martiel, Attachment conditions control actin filament buckling and the production of forces, *Biophysical journal* **92** no. 7 (2007), 2546–2558.

[Bil99] P. Billingsley, *Convergence of probability measures*, second ed., Wiley Series in Probability and Statistics, John Wiley and Sons Inc., 1999.

[BODM04] M. Bindschadler, E. A. Osborn, C. F. Dewey, and J. L. McGrath, A mechanistic model of the actin cycle., *Biophysj* **86** no. 5 (2004), 2720–2739.

[Bin10] M. Bindschadler, Modeling actin dynamics., *Wiley interdisciplinary reviews. Systems biology and medicine* **2** no. 4 (2010), 481–488.

[BBPSP14] L. Blanchoin, R. Boujemaa-Paterski, C. Sykes, and J. Plastino, Actin dynamics, architecture, and mechanics in cell motility, *Physiological reviews* **94** no. 1 (2014), 235–263.

[BR19] V. Boeuf and P. Robert, A stochastic analysis of a network with two levels of service, *Queueing Systems* **92** no. 3-4 (2019), 203–232.

[BCB+14] T. A. Burke, J. R. Christensen, E. Barone, C. Suarez, V. Sirotkin, and D. R. Kovar, Homeostatic actin cytoskeleton networks are regulated by assembly factor competition for monomers., *Current biology : CB* **24** no. 5 (2014), 579–585.

[CLZ91] M. CANO, D. LAUFFENBURGER, and S. ZIGMOND, Kinetic analysis of F-actin depolymerization in polymorphonuclear leukocyte lysates indicates that chemoattractant stimulation increases actin filament number without altering the filament length distribution, *J. Cell Biol.* **115** (1991), 677–687.

[CS17] M.-F. CARLIER and S. SHEKHAR, Global treadmilling coordinates actin turnover and controls the size of actin networks, *Nature Reviews Molecular Cell Biology* **18** no. 6 (2017), 389.

[CWC04] A. CARLSSON, M. WEAR, and J. COOPER, End versus side branching by arp2/3 complex, *Biophysical journal* **86** no. 2 (2004), 1074–1081.

[Car01] A. E. CARLSSON, Growth of branched actin networks against obstacles, *Biophysical journal* **81** no. 4 (2001), 1907–1923.

[CP11] Q. CHEN and T. D. POLLARD, Actin filament severing by cofilin is more important for assembly than constriction of the cytokinetic contractile ring, *Journal of Cell Biology* **195** no. 3 (2011), 485–498.

[CEK94] G. CIVELEKOGLU and L. EDELSTEIN-KESHET, Modelling the dynamics of F-actin in the cell, *Bulletin of Mathematical Biology* **56** no. 4 (1994), 587–616.

[CPGP+21] V. COSTACHE, S. PRIGENT GARCIA, C. N. PLANCKE, J. LI, S. BEGNAUD, S. K. SUMAN, A.-C. REYMANN, T. KIM, and F. B. ROBIN, Rapid assembly of a polar network architecture drives efficient actomyosin contractility, *Available at SSRN 3780278* (2021).

[DG19] A. DEPPERSCHMIDT and A. GREVEN, Tree-valued feller diffusion, *arXiv preprint arXiv:1904.02044* (2019).

[DGP12] A. DEPPERSCHMIDT, A. GREVEN, and P. PFAFFELHUBER, Tree-valued fleming–viot dynamics with mutation and selection, *The Annals of Applied Probability* **22** no. 6 (2012), 2560–2615.

[EKE98] L. EDELSTEIN-KESHET and G. B. ERMENTROUT, Models for the length distributions of actin filaments: I. Simple polymerization and fragmentation., *Bulletin of Mathematical Biology* **60** no. 3 (1998), 449–475.

[EEK98] G. B. ERMENTROUT and L. EDELSTEIN-KESHET, Models for the length distributions of actin filaments: II. Polymerization and fragmentation by gelsolin acting together., *Bulletin of Mathematical Biology* **60** no. 3 (1998), 477–503.

[Eth00] A. ETHERIDGE, *An introduction to superprocesses*, no. 20, American Mathematical Soc., 2000.

[EK86] S. N. ETHIER and T. G. KURTZ, Markov Processes: Characterization and Convergence., *John Wiley & Sons Inc.* (1986).

[EPW06] S. N. EVANS, J. PITMAN, and A. WINTER, Rayleigh processes, real trees, and root growth with re-grafting, *Probability Theory and Related Fields* **134** no. 1 (2006), 81–126.

[EW06] S. N. EVANS and A. WINTER, Subtree prune and regraft: a reversible real tree-valued markov process, *The Annals of Probability* **34** no. 3 (2006), 918–961.

[Glö13] P. K. GLÖDE, Dynamics of genealogical trees for autocatalytic branching processes, (2013).

[GPW09] A. GREVEN, P. PFAFFELHUBER, and A. WINTER, Convergence in distribution of random metric measure spaces (λ-coalescent measure trees), *Probability Theory and Related Fields* **145** no. 1 (2009), 285–322.

[GPW13] A. Greven, P. Pfaffelhuber, and A. Winter, Tree-valued resampling dynamics martingale problems and applications, *Probability Theory and Related Fields* **155** no. 3-4 (2013), 789–838.

[Gro99] M. Gromov, Metric structures for riemannian and non-riemannian spaces, with appendices by m. katz, p. pansu and s. semmes, *Progress in Mathematics* **152** (1999).

[HMPW08] B. Haas, G. Miermont, J. Pitman, and M. Winkel, Continuum tree asymptotics of discrete fragmentations and applications to phylogenetic models, *The Annals of Probability* **36** no. 5 (2008), 1790–1837.

[HNM+09] A. A. Halavatyi, P. V. Nazarov, S. Medves, M. Van Troys, C. Ampe, M. Yatskou, and E. Friederich, An integrative simulation model linking major biochemical reactions of actin-polymerization to structural properties of actin filaments, *Biophysical chemistry* **140** no. 1-3 (2009), 24–34.

[HW14] H. He and M. Winkel, Invariance principles for pruning processes of galton-watson trees, *arXiv preprint arXiv:1409.1014* (2014).

[HO11] J. Hu and H. G. Othmer, A theoretical analysis of filament length fluctuations in actin and other polymers, *Journal of Mathematical Biology* **63** no. 6 (2011), 1001–1049.

[Jak86] A. Jakubowski, On the skorokhod topology, in *Annales de l'IHP Probabilités et statistiques*, **22**, 1986, pp. 263–285.

[Kov06] D. R. Kovar, Molecular details of formin-mediated actin assembly, *Current Opinion in Cell Biology* **18** no. 1 (2006), 11–17.

[KHM+06] D. R. Kovar, E. S. Harris, R. Mahaffy, H. N. Higgs, and T. D. Pollard, Control of the Assembly of ATP- and ADP-Actin by Formins and Profilin, *CELL* **124** no. 2 (2006), 423–435.

[LL07] T. Lecuit and P.-F. Lenne, Cell surface mechanics and the control of cell shape, tissue patterns and morphogenesis, *Nature reviews Molecular cell biology* **8** no. 8 (2007), 633.

[LGD12] M. Lenz, M. L. Gardel, and A. R. Dinner, Requirements for contractility in disordered cytoskeletal bundles, *New journal of physics* **14** no. 3 (2012), 033037.

[MB01] I. V. Maly and G. G. Borisy, Self-organization of a propulsive actin network as an evolutionary process, *Proceedings of the National Academy of Sciences* **98** no. 20 (2001), 11324–11329.

[MSC+20] B. Martins, S. Sorrentino, W.-L. Chung, M. Tatli, O. Medalia, and M. Eibauer, Revealing the polarity of actin filaments by cryo-electron tomography, *BioRxiv Preprint* (2020).

[MEK02a] A. Mogilner and L. Edelstein-Keshet, Regulation of actin dynamics in rapidly moving cells: a quantitative analysis., *Biophysj* **83** no. 3 (2002), 1237–1258.

[MEK02b] A. Mogilner and L. Edelstein-Keshet, Regulation of actin dynamics in rapidly moving cells: a quantitative analysis, *Biophysical journal* **83** no. 3 (2002), 1237–1258.

[MLH+17] L. Mohapatra, T. J. Lagny, D. Harbage, P. R. Jelenkovic, and J. Kondev, The limiting-pool mechanism fails to control the size of multiple organelles, *Cell systems* **4** no. 5 (2017), 559–567.

[MBS+07] D. A. MOULDING, M. P. BLUNDELL, D. G. SPILLER, M. R. WHITE, G. O. CORY, Y. CALLE, H. KEMPSKI, J. SINCLAIR, P. J. ANCLIFF, C. KINNON, and OTHERS, Unregulated actin polymerization by wasp causes defects of mitosis and cytokinesis in x-linked neutropenia, *The Journal of experimental medicine* **204** no. 9 (2007), 2213–2224.

[MOLG15] M. MURRELL, P. W. OAKES, M. LENZ, and M. L. GARDEL, Forcing cells into shape: the mechanics of actomyosin contractility, *Nature reviews Molecular cell biology* **16** no. 8 (2015), 486–498.

[NSK08] E. M. NEIDT, C. T. SKAU, and D. R. KOVAR, The cytokinesis formins from the nematode worm and fission yeast differentially mediate actin filament assembly, *Journal of Biological Chemistry* **283** no. 35 (2008), 23872–23883.

[NDHH05] S. NICHOLSON-DYKSTRA, H. N. HIGGS, and E. S. HARRIS, Actin dynamics: growth from dendritic branches, *Current Biology* **15** no. 9 (2005), R346–R357.

[PP15] P. PFAFFELHUBER and L. POPOVIC, How spatial heterogeneity shapes multiscale biochemical reaction network dynamics, *Journal of the Royal Society Interface* **12** no. 104 (2015), 20141106.

[Pol86] T. D. POLLARD, Rate constants for the reactions of atp-and adp-actin with the ends of actin filaments., *The Journal of cell biology* **103** no. 6 (1986), 2747–2754.

[PB09] T. D. POLLARD and J. BERRO, Mathematical models and simulations of cellular processes based on actin filaments, *Journal of Biological Chemistry* **284** no. 9 (2009), 5433–5437.

[PC09] T. D. POLLARD and J. A. COOPER, Actin, a central player in cell shape and movement, *science* **326** no. 5957 (2009), 1208–1212.

[Reb80] R. REBOLLEDO, Sur l'existence de solution à certains problèmes de semimartingales, *CR Acad Sci Paris* **290** (1980).

[RS17] P. ROBERT and W. SUN, On the asymptotic distribution of nucleation times of polymerization processes, *arXiv preprint arXiv:1712.08347* (2017).

[RVGV21] F. ROBIN, A. VAN GORP, and A. VÉBER, The role of mode switching in a population of actin polymers with constraints, *Journal of Mathematical Biology* **82** no. 3 (2021), 1–43.

[RLCD+04] S. ROMERO, C. LE CLAINCHE, D. DIDRY, C. EGILE, D. PANTALONI, and M.-F. CARLIER, Formin is a processive motor that requires profilin to accelerate actin assembly and associated atp hydrolysis, *Cell* **119** no. 3 (2004), 419–429.

[RWH+15] J. D. ROTTY, C. WU, E. M. HAYNES, C. SUAREZ, J. D. WINKELMAN, H. E. JOHNSON, J. M. HAUGH, D. R. KOVAR, and J. E. BEAR, Profilin-1 Serves as a Gatekeeper for Actin Assembly by Arp2/3-Dependent and -Independent Pathways., *Developmental Cell* **32** no. 1 (2015), 54–67.

[SJC96] D. A. SCHAFER, P. B. JENNINGS, and J. A. COOPER, Dynamics of capping protein and actin assembly in vitro: uncapping barbed ends by polyphosphoinositides., *The Journal of cell biology* **135** no. 1 (1996), 169–179.

[SM01] D. SEPT and J. A. MCCAMMON, Thermodynamics and kinetics of actin filament nucleation, *Biophysical journal* **81** no. 2 (2001), 667–674.

[SHA95] J. Small, M. Herzog, and K. Anderson, Actin filament organization in the fish keratocyte lamellipodium, *J. Cell Biol.* **129** (1995), 1275–1286.

[SCB+15] C. Suarez, R. T. Carroll, T. A. Burke, J. R. Christensen, A. J. Bestul, J. A. Sees, M. L. James, V. Sirotkin, and D. R. Kovar, Profilin Regulates F-Actin Network Homeostasis by Favoring Formin over Arp2/3 Complex., *Developmental Cell* **32** no. 1 (2015), 43–53.

[SK16] C. Suarez and D. R. Kovar, Internetwork competition for monomers governs actin cytoskeleton organization, *Nature Reviews Molecular Cell Biology* **17** no. 12 (2016), 799.

[SVMB97] T. Svitkina, A. Verkhovsky, K. McQuade, and G. Borisy, Analysis of the actin-myosin II system in fish epidermal keratocytes: mechanism of cell body translocation, *J Cell Biol.* **139** no. 2 (1997), 397–415.

[WP05] J.-Q. Wu and T. D. Pollard, Counting cytokinesis proteins globally and locally in fission yeast, *Science* **310** no. 5746 (2005), 310–314.

[XCP99] J. Xu, J. Casella, and T. Pollard, Effect of capping protein, CapZ, on the length of actin filaments and mechanical properties of actin filament networks, *Cell Motil. Cytoskeleton* **42** (1999), 73–81.

[YB09] E. G. Yarmola and M. R. Bubb, How depolymerization can promote polymerization: the case of actin and profilin, *BioEssays* **31** no. 11 (2009), 1150–1160.

[YDB08] E. G. Yarmola, D. A. Dranishnikov, and M. R. Bubb, Effect of Profilin on Actin Critical Concentration: A Theoretical Analysis, *Biophysj* **95** no. 12 (2008), 5544–5573.

ECOLE
DOCTORALE
DE MATHEMATIQUES
HADAMARD

Titre : Modélisation multi-échelle de la dynamique intra-cellulaire de l'actine

Mots clés : Modélisation mathématique de l'actine, Processus de Markov à valeurs mesure, Processus à valeurs arbre, Théorèmes de convergence, Théorème d'homogénéisation, Espaces métriques mesurés

Résumé : Cette thèse porte sur la compréhension de la dynamique du réseau de polymères d'actine constituant l'enveloppe mécanique d'une cellule. Nous avons élaboré différents modèles, prenant en compte différentes échelles, afin de comprendre les différents mécanismes.
Les propriétés mécaniques des cellules embryonnaires, responsables d'un grand nombre de comportements cellulaires sont largement définies par la dynamique d'un réseau de polymères, le cortex d'actomyosine. Cette dynamique est soumise à une forte stochasticité, encore peu considérée dans les modèles existants. Afin de comprendre comment les propriétés mécaniques des cellules émergent de la biochimie de l'actine, il faut comprendre comment la dynamique d'assemblage de ces polymères fonctionne dans la cellule, et ce à différentes échelles.
La première partie de cette thèse est consacrée à l'étude de la dynamique d'élongation-raccourcissement de l'actine.
Dans un premier temps, nous considérons un seul polymère d'actine. La dynamique d'élongation-raccourcissement des polymères est entièrement décrite par un processus markovien de sauts. Afin d'obtenir un comportement moyen, nous avons étudié la limite de champs moyen. Dans un second temps, nous nous plaçons à l'échelle d'une population de polymères. La dynamique de la population est cette fois décrite par un processus de Markov à valeurs mesure. L'étude de la limite en grande population nous permet d'obtenir une distribution des tailles des polymères mais surtout de constater qu'il existe une compétition entre les polymères pour les monomères libres.
La deuxième partie se concentre sur l'étude de la dynamique de branchement d'un polymère d'actine. Afin de modéliser un polymère ramifié, nous utilisons le formalisme des espaces métriques mesurés. Cette partie a pour but de mieux comprendre la structure des polymères ramifiés.La dynamique du polymère ramifié est alors entièrement décrite par un processus à valeurs arbre. Nous avons prouvé que le processus décrivant le nombre d'extrémités (renormalisé) converge en distribution vers une diffusion de Feller.
La troisième partie est dédiée à l'exploration numérique des différents modèles.

Title : Multi-Scale Modelling of the Intra-Cellular Actin Dynamics

Keywords : Mathematical modelling of actin, Measure-valued Markov processes, Tree-valued processes, Convergence theorems, Homogenisation theorem, Metric measure spaces

Abstract : This thesis focuses on understanding the dynamics of the actin polymer network constituting the mechanical envelope of a cell. We have developed different mathematical models, considering different scales, in order to understand the different mechanisms.
The mechanical properties of embryonic cells, which are involved in many cellular behaviours, are largely defined by the dynamics of a network of polymers, the actomyosin cortex. These dynamics are subject to a high degree of stochasticity, which is not yet fully considered in existing models. In order to understand how the mechanical properties of cells stem from actin biochemistry, it is necessary to understand how the assembly dynamics of these polymers operate in the cell, and this at different scales.
The first part of this thesis is focused on the study of the elongation-shortening dynamics of actin. Initially, we consider a single actin polymer. The elongation-shortening dynamics of the polymers is then entirely described by a jump Markov process. In order to obtain an average behaviour, we studied the average field limit. Secondly, we consider a population of polymers. The dynamics of the population is this time described by a measure-valued Markov process. The study of the large population limit allows us to obtain a distribution of polymer lengths and specially to observe that there is competition between the polymers for free monomers.
The second part deals with the study of the branching dynamics of an actin polymer. In order to model a branched polymer, we use the formalism of metric measure spaces. The aim of this part is to better understand the structure of branched polymers. The dynamics of the branched polymer is then fully described by a tree-valued process. We proved that the process describing the number of extremities (renormalized) converges in distribution to a Feller diffusion.
The third part is focused on numerical exploration of the different models.

Institut Polytechnique de Paris
91120 Palaiseau, France