Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)
Факультет инноваций и высоких технологий
Кафедра «Анализ данных»

Вольхин Артем Васильевич

Влияние микротрубочек на скорость образования стрессовых гранул

010400 — Прикладная математика и информатика

Выпускная квалификационная работа бакалавра

Научный руководитель:
Шпильман Алексей Александрович

Москва
2013

Содержание

1 Введение

Стрессовые гранулы - скопления РНП в цитоплазме живых клеток, наблюдающиеся после воздействия стрессовых факторов, например, теплового шока, окислительного стресса, ультрафиолетового облучения. В живых клетках образование стрессовых гранул занимает порядка 10-20 минут. Размеры стрессовых гранул составляют обычно от 20 нм до нескольких мкм. Столь быстрое образование нарушает законы затрудненной диффузии, согласно которым большие гранулы практически неподвижны.

Роль микротрубочек в образовании стрессовых гранул до сих пор не до конца изучена. Существует предположение, что без микротрубочек образование стрессовых гранул невозможно, однако существуют и другие предположения. В частности, что микротрубочки ускоряют образование стрессовых гранул. Одним из механизмов может быть осуществление транспорта компонентов стрессовых гранул микротрубочками, облегчающий их встречу в пространстве клетки. Для доказательства этого механизма необходимо количественное изучение динамики сборки стрессовых гранул в присутствии и отсутствии микротрубочек в клетке и компьютерное моделирование такой системы.

Изучение числа, расположения и движения стрессовых гранул при различных условиях, а также взаимодействия со структурой микротрубочек в клетке, является основной задачей исследования стрессовых гранул. Важным для понимания динамики стрессовых гранул является определение их скорости, типа движения и взаимодействия гранул между собой.

Целью данной работы является построение модели движения и взаимодействия стрессовых гранул между собой и с микротрубочками и проверка на этой модели достоверности имеющихся теоретических

формул. Выбор численного моделирования вместо изучения живых клеток с помощью флуоресцентной микроскопии обусловлен тем, что разрешающая способность микроскопа позволяет наблюдать за частица размером не менее 100 нм, в то время как радиусы стрессовых гранул могут быть от 20 нм. Модель представляет собой частицы, случайным образом перемещающиеся в некотором объеме, и сеть микротрубочек, вдоль которых они могут скользить.

2 Обзор литературы

2.1 Броуновское движение

Броуновское движение — беспорядочное движение микроскопических видимых, взвешенных в жидкости или газе частиц твердого вещества, вызываемое тепловым движением частиц жидкости или газа. Броуновское движение никогда не прекращается. Броуновское движение связано с тепловым движением, но не следует смешивать эти понятия. Броуновское движение является следствием и свидетельством существования теплового движения.

Броуновское движение — наиболее наглядное экспериментальное подтверждение представлений молекулярно-кинетической теории о хаотическом тепловом движении атомов и молекул. Если промежуток наблюдения достаточно велик, чтобы силы, действующие на частицу со стороны молекул среды, много раз меняли своё направление, то средний квадрат проекции её смещения на какую-либо ось (в отсутствие других внешних сил) пропорционален времени.

Количественное описание броуновского движения было предложено Альбертом Эйнштейном в 1905 г. Его теория содержит 2 основные части. Первая часть — формулировка уравнения диффузии — описывает связь коэффициента диффузии и среднеквадратичного смещения (mean squared displacement, MSD) броуновской частицы в одномерном случае:

$$\overline{x^2} = 2Dt \tag{2.1}$$

(для двумерного случая: $\overline{r^2} = 4Dt$, где $r^2 = x^2 + y^2$). Вторая — связывает коэффициент диффузии с измеряемыми физическими величинами [4]:

$$D = \frac{RT}{N} = \frac{1}{6\pi kP}. \tag{2.2}$$

Теория Эйнштейна была проверена и подтверждена опытами Жана Батиста Перрена в 1908–1909 гг. Справедливость формул была доказана для частиц с размерами от 0,212 мкм до 5,5 мкм [15].

2.2 Стрессовые гранулы

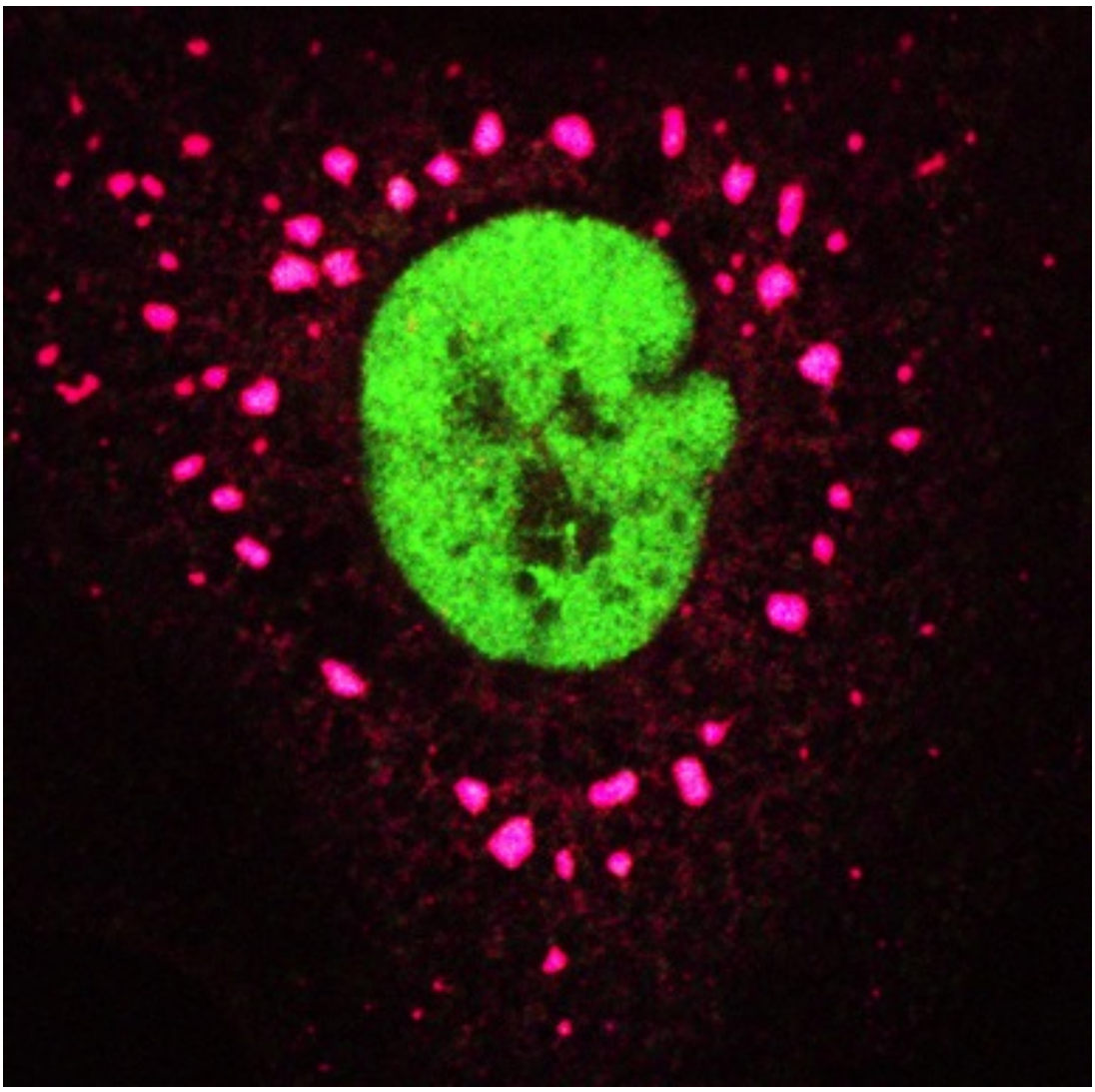


Рисунок 1. Внешний вид стрессовых гранул при флуоресцентной микроскопии.

Стрессовые гранулы - скопления РНП в цитоплазме живых клеток, наблюдающиеся после воздействия стрессовых факторов, например, теплового шока, окислительного стресса, ультрафиолетового облучения [12]. После прекращения стрессирующего воздействия стрессовая гранула

рассасывается. В состав стрессовых входят мРНК, РНК-связывающие белки, которые могут под действием факторов инициации трансляции elF1, elF3, elF4 использоваться для хранения, сортировки или других неизвестных процессов [1, 7]. Для включения в состав стрессовой гранул мРНА не должна обладать никакой особой структурой, а вот не входящие в ее состав содержат в себе специальную последовательность сигналов, например Hsp70 [9]. Предполагается, что образование стрессовых гранул является следствием ингибирования трансляции на стадии инициации, в частности, за счет фосфорилирования elF2a [10]. В результате этого фосфорилирования в клетке резко снижается количество тройственного комплекса, и на 5' концах мРНК собирается неканонический инициаторный комплекс, не содержащий тройственного комплекса. Получающиеся мРНП-частицы объединяются в цитоплазме в стрессовые гранулы.

Для минимизации ущерба мРНК, образование стрессовых гранул должны происходить достаточно быстро. В живых клетках оно занимает порядка 10-20 минут. Размеры стрессовых гранул составляют обычно от 20 нм до нескольких мкм. Однако, столь быстрое образование нарушает законы затрудненной диффузии, согласно которым большие гранулы практически неподвижны.

Стрессовые гранулы являются достаточно плотными структурами. Несмотря на достаточно продолжительную историю их исследования, попытки выделить их in vitro оказались неудачными. Стрессовые гранулы изучают в основном методами световой микроскопии, используя иммунофлуоресцентное окрашивание, экспрессию в клетках белков-маркеров стрессовых гранул, слитых с флуоресцентными белками, гибридизацию с ДНК-зондами. Изучение числа, расположения в клетке и движения стрессовых гранул при различных стрессовых условиях, а также взаимодействие со структурой микротрубочек в клетке, является

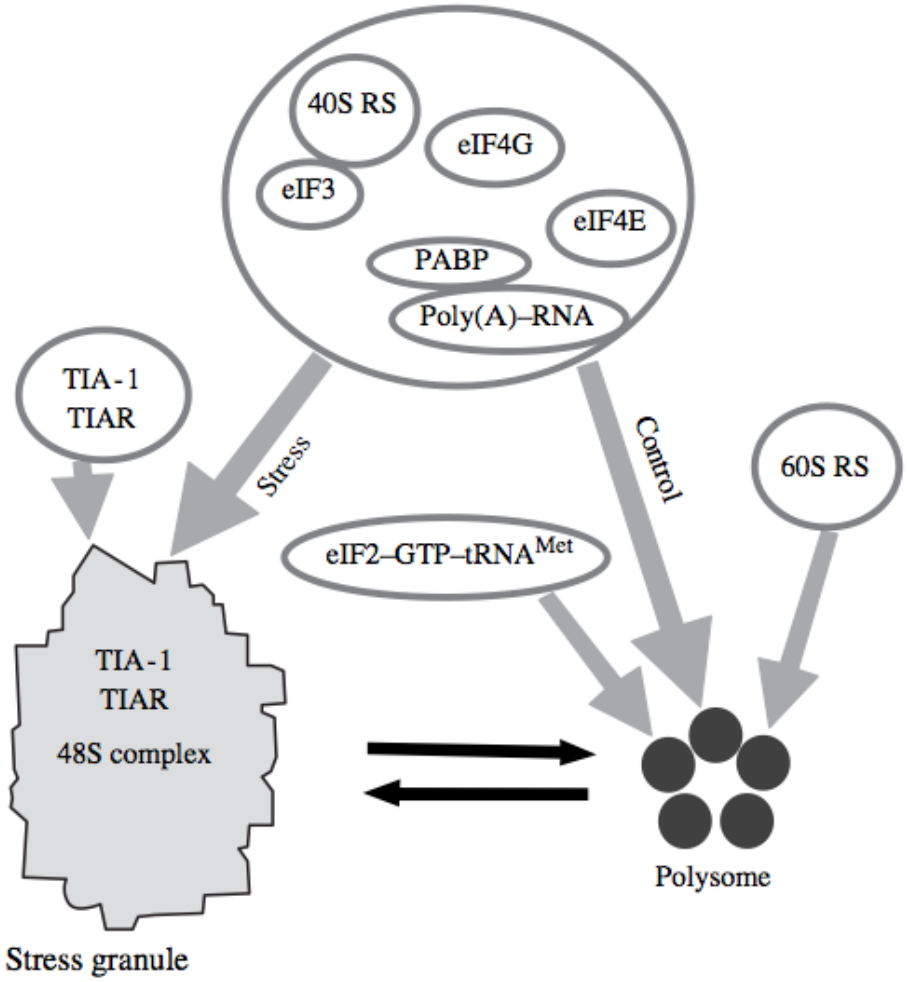


Рисунок 2. Образование стрессовой гранулы.

основной задачей исследования стрессовых гранул.

Роль микротрубочек в образовании стрессовых гранул до сих пор не до конца изучена. Существует предположение, что без микротрубочек образование стрессовых гранул не возможно, однако это кажется не верным. Более верным кажется предположение, что микротрубочки ускоряют образование стрессовых гранул. Каким образом также не совсем ясно, считается, что микротрубочки осуществляют транспорт компонентов стрессовых гранул, облегчающий их встречу в пространстве клетки. Для доказательства этого механизма необходимо количественное изучение динамики сборки стрессовых гранул в присутствии и отсутствии микротрубочек в клетке. Существенным при это является вопрос, достоверна ли оценка интенсивности образования стрессовых

гранул по доле клеток, в которых они образовались.

2.3 Микротрубочки

Микротрубочки — это белковые внутриклеточные структуры, входящие в состав цитоскелета. Микротрубочки представляют собой полые внутри цилиндры диаметром 25 нм. Длина их может быть от нескольких микрометров до, вероятно, нескольких миллиметров в аксонах нервных клеток. Их стенка образована димерами тубулина. Микротрубочки, подобно актиновым микрофиламентам, полярны: на одном конце происходит самосборка микротрубочки, на другом — разборка. В клетках микротрубочки играют роль структурных компонентов и участвуют во многих клеточных процессах, включая митоз, цитокинез и везикулярный транспорт.

Микротрубочки — это структуры, в которых 13 протофиламентов, состоящих из гетеродимеров α- и β- тубулина, уложены по окружности полого цилиндра. Внешний диаметр цилиндра около 25 нм, внутренний — около 15. Один из концов микротрубочки, называемый плюс-концом, постоянно присоединяет к себе свободный тубулин. От противоположного конца (минус-конца) тубулиновые единицы отщепляются. В образовании микротрубочки выделяют три фазы:

- замедленная фаза, или нуклеация. Это этап зарождения микротрубочки, когда молекулы тубулина начинают соединяться в более крупные образования. Такое соединение происходит медленнее, чем присоединение тубулина к уже собранной микротрубочке, поэтому фаза и называется замедленной;

- фаза полимеризации, или элонгация. Если концентрация свободного тубулина высока, его полимеризация происходит быстрее,

чем деполимеризация на минус-конце, за счет чего микротрубочка удлиняется. По мере её роста концентрация тубулина падает до критической и скорость роста замедляется вплоть до вступления в следующую фазу;

- фаза стабильного состояния. Деполимеризация уравновешивает полимеризацию, и рост микротрубочки останавливается.

Лабораторные исследования показывают, что сборка микротрубочек из тубулинов происходит только в присутствии гуанозинтрифосфата и ионов магния.

Микротрубочки в клетке используются в качестве «рельсов» для транспортировки частиц. По их поверхности могут перемещаться мембранные пузырьки и митохондрии. Транспортировку по микротрубочкам осуществляют белки, называемые моторными. Это высокомолекулярные соединения, состоящие из двух тяжёлых (массой около 300 кДа) и нескольких лёгких цепей. В тяжёлых цепях выделяют головной и хвостовой домены. Два головных домена связываются с микротрубочками и являются собственно двигателями, а хвостовые — связываются с органеллами и другими внутриклеточными образованиями, подлежащими транспортировке. Выделяют два вида моторных белков:

- цитоплазматические динеины
- кинезины

Динеины перемещают груз только от плюс-конца к минус-концу микротрубочки, то есть из периферийных областей клетки к центросоме. Кинезины, напротив, перемещаются к плюс-концу, то есть к клеточной периферии. Перемещение осуществляется за счёт энергии АТФ. Головные домены моторных белков для этого содержат АТФ-связывающие

участки. Помимо транспортной функции, микротрубочки формируют центральную структуру ресничек и жгутиков — аксонему. Типичная аксонема содержит 9 пар объединённых микротрубочек по периферии и две полных микротрубочки в центре. Из микротрубочек состоят также центриоли и веретено деления, обеспечивающее расхождение хромосом к полюсам клетки при митозе и мейозе. Микротрубочки участвуют в поддержании формы клетки и расположения органоидов (в частности, аппарата Гольджи) в цитоплазме клеток.

2.4 Материалы

В большинстве работ исследование микротрубочек проводится при помощи световой микроскопии, используя иммунофлуоресцентное окрашивание [14]. Исследования проводили на клетках HeLa (человеческая карцинома шейки матки), CV-1 (эпителий почки зеленой мартышки) и CHO (яичник китайского хомяка). Клетки выращивали на среде, содержащей 45% среды F12, 45% среды DMEM, 10% эмбриональной сыворотки крупного рогатого скота с добавлением гентамицина на пластиковой культуральной посуде при стандартных условиях (температура 37° C, содержание углекислого газа в среде –5%). Для экспериментов культивируемые клетки рассаживали на покровные стекла.

Для разборки клеточных микротрубочек в среду добавляли нокодазол до конечной концентрации 6 мкг/мл на 2 ч. Для индукции СГ использовали арсенит натрия (NaH_2AsO_3); максимальная конечная концентрация арсенита в среде культивирования составляла 250 мкМ (для клеток HeLa) и 625 мкМ (для клеток CV-1). Время воздействия арсенитом составляло 30–60 мин. Для расщепления стрессовых гранул использовали циклогексамид (CHX), а для разборки

микрофилламентов — летрункулин.

Клетки фиксировали в охлажденном до -20° C абсолютном метаноле. Для иммунофлуоресцентного окрашивания покровные стекла с клетками инкубировали в течение 30 мин с первыми антителами, разведенными на PBS, промывали два раза по 5–7 мин PBS и инкубировали в течение 30 мин со вторыми антителами, конъюгированными с флуорохромами. После инкубации со вторыми антителами стекла промывали три раза по 5–7 мин буфером PBS и заключали в Aqua Poly/Mount (Polysciences, США).

В работе использовали поликлональные кроличьи антитела к eIF3a (Шанина и др., 2001) и моноклональные антитела к тубулину DM-1A (Sigma, США), антитела против IgG кролика и IgG мыши, конъюгированные с тетраметилродаминизотиоцианатом (TRITC) или с флуоресцеинизотиоцианатом (FITC) (Molecular Probes, США).

Препараты клеток просматривали в микроскопе Axiovert-200M (Carl Zeiss, Германия) с объективами Planapo 63x и 100x. Фотографии (16-битовые черно- белые изображения) получали с помощью цифровой видеокамеры AxioCam HRc, управляемой программным обеспечением AxioVision 3.1 (Carl Zeiss Vision, Германия).

2.5 Сбор и анализ данных

Серии изображений сохранялись в виде 16-битных цифровых файлов и обрабатывались в приложении MetaMorph (Universal imaging). Стрессовые гранулы отслеживались при помощи Point-Tracking функции в MetaMorph. Данные импортировались в Excel для вычисления мгновенной скорости. Разработанная авторами программа вычисляла и рисовала график зависимости среднеквадратичного отклонения MSD от времени τ, согласно [16]. MSD вычислялось для каждой частицы в

каждый момент времени как

$$MSD(n\tau) = \frac{1}{N-n}\sum_{i=1}^{N-n}[(x((i+n)\tau) - x(i\tau))^2 + (y((i+n)\tau) - y(i\tau))^2]$$

На основе полученной зависимости вычислялся коэффициент диффузии

$$D_{app} = MSD(n\tau)/\Delta\tau$$

согласно [5].

Для выделения стрессовых гранул и определения посещенной области видео-кадры сперва нормализовались при помощи Bleaching Normalization плагина из ImageJ. Число стрессовых гранул и общая посещенная площадь вычислялись через Region Measurments и Integrated Morphometry из Metamorph и ImageJ Analyze Particles plug-in.

Временные серии FRAP анализировались в MetaMorph. Сырые данные флуоресцентной интенсивности сперва регулировались путем вычитания фона в каждый момент времени. Интенсивность каждой восстановленной гранулы нормализовалась к среднему уровню сигнала и по этим данным строились графики в Origin software (OriginalLab Corp.).

2.6 Движение, объединение и разъединение СГ

Было выяснено, что стрессовые гранулы в цитоплазме движутся, объединяются в большие гранулы и, наоборот, разъединяются. В течение первых 5–7 минут эксперимента число стрессовых гранул росло, после чего начало снижаться. После 10–12 минут их число стало постоянным. Уменьшение числа стрессовых гранул объясняется их объединением и формированием больших гранул. Двигаться гранулы начали с момента образования и продолжали даже через 150 минут после начала эксперимента. Существенной зависимости от температурной обработки

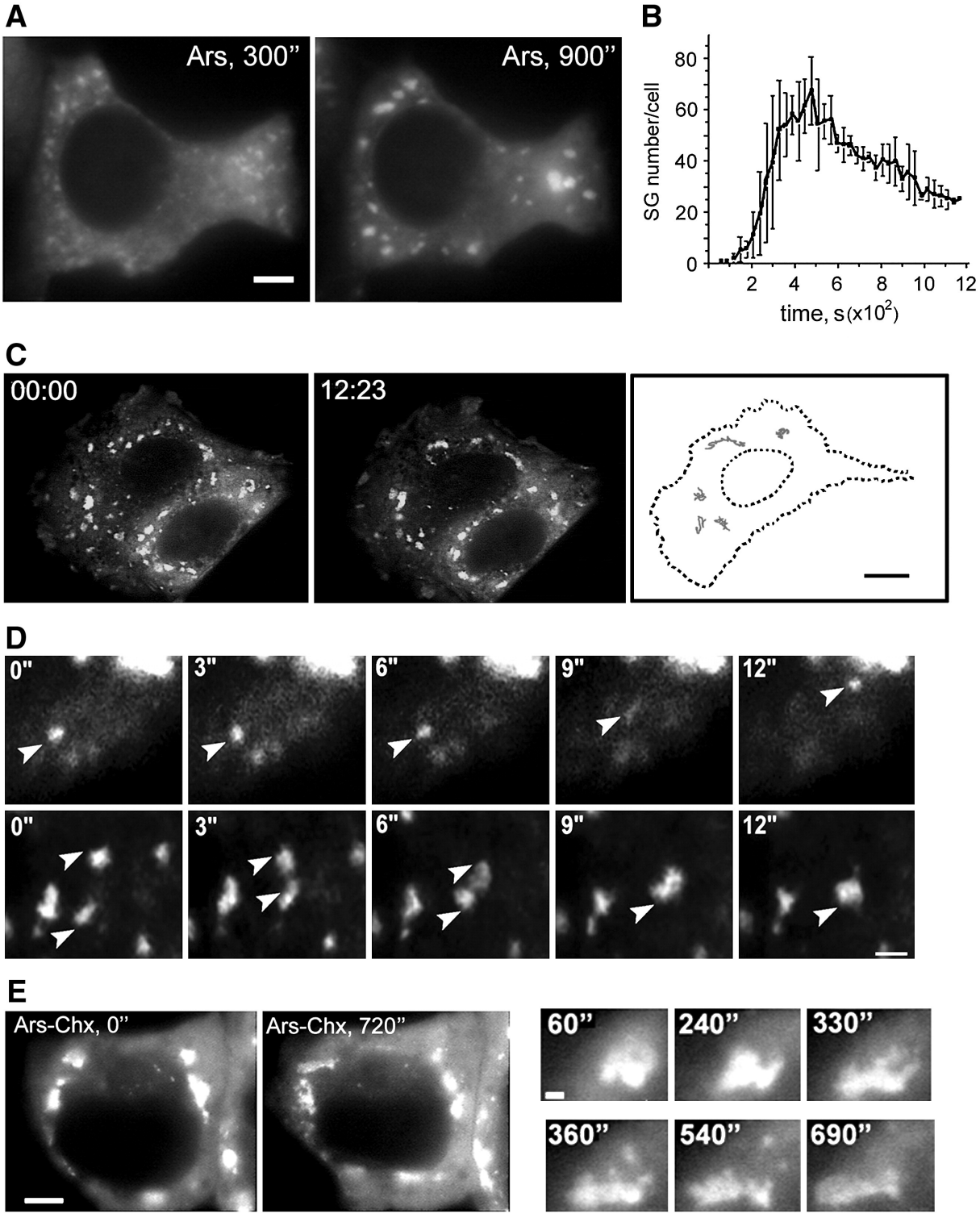


Рисунок 3. Движение, объединение и расщепление стрессовых гранул.

клеток выявлено не было. Многие гранулы хаотически двигались в цитоплазме, некоторые были практически неподвижны, в то время как другие преодолевали достаточно большие расстояния за короткие промежутки времени [18].

Объединение гранул происходило при их столкновении: две гранулы сливались в одну и продолжали движение уже как одна гранула. Расщепление происходило только при добавлении CHX следующим образом: гранула разделялась на части, которые разлетались; выглядело так, будто часть гранулы выдергивает какая-то внешняя сила. Без добавления CHX расщепления гранул не наблюдалось.

Для исключения связи наблюдаемых результатов с типом клеток (использовались клетки HeLa), были также проведены эксперименты с клетками CV-1 и CHO. Предполагается, что подвижность стрессовых гранул — это непосредственно их свойство, а не влияние определенного типа клеток.

2.7 Движение СГ за счет диффузии

Была исследована природа подвижности стрессовых гранул. Во-первых, была измерена мгновенная скорость (на 2–3 секундных интервалах). Рассматривались 12 клеток, 75 стрессовых гранул на 6524 временных интервалах. Около 10% гранул были неподвижны, у 8% скорость была меньше 0.2 мкм/с, максимальная скорость была порядка 0.7 мкм/с, а средняя (не учитывая неподвижные гранулы) — 0.1 мкм/с (рис. 4). Не было выявлено существенной зависимости скорости гранул от их размера. Изучив траектории движения за 3 минуты, можно найти среди них признаки ограниченной, затрудненной и обычной диффузии, а также направленного движения [17]. Среди наблюдаемых гранул 10% двигались направленно, 17% — согласно обычной диффузии, а 73% — затрудненной диффузии.

Посчитав MSD и D_{app} получили, что коэффициент диффузии со временем уменьшается, сходясь к ассимптотическому значению 1.9×10^{-3} мкм^2/с. Исследовав посещенную гранулами область получили,

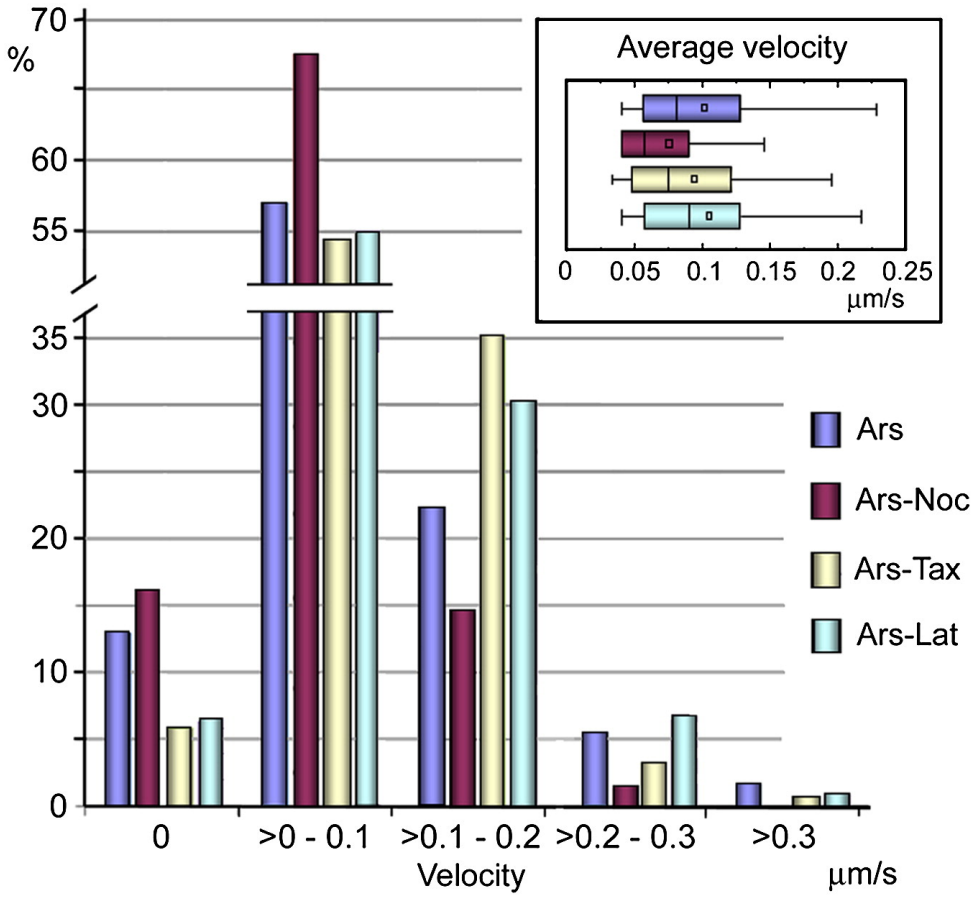


Рисунок 4. Распределение скоростей.

что за 4 минуты движения она в 3 раза превышает суммарный размер самих гранул. Предположительно, стрессовый гранулы способны собирать все подходящие компоненты с посещенной территории, такие как другие мелкие стрессовые гранулы.

2.8 Движение СГ вдоль микротрубочек

Плотная и хаотическая система микротрубочек в клетках HeLa не позволяет отслеживать движение стрессовых гранул вдоль микротрубочек. Для экспериментов были взяты клетки CHO с радиальной системой микротрубочек и наблюдали при помощи флюоресцентной микроскопии за живыми клетками. Наблюдения подтвердили движение стрессовых гранул вдоль микротрубочек и формирование скоплений. Их движение

было непостоянным: гранулы двигались, останавливались, продолжали движение в обратном направлении. Но при этом не отрывались от микротрубочек. Типичные размеры гранул составляли от 200 нм до 5 мкм.

При добавлении нокодазола для разрушения микротрубочек стрессовые гранулы не пропадали, более того, их количество увеличивалось. Из чего можно сделать вывод, что микротрубочки не являются необходимыми для образования стрессовых гранул. Отличием стало лишь то, что концетрация гранул стала более беспорядочная по всему объему цитоплазмы, исчезли сгустки.

2.9 Роль микротрубочек в движении СГ

После разрушения микротрубочек нокодазолом подвижность стрессовых гранул значительно снизилась. Так, только у 1.5% гранул скорость осталась более 0.2 мкм/с (рис. 5), а средняя скорость упала до 0.076 мкм/с. Движение уже 92% гранул стало напоминать затрудненную диффузию, тогда как доля гранул с ограниченной и обычной диффузией снизилась до 3% и 5% соответственно. Гранул с направленным движением не осталось вовсе. Коэффициент диффузии уменьшился более чем в три раз и стал равен 6×10^{-4} мкм^2/с.

Для изучения роли микротрубочек в движении стрессовых гранул микротрубочки были зафиксированы при помощи таксола. Скорость стрессовых гранул уменьшилась всего на 6%, из чего следует, что не самостоятельное движение микротрубочек увеличивало скорость стрессовых гранул, а сам факт наличия микротрубочек. Это легко объяснить, если считать, что стрессовые гранулы находят микротрубочку, прикрепляются к ней и дальше двигаются вдоль нее.

Проведя аналогичные эксперименты со стрессовыми гранулами

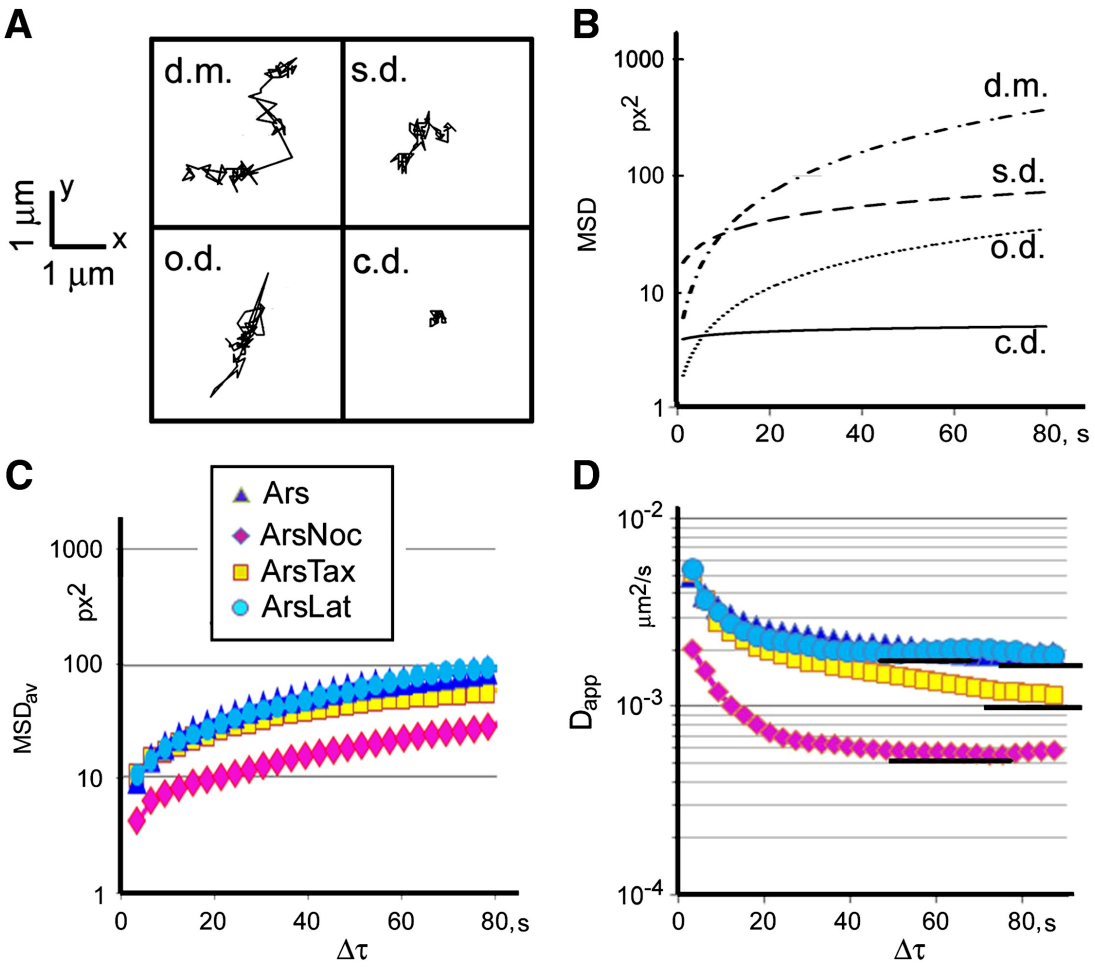


Рисунок 5. Анализ диффузионного движения. d.m. — directed motion; s.d. — simple diffusion; o.d. — obstructed diffusion; c.d. — confined diffusion.

и микрофиламентами, выяснилось, движение СГ слабо зависит от наличия или отсутствия микрофиламентов. Таким образом, именно микротрубочки играют ключевую роль в транспорте стрессовых гранул.

В других работах [3] утверждается, что движение миктротрубочек существенно влияет на подвижность стрессовых гранул. Согласно их наблюдениям, микрофилламенты припятствуют движению достаточно больших стрессовых гранул (типичное расстояние между микрофилламентами - 100 нм). В результате стрессовые большие гранулы остаются неподвижны и могут продолжать увеличиваться в размерах за счет слияния с более мелкими, столкнувшимися с ними. Однако движение

микротрубочек позволяет перемещать большие гранулы, застрявшие в сети микрофилламентов — при элонгации гранулы, находящиеся вблизи минус-конца «тянутся» за микротрубочкой. При этом предположение о движении стрессовых гранул вдоль микротрубочек не опровергается, но высказываются сомнения в эффективности такого метода движения.

2.10 Теоретические оценки

Для изучения модели движения стрессовых гранул интересным представляется таких величин, как среднее время столкновения между гранулами, время до столкновения стрессовой гранулы с микротрубочкой, время до столкновения между стрессовыми гранулами на одной микротрубочки и другие. работе [3] представлены теоретические оценки для этих величин.

Так, время t_g, требуемое для формирования стрессовой гранулы радиуса R равно

$$t_g = t_d \cdot (R/r)^{df}, \tag{2.3}$$

где $t_d \approx \eta/(K_B T C_{RNP}) \approx 1/(4\pi D r C_{RNP})$ — время между столкновениями mRPN, η — вязскость цитоплазмы, r и C_{RNP} — радиус и концетрация изначальных mRNP частиц, соответственно, D — коэффициент диффузии, $K_B T$ — тепловая энергия, а df — фрактальная размерность, равная ~ 1.8 [20]. В клетке цитоплазмы млекопитающих содержится $\sim$ 15,000–150,000 мРНА молекул, объем клетки порядка 2000 мкм^3, а значит концетрация мРНА 12–120 нМ. $r \simeq 10$ нм, $\eta \simeq 10^{-2}\ pascal/s$, $T = 37^\circ\ C$. При этих параметрах $t_d \sim 0.2s$, то есть для формирования стрессовой гранулы радиусом 100 нм и 1 мкм потребуется 12 с и 12 мин соответственно. В такой идеальной модели свободной диффузии образование больших стрессовых гранул легко объяснимо. Однаком в реальности существет сеть из микрофилламентов, которые

препятствуют перемещению гранул уже при радиусе порядка 50–100 нм.

Теперь рассмотрим модель, в которой присутствуют микротрубочки. Типичной ситуацией будет блуждание стрессовой гранулы в цитоплазме до столкновения с микротрубочкой, дальнейшее движиение вдоль микротрубочки и столкновение с другой гранулой. Получим оценки для времени до столкновения с микротрубочкой t_m:

$$t_m \sim \frac{1}{4\pi D \left((a+r)^2 L\right)^{1/3} C_{RNP}} \tag{2.4}$$

$$\frac{t_m}{t_d} \sim \left(\frac{r}{(1+a/r)^2 L}\right)^{1/3} \tag{2.5}$$

и после начала движения вдоль нее до столкновения с гранулой — t_s:

$$t_S \sim \frac{l_S^2}{D_S} \tag{2.6}$$

$$\frac{t_s}{t_d} \sim \frac{D}{D_s} r (C_{RNP})^{1/3} \tag{2.7}$$

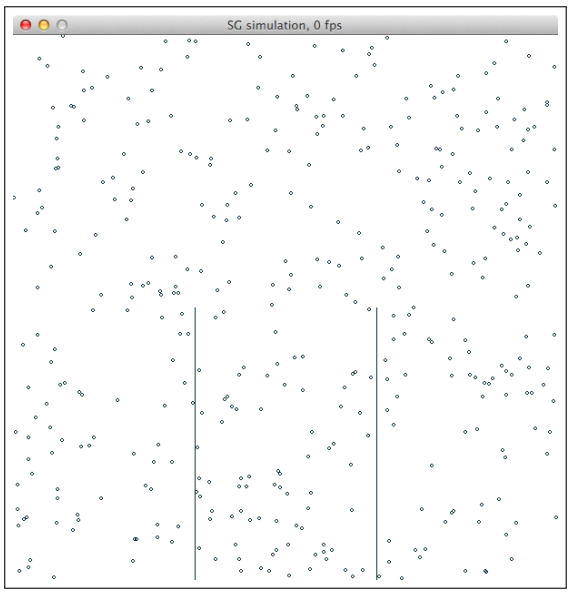
где a — радиус микротрубочки, L — ее длина, D и D_s — коэффициенты диффузии в цитоплазме и при движении вдоль микротрубочки, l_S — среднее расстояние между двумя частицами в цитоплазме.

3 Материалы и методы

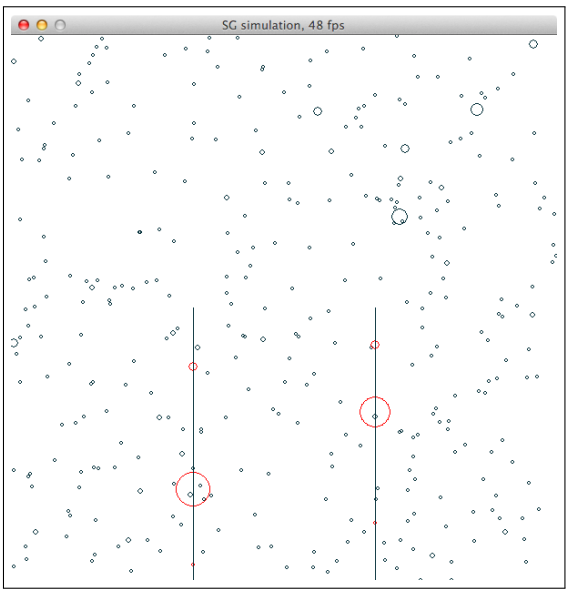
Для проведения исследований был реализован симулятор движения стрессовых гранул и их взаимодействия с микротрубочками. Модель представляет собой трехмерный замкнутый параллелепипед, в котором расположены неподвижные микротрубочки в подвижные стрессовые гранулы. Для реализации был выбран язык Python в силу удобства разработки и наглядности. Для визуализации использовались библиотеки pygame и pylab. В симуляторе имеется возможность настраивать такие параметры как:

- размеры ограничивающего параллелепипеда
- фрактальная размерность пространства
- начальное количество частиц и микротрубочек
- начальный радиус частиц
- размеры микротрубочек
- порог неподвижности, начиная с которого стрессовые гранулы перестают перемещаться
- коэффициенты диффузии (отдельно для движения в цитоплазме и по микротрубочкам)

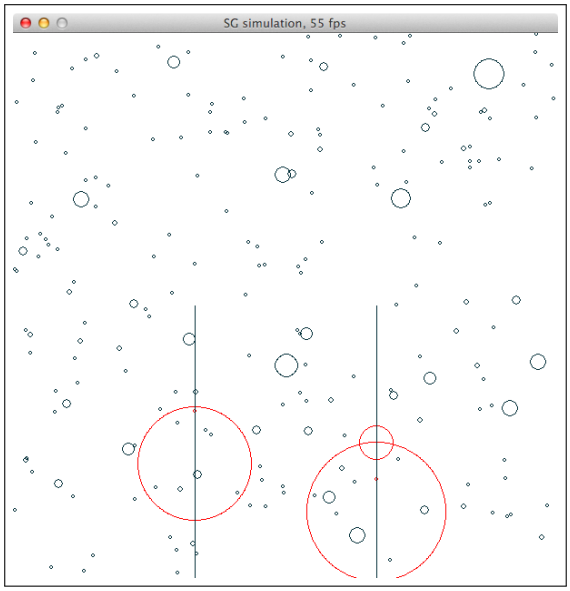
Микротрубочки в модели имеют вид вертикальных цилиндров с длиной, равной половине боковой стороны ограничивающего куба. В дальнейшем планируется усложнить модель, сделав более реалистичную структуру микротрубочек, например, представив их сеть в виде планарного графа. В таком случае, движение стрессовых гранул по ним будет возможно не только в одном направлении. При столкновении



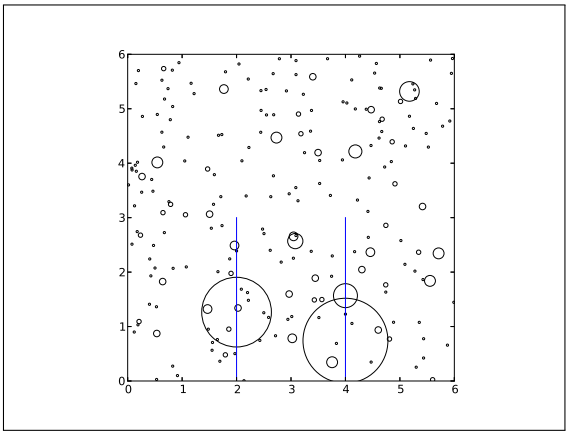
(a) В момент времени $t = 0$ с



(b) В момент времени $t = 300$ с



(c) В момент времени $t = 600$ с



(d) В момент времени $t = 600$ с, вывод через pyplot

Рисунок 6. Внешний вид визуализатора.

стрессовое гранулы с микротрубочкой они сцеплялись и в дальнейшем отсоединение не происходило, гранула начинала двигаться в одном из двух возможных направлений. Интересный вопрос, что делать при достаточно большом размере гранулы, когда она начинает касаться более одной микротрубочки. Согласно наблюдениям, гранула может двигаться вдоль любой из микротрубочек, что и объясняет сложность наблюдения за СГ в условиях запутанной сети. В данной модели считается, что

переход от одной микротрубочки к другой невозможен.

Случайное движение моделируется выбором шага по каждой оси координат согласно нормальному распределению со средним 0 и стандартным отклонением, зависящим от коэффициента диффузии следующим образом [13]:

$$\overline{x^2} = 6Dt$$

Влияние микрофилламентов учитывается в пороге подвижности — считается, что при радиусе больше некоторого порога, гранула застревает в сети микрофилламентов и прекращает самостоятельное движение. В то же время движение возможно при движении микротрубочек — раз гранула не может оторваться от микротрубочки, она невольно будет двигаться вслед за ней.

Была реализована платформа для проведение экспериментов. Для этого имеется возможность запуска серии симуляций с заданным набором меняющихся параметров. В системе предусмотрен вывод всех промежуточных результатов как на экран, так и в файл. Все результаты запусков сохраняются вместе с параметрами модели, при которых они были получены. Предусмотрен многократный запуск модели на одних и тех же входных данных для повышения статистической достоверности получаемых результатов. Система легко расширяема, внутри нее ведется журнал всех произошедших событий в хронологическом порядке для удобства анализа проведенной симуляции, на основе журнала возможен рассчет любых интересующих оценок.

4 Результаты и обсуждение

4.1 Частота столкновения между гранулами

Исследуем время столкновения между стрессовыми гранулами при свободном блуждании без участия микротрубочек. Для это запускали моделирование со следующими параметрами:

- размеры ограничивающего параллелепипеда: 6x6x6 мкм
- фрактальная размерность пространства: 1.8
- начальное количество частиц: 400
- количество микротрубочек: 0
- начальный радиус частиц: 20 нм
- порог неподвижности, начиная с которого стрессовые гранулы перестают перемещаться: 100 нм

Запускали при различных значениях коэффициента диффузии, для каждого значения запускали несколько (9) раз и брали значение медианы. Получили обратно пропорциональную зависимость (рис. 7).

Теперь рассмотрим зависимость времени столкновения между гранулами от концентрации стрессовых гранул или же от их количества. Согласно теоретическим расчетам зависимость опять же должна быть обратно пропорциональной, что подтверждается моделированием (рис. 8). Запускали при тех же параметрах, что и в предыдущем эксперименте.

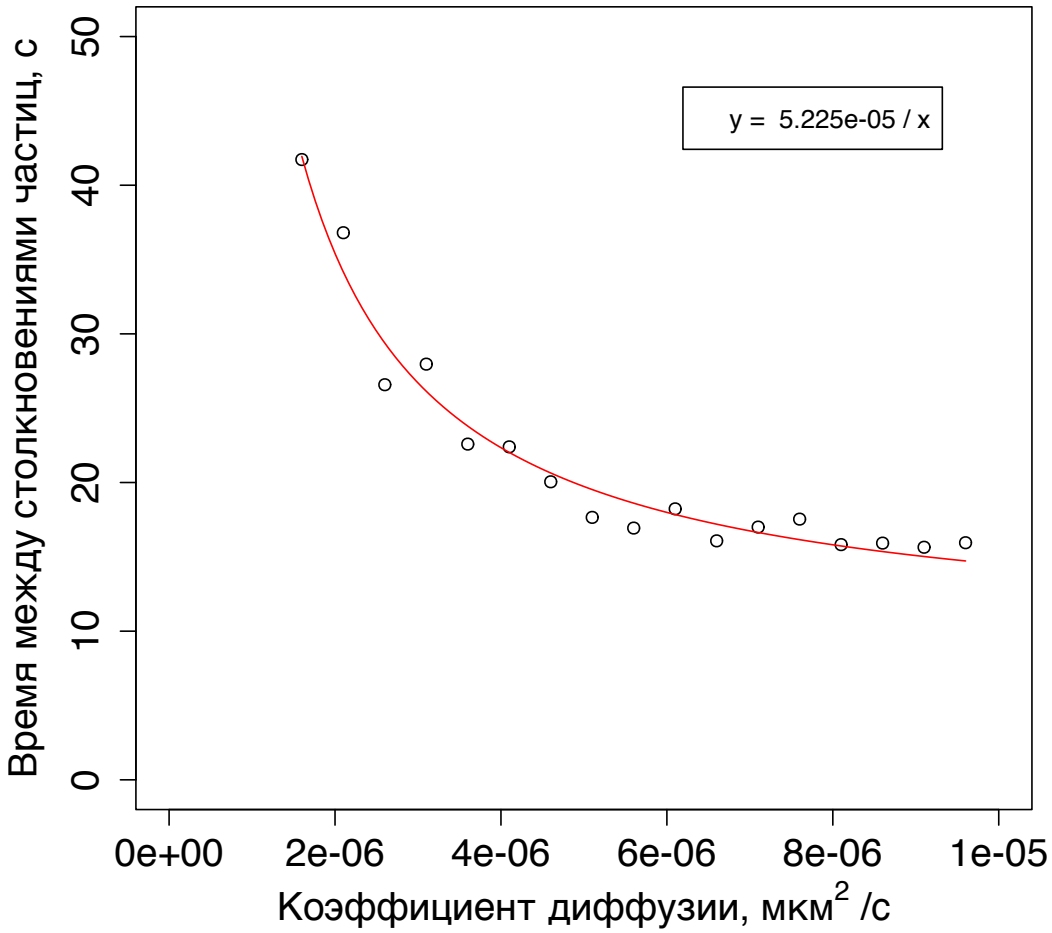


Рисунок 7. Зависимость времени столкновения между СГ от коэффициента диффузии.

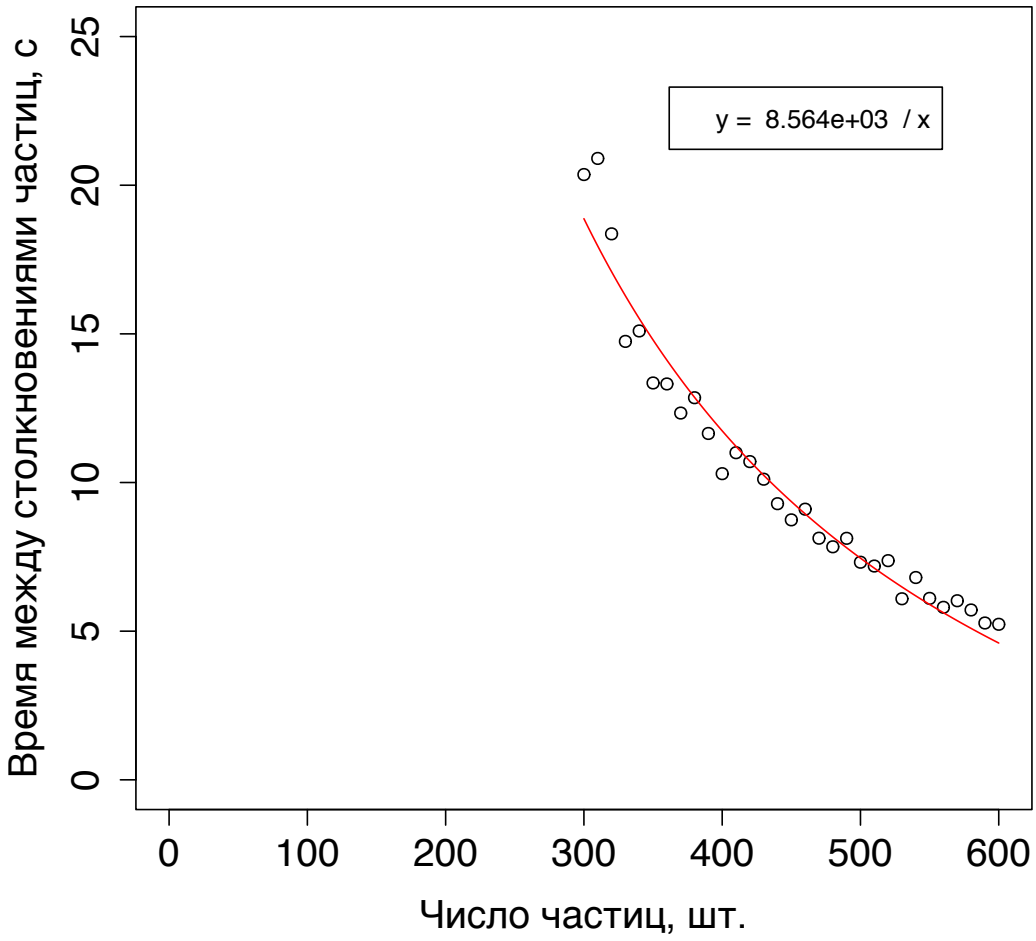


Рисунок 8. Зависимость времени столкновения между СГ от их концентрации.

4.2 Частота столкновения СГ с микротрубочками

Проверим выр. 2.4, а именно зависимость среднего времени между столкновениями СГ с микротрубочками от длины и радиуса микротрубочек. Для проверки модель запускалась при переменной длине и радиусе микротрубочек соответственно и фиксированных остальных параметрах. Моделировались 300 секунд движения 400 гранул. Для чистоты эксперимента было отключено слияние стрессовых гранул при столкновении.

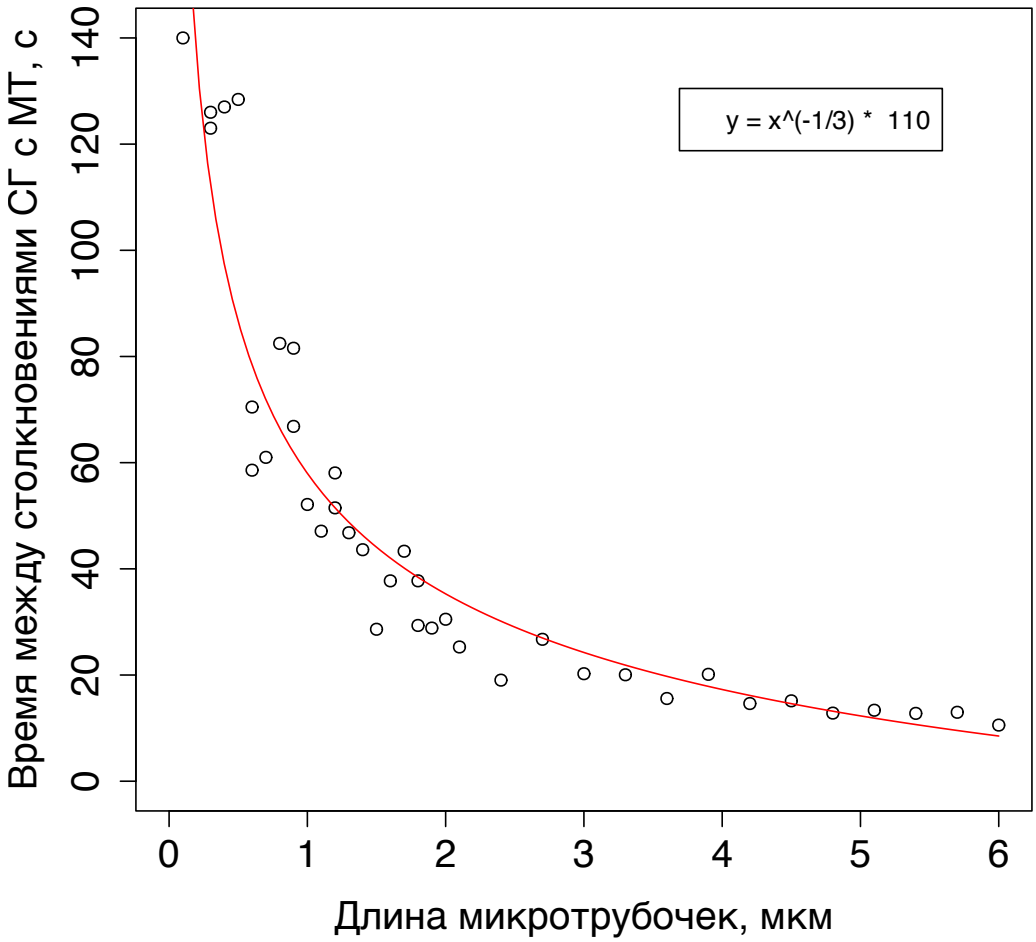


Рисунок 9. Зависимость времени столкновения СГ с микротрубочками от длины микротрубочек.

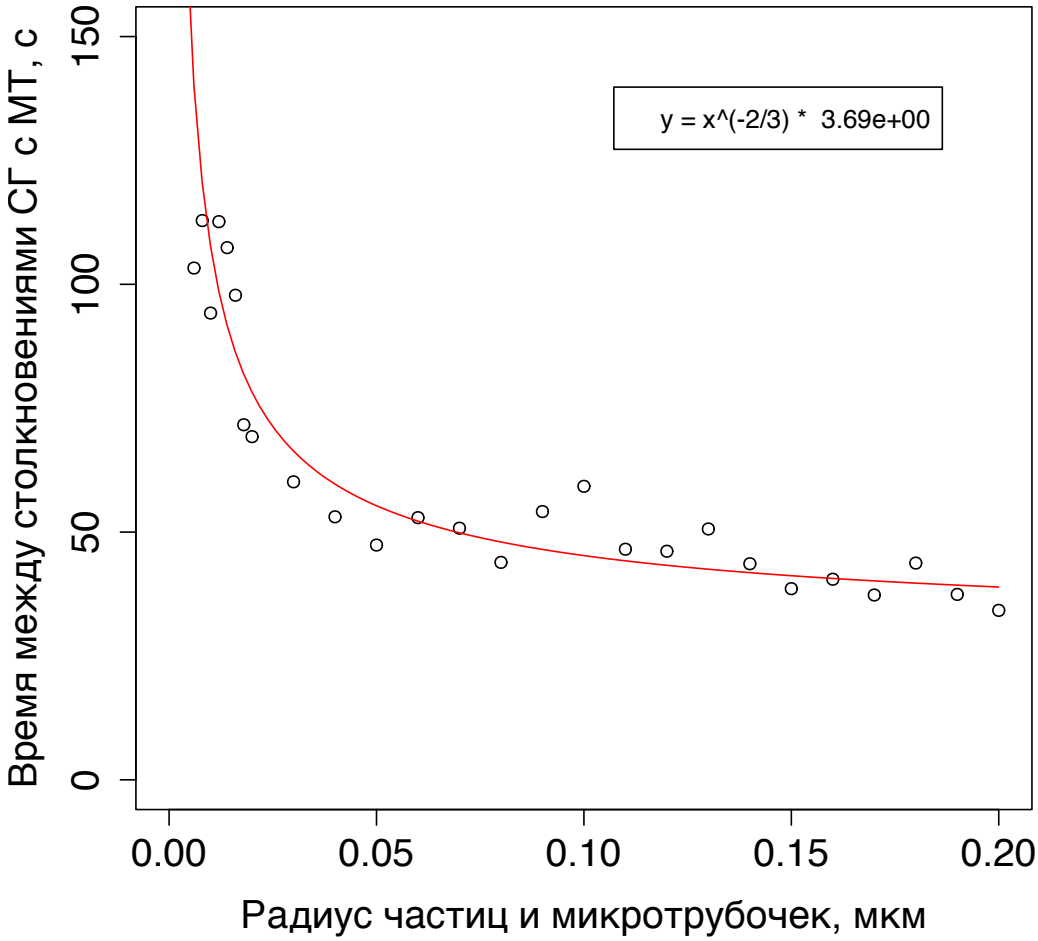


Рисунок 10. Зависимость времени столкновения СГ с микротрубочками от их радиуса.

4.3 Зависимость размера СГ от времени образования

Проверим выр. 2.3, а именно зависимость времени образования гранулы определенного размера от ее радиуса. Моделировались 300 секунд движения 400 гранул. Для чистоты эксперимента было отключено слияние стрессовых гранул при столкновении. Получили зависимость виду $y \sim x^{1.8}$, как и ожидалось.

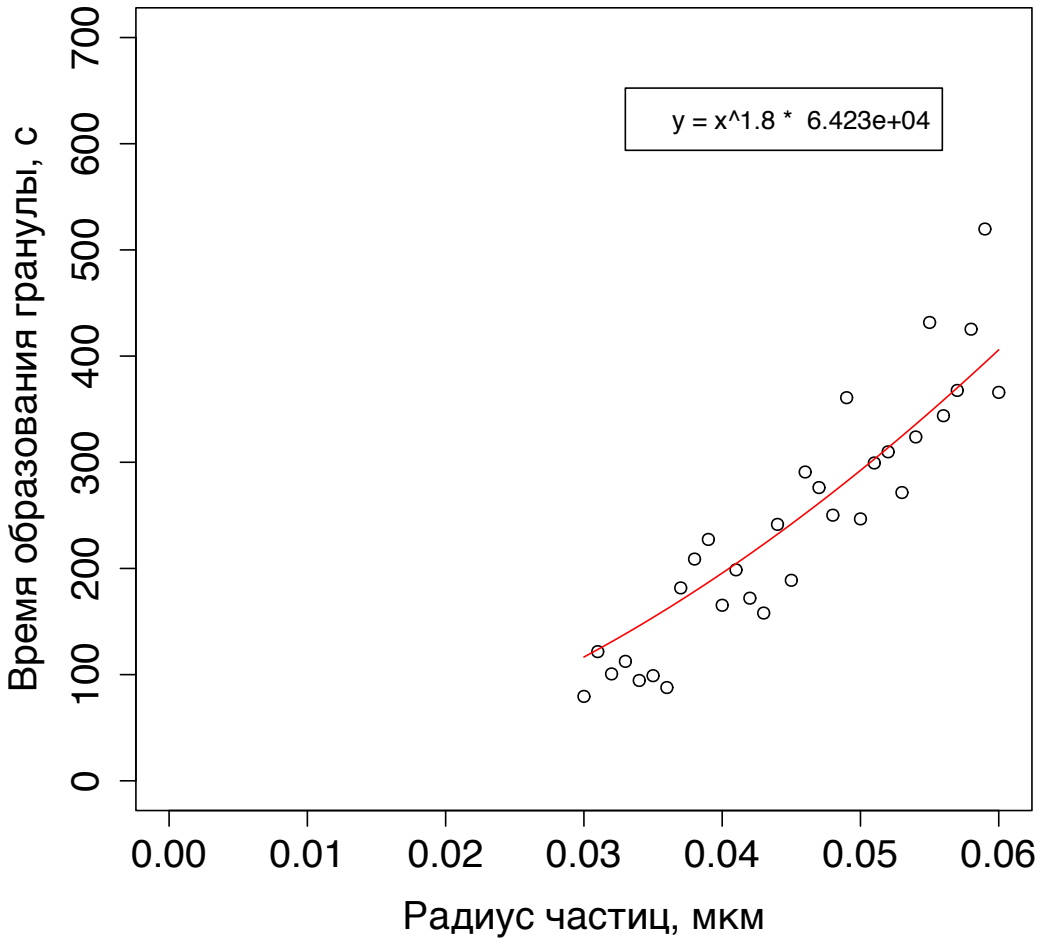


Рисунок 11. Зависимость размера СГ от времени образования.

4.4 Зависимость времени столкновения между СГ от размеров клетки

Рассмотрим зависимость времени столкновения между СГ от размеров клетки. Для этой зависимости нет теоретических формул. Результаты эксперимента говорят об обратно пропорциональной зависимости между величина, если число число гранул брать пропорциональным объему клетки, то есть третьей степени от размера боковой стороны.

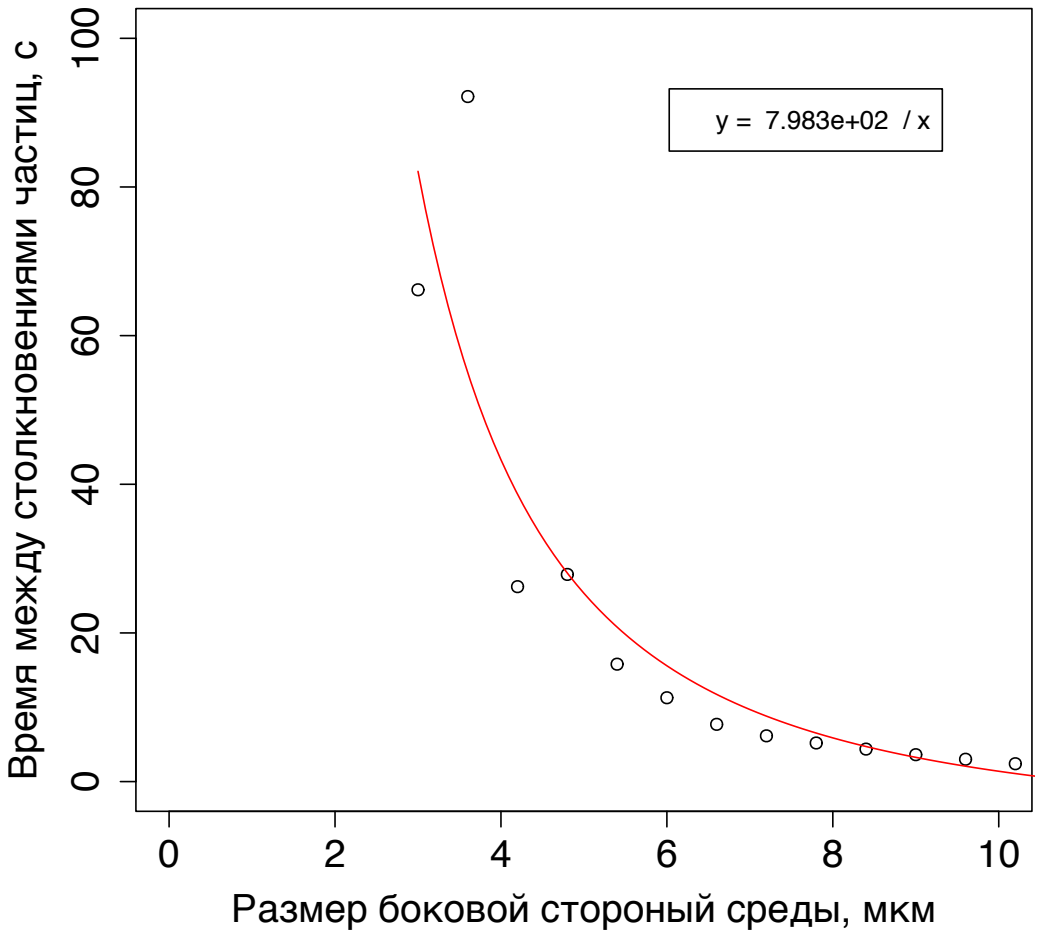


Рисунок 12. Зависимость времени между столкновения СГ от размера боковой стороны среды.

5 Заключение

Была реализована модель движения стрессовых гранул в среде в микротрубочками с броуновским движением частиц и взаимодействием с микротрубочками.

С помощью модели можно точно воспроизвести количество и размеры гранул, наблюдаемые *in vivo*. Были поставлены эксперименты и получены параметры систем, отвечающих воздействию различных веществ на клетку.

Нами также были изучены зависимости между различными характеристиками, таким как время между столкновениями части и время образования гранул определенного размера и параметрами, такими как вязкость среды, концентрация инициаторных комплексов и прочими.

Полученные результаты согласовались с теоретическими выводами и позволили для формул с пропорциональностями получить точные коэффициенты.

Помимо этого, нами были изучены невстречающиеся в научной литературе зависимости между упомянутыми выше характеристиками и параметрами объема клетки и от размера ячейки актиновой сети.

Полученные закономерности позволят в лучшей степени изучить процесс образования стрессовых гранул в клетке и влияющие на него факторы.

6 Выводы

1. Были изучены стрессовые гранулы и роль микротрубочек в их движении.

2. Была построена математическая модель и реализована программа для симуляции и визуализации движения стрессовых гранул.

3. Были разработаны инструменты для проведения экспериментов и изучения зависимости модели от некоторого набора параметров.

4. На основе модели были проверены теоретические оценки: время столкновения между СГ от коэффициента диффузии и концентрации частиц, время столкновения между СГ на микротрубочках от длины и радиуса МТ и зависимость размера СГ от времени ее образования.

5. Были получены закономерности между временем столкновения между СГ от размера клетки и от размера ячейки актиновой сети.

7 Список литературы

[1] P. Anderson and N. Kedersha. Stress granules: the tao of rna triage. *Trends Biochem Sci*, 33(3):141–50, Mar 2008.

[2] C. P. Bacher, M. Reichenzeller, C. Athale, H. Herrmann, and R. Eils. 4-d single particle tracking of synthetic and proteinaceous microspheres reveals preferential movement of nuclear particles along chromatin - poor tracks. *BMC Cell Biol*, 5:45, Nov 2004.

[3] K. G. Chernov, A. Barbet, L. Hamon, L. P. Ovchinnikov, P. A. Curmi, and D. Pastré. Role of microtubules in stress granule assembly: microtubule dynamical instability favors the formation of micrometric stress granules in cells. *J Biol Chem*, 284(52):36569–80, Dec 2009.

[4] A. Einstein. Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen. *Annalen der Physik*, 322(8):549–560, 1905.

[5] S. Heinzer, S. Wörz, C. Kalla, K. Rohr, and M. Weiss. A model for the self-organization of exit sites in the endoplasmic reticulum. *J Cell Sci*, 121(Pt 1):55–64, Jan 2008.

[6] S. Huet, E. Karatekin, V. S. Tran, I. Fanget, S. Cribier, and J.-P. Henry. Analysis of transient behavior in complex trajectories: application to secretory vesicle dynamics. *Biophys J*, 91(9):3542–59, Nov 2006.

[7] P. A. Ivanov and E. S. Nadezhdina. [stress granules: Rnp-containing cytoplasmic bodies springing up under stress. the structure and mechanism of organization]. *Mol Biol (Mosk)*, 40(6):937–44, 2006.

[8] R. P. Jansen. mrna localization: message on the move. *Nat Rev Mol Cell Biol*, 2(4):247–56, Apr 2001.

[9] N. Kedersha and P. Anderson. Stress granules: sites of mrna triage that regulate mrna stability and translatability. *Biochem Soc Trans*, 30(Pt 6):963–9, Nov 2002.

[10] N. Kedersha, S. Chen, N. Gilks, W. Li, I. J. Miller, J. Stahl, and P. Anderson. Evidence that ternary complex (eif2-gtp-trna(i)(met))-deficient preinitiation complexes are core constituents of mammalian stress granules. *Mol Biol Cell*, 13(1):195–210, Jan 2002.

[11] N. Kedersha, M. R. Cho, W. Li, P. W. Yacono, S. Chen, N. Gilks, D. E. Golan, and P. Anderson. Dynamic shuttling of tia-1 accompanies the recruitment of mrna to mammalian stress granules. *J Cell Biol*, 151(6):1257–68, Dec 2000.

[12] N. L. Kedersha, M. Gupta, W. Li, I. Miller, and P. Anderson. Rna-binding proteins tia-1 and tiar link the phosphorylation of eif-2? to the assembly of mammalian stress granules. *The Journal of Cell Biology*, 147(7):1431–1442, 1999.

[13] X. Michalet". Mean square displacement analysis of single-particle trajectories with localization error: Brownian motion in an isotropic medium. *Physical Review E*, 82(4), 2010.

[14] E. S. Nadezhdina, A. J. Lomakin, A. A. Shpilman, E. M. Chudinova, and P. A. Ivanov. Microtubules govern stress granule mobility and dynamics. *Biochim Biophys Acta*, 1803(3):361–71, Mar 2010.

[15] J. B. Perrin. *Atoms*. Van Nostard, 1906.

[16] M. J. Saxton. Lateral diffusion in an archipelago. single-particle diffusion. *Biophysical journal*, 64(6):1766–1780, 06 1993.

[17] M. J. Saxton. Single-particle tracking: effects of corrals. *Biophys J*, 69(2):389–98, Aug 1995.

[18] M. P. Sheetz. Motor and cargo interactions. *Eur J Biochem*, 262(1):19–25, May 1999.

[19] M. G. Thomas, L. J. Martinez Tosar, M. A. Desbats, C. C. Leishman, and G. L. Boccaccio. Mammalian staufen 1 is recruited to stress granules and impairs their assembly. *J Cell Sci*, 122(Pt 4):563–73, Feb 2009.

[20] Weitz and Lin. Dynamic scaling of cluster-mass distributions in kinetic colloid aggregation. *Phys Rev Lett*, 57(16):2037–2040, Oct 1986.

8 Приложение 1. Исходный код

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-

from __future__ import division
import copy
import os
import math
import numbers
import sys
import matplotlib.pyplot as pyplot
import time
import random
import pygame


DF = 1.8
DIMENSIONS = 3

class Config(object):
    MAX_TICKS = 600
    NUMBER_OF_PARTICLES = 400
    INIT_PARTICLE_RADIUS = 0.02
    INIT_PARTICLE_RADIUS_VISUAL = INIT_PARTICLE_RADIUS
    BOARD_SIZE = 6.
    IMMOBILE_THRESHOLD = 0.1
    IMMOBILE_THRESHOLD_MT = IMMOBILE_THRESHOLD
    DIFFUSION_COEFFICIENT = 1. * 10 ** -4
    DIFFUSION_COEFFICIENT_MT = DIFFUSION_COEFFICIENT
    MICROTUBULES_NUMBER = 2
    MICROTUBULE_RADIUS = 0.025
    MICROTUBULE_LENGTH = BOARD_SIZE * 0.5
    FUSE_PARTICLES = True

    def __init__(self, **kws):
        for k, v in kws.items():
            if hasattr(self, k):
                setattr(self, k, v)

```

```
39      def __repr__(self):
40          return '\n'.join("{0}: {1}".format(k, getattr(self, k))
41                           for k in dir(self) if not k.startswith('__'))
42
43
44  class ResultWriter(object):
45      SEPARATOR = '\t'
46      OUTPUT_DIRECTORY = '../results'
47
48      def __init__(self, filename):
49          filepath = os.path.join(self.OUTPUT_DIRECTORY, filename + '.txt')
50          self.writer = open(filepath, 'w')
51
52      def write(self, *args):
53          values = ['{:.08f}'.format(x) if isinstance(x, numbers.Number)
54                  else str(x) for x in args]
55          output = self.SEPARATOR.join(map(str, values))
56          print output
57          print >>self.writer, output
58          self.writer.flush()
59
60      def __del__(self):
61          self.writer.close()
62
63
64  def track_time(func):
65      def f(*args, **kws):
66          t = time.time()
67          result = func(*args, **kws)
68          print 'Run {}, elapsed time: {}'.format(func.func_name, time.time() - t)
69          return result
70      return f
71
72
73  class EventTracker(object):
74      def __init__(self):
75          self.events = []
76          self.total_speed = 0
77          self.total_speed_num = 0
78          self.tracked_time = 0
```

```
79
80      def track(self, tick, event, *args):
81          # print "Event", tick, event, args
82          self.events.append((tick, event, args))
83
84      def track_speed(self, tick, step):
85          global total_speed, total_speed_num
86          speed = math.sqrt(step[0]**2 + step[1]**2 + step[2]**2)
87          self.total_speed += speed
88          self.total_speed_num += 1
89
90      def get_last_event_time(self):
91          return self.events[-1][0]
92
93      def get_average_speed(self):
94          return self.total_speed / self.total_speed_num
95
96      def get_average_event_time(self, event):
97          try:
98              last_event = [x for x in self.events if x[1] == event][-1][0]
99              number_of_events = sum(1 for x in self.events if x[1] == event)
100             return last_event / number_of_events
101         except:
102             return None
103
104     def get_average_p2p_collision_time(self):
105         return self.get_average_event_time('p2p_collistion')
106
107     def get_average_p2m_collision_time(self):
108         return self.get_average_event_time('p2m_collistion')
109
110 global_event_tracker = EventTracker()
111
112
113 class Point(object):
114     def __init__(self, (x, y, z)):
115         self.x, self.y, self.z = x, y, z
116
117     def coords(self):
118         return [self.x, self.y, self.z]
```

```
119
120     def __add__(self, o):
121         return Point((self.x + o.x, self.y + o.y, self.z + o.z))
122
123     def __sub__(self, o):
124         return Point((self.x - o.x, self.y - o.y, self.z - o.z))
125
126     def __mul__(self, k):
127         return Point((self.x * k, self.y * k, self.z * k))
128
129     def __truediv__(self, k):
130         return Point((self.x / k, self.y / k, self.z / k))
131
132     def __repr__(self):
133         return "(" + ','.join('{0:.2}'.format(x) for x in self.coords()) + ')'
134
135
136 class Particle(object):
137     def __init__(self, center, r):
138         self.center = center
139         self.r = r
140         self.mt = None
141
142     def __repr__(self):
143         return "Particle({0}, {1})".format(self.center, self.r)
144
145     def collide(self, o):
146         dx = abs(self.center.x - o.center.x)
147         dy = abs(self.center.y - o.center.y)
148         dz = abs(self.center.z - o.center.z)
149         dist = (dx ** DF + dy ** DF + dz ** DF) ** (1. / DF)
150         return dist < self.r + o.r
151
152     def get_mass(self):
153         return self.r
154
155     def get_visual_radii(self, config):
156         enlargment = (self.r / config.INIT_PARTICLE_RADIUS) ** DF
157         return enlargment * config.INIT_PARTICLE_RADIUS_VISUAL
158
```

```
159
160 class Microtubule(object):
161     def __init__(self, x, z, min_y, max_y, r):
162         self.x = x
163         self.z = z
164         self.min_y = min_y
165         self.max_y = max_y
166         self.r = r
167
168     def __repr__(self):
169         return "Microtubule({0}, {1}, {2} - {3}, {4})".format(
170                 self.x, self.z, self.min_y, self.max_y, self.r)
171
172
173 class GUI(object):
174     def __init__(self, config):
175         self.config = config
176         self.is_running = True
177         self.screen = pygame.display.set_mode((600, 600))
178         self.clock = pygame.time.Clock()
179
180     def process_events(self):
181         for event in pygame.event.get():
182             if (event.type == pygame.QUIT or
183                 event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE):
184                 self.quit()
185
186     def plot_pygame(self, env):
187         COLOR_BACKGROUND = 255, 255, 255
188         COLOR_PARTICLE = 16, 46, 55
189         COLOR_PARTICLE_ON_MT = 243, 40, 40
190         COLOR_MICROTUBULE = 16, 46, 55
191
192         if not self.is_running:
193             return
194         BOARD_SIZE = self.config.BOARD_SIZE
195         fps = self.clock.get_fps()
196         pygame.display.set_caption(
197                 'SG simulation, {:.0f} fps, {}/{} tick'.format(
198                     fps, env.tick, self.config.MAX_TICKS))
```

```
199         self.screen.fill(COLOR_BACKGROUND)
200         width, height = self.screen.get_width(), self.screen.get_height()
201         for particle in env.particles:
202             x = int(particle.center.x * width / BOARD_SIZE)
203             y = height - int(particle.center.y * height / BOARD_SIZE)
204             r = int(particle.get_visual_radii(self.config) * width / BOARD_SIZE)
205             color = COLOR_PARTICLE_ON_MT if particle.mt else COLOR_PARTICLE
206             pygame.draw.circle(self.screen, color, (x, y), r, 1)
207         for mt in env.vertical_mt:
208             x1 = mt.x * width / BOARD_SIZE
209             y1 = height - mt.min_y * height / BOARD_SIZE
210             x2 = mt.x * width / BOARD_SIZE
211             y2 = height - mt.max_y * height / BOARD_SIZE
212             pygame.draw.line(self.screen, COLOR_MICROTUBULE, (x1, y1), (x2, y2))
213         pygame.display.flip()
214         self.clock.tick(100)
215
216     def quit(self):
217         self.is_running = False
218         pygame.quit()
219
220
221 class Environment(object):
222     def __init__(self, config, stop_condition=None):
223         self.event_tracker = EventTracker()
224         self.stop_condition = stop_condition
225         config = config or Config()
226         self.config = config
227         self.vertical_mt = []
228         microtubules_number = self.config.MICROTUBULES_NUMBER
229         x_pos = range(microtubules_number)
230         y_pos = range(microtubules_number)
231         random.shuffle(y_pos)
232         for x, y in zip(x_pos, y_pos):
233             x = self.config.BOARD_SIZE * ((x + 1) / (microtubules_number + 1))
234             # y = self.config.BOARD_SIZE * ((y + 1) / (microtubules_number + 1))
235             y = self.config.BOARD_SIZE * 0.5
236             z = 0
237             length = self.config.MICROTUBULE_LENGTH
238             mt = Microtubule(x, y, z, length, self.config.MICROTUBULE_RADIUS)
```

```
239             self.vertical_mt.append(mt)
240         self.particles = []
241         for _ in range(self.config.NUMBER_OF_PARTICLES):
242             center = Point([random.uniform(0, self.config.BOARD_SIZE)
243                             for _ in range(DIMENSIONS)])
244             particle = Particle(center, self.config.INIT_PARTICLE_RADIUS)
245             self.particles.append(particle)
246         self.is_running = False
247 
248     def move_particles(self):
249         for particle in self.particles:
250             if particle.mt and particle.r < self.config.IMMOBILE_THRESHOLD_MT:
251                 # sigma = get_sigma(self.config.DIFFUSION_COEFFICIENT_MT)
252                 d = self.config.DIFFUSION_COEFFICIENT_MT / particle.r
253                 sigma = math.sqrt(2 * d)
254                 y_step = random.gauss(0, sigma)
255                 step = [0, y_step, 0]
256                 particle.center += Point(step)
257             elif not particle.mt and particle.r < self.config.IMMOBILE_THRESHOLD:
258                 d = self.config.DIFFUSION_COEFFICIENT / particle.r
259                 sigma = math.sqrt(2 * d)
260                 step = [random.gauss(0, sigma) for _ in range(DIMENSIONS)]
261                 self.event_tracker.track_speed(self.tick, step)
262                 particle.center += Point(step)
263             self.bound_particle(particle)
264             self.find_nearest_mt_for_particle(particle)
265 
266     def bound_particle(self, particle):
267         if particle.mt:
268             bounds = [(particle.mt.x, particle.mt.x),
269                       (particle.mt.min_y, particle.mt.max_y),
270                       (particle.mt.z, particle.mt.z)]
271         else:
272             bounds = [(0, self.config.BOARD_SIZE) for _ in range(DIMENSIONS)]
273         coords = particle.center.coords()
274         for i, limits in enumerate(bounds):
275             len = limits[1] - limits[0]
276             if coords[i] > limits[1]:
277                 coords[i] = limits[1] - min(coords[i] - limits[1], len)
278             if coords[i] < limits[0]:
```

```
                coords[i] = limits[0] + min(limits[0] - coords[i], len)
        particle.center.x, particle.center.y, particle.center.z = coords

    def find_nearest_mt_for_particle(self, particle):
        if not particle.mt:
            for mt in self.vertical_mt:
                dx = abs(particle.center.x - mt.x)
                if dx > particle.r + mt.r:
                    continue
                dz = abs(particle.center.z - mt.z)
                if dz > particle.r + mt.r:
                    continue
                dy = max(0, particle.center.y - mt.max_y,
                        mt.min_y - particle.center.y)
                dist = (dx ** DF + dy ** DF + dz ** DF) ** (1. / DF)
                if dist < particle.r + mt.r:
                    particle.mt = mt
                    self.event_tracker.track(self.tick, "p2m_collistion")
                    break

    def fuse_particles(self):
        def weighted_mean(a, wa, b, wb):
            return (a * wa + b * wb) / (wa + wb)
        self.particles.sort(key=lambda particle: particle.center.x)
        for i, particle in enumerate(self.particles):
            if hasattr(particle, 'dead'):
                continue
            x, r = particle.center.x, particle.r
            j = i
            while j > 0 and self.particles[j].center.x > x - 2 * r:
                j -= 1
            while j < len(self.particles) and self.particles[j].center.x < x + 2 * r:
                particle2 = self.particles[j]
                if (i != j and particle.collide(particle2) and
                        not hasattr(particle2, 'dead') and
                        not (particle.mt and particle2.mt and particle.mt != particle2.mt)):
                    self.event_tracker.track(self.tick, "p2p_collistion")
                    particle.center = weighted_mean(particle.center,
                                                    particle.get_mass(),
                                                    particle2.center,
```

```
                                                particle2.get_mass())
                    particle.r = (particle.r ** DF + particle2.r ** DF) ** (1. / DF)
                    particle2.r = 0
                    self.particles[j].dead = True
                j += 1
        self.particles[:] = [particle for particle in self.particles
                             if not hasattr(particle, 'dead')]

    def run_simulation(self, gui_enabled=True, verbose=True):
        t = time.time()
        if gui_enabled:
            gui = GUI(self.config)
        for self.tick in range(self.config.MAX_TICKS):
            if gui_enabled and not gui.is_running:
                break
            if self.stop_condition and self.stop_condition(self):
                break
            self.event_tracker.track(self.tick, "tick")
            self.move_particles()
            if self.config.FUSE_PARTICLES:
                self.fuse_particles()
            if verbose and self.tick % 10 == 0:
                print self.tick, len(self.particles)
            if gui_enabled:
                gui.process_events()
                gui.plot_pygame(self)
        if verbose:
            print 'top 5 radiuses:', sorted(["{:.4f}".format(particle.r)
                for particle in self.particles], reverse=True)[:5]
            print 'particles left:', len(self.particles)
            print 'total time:', time.time() - t
            print 'average speed:', self.event_tracker.get_average_speed()
            print 'average particle/particle collision time:',\
                    self.event_tracker.get_average_event_time('p2p_collistion')
            print 'average particle/microtubule collision time:',\
                    self.event_tracker.get_average_event_time('p2m_collistion')

    def plot(self):
        BOARD_SIZE = self.config.BOARD_SIZE
        pyplot.clf()
```

```
359         for particle in self.particles:
360             circle = pyplot.Circle(particle.center.coords(),
361                     particle.get_visual_radii(self.config), fill=False)
362             pyplot.gcf().gca().add_artist(circle)
363         for mt in self.vertical_mt:
364             pyplot.plot([mt.x, mt.x], [mt.min_y, mt.max_y], 'b')
365         pyplot.axis([0, BOARD_SIZE, 0, BOARD_SIZE])
366         pyplot.axes().set_aspect('equal')
367         # pyplot.ion()
368         # pyplot.show()
369         timestamp = int(time.time())
370         image_filename = "../output/{0}.eps".format(timestamp)
371         pyplot.savefig(image_filename)
372         description_filename = "../output/{0}.txt".format(timestamp)
373         with open(description_filename, 'w') as writer:
374             print >>writer, self.config
375
376 @track_time
377 def run_visualizer():
378     env = Environment(Config(MAX_TICKS=600))
379     env.run_simulation()
380     env.plot()
381
382 def get_average(data):
383     size = len(data[0])
384     result = []
385     for i in range(size):
386         values = sorted(x[i] for x in data if x[i] is not None)
387         sz = max(3, len(values) / 2)
388         while len(values) >= sz + 2:
389             del values[0]
390             del values[-1]
391         try:
392             value = sum(values) / len(values)
393         except:
394             value = None
395         result.append(value)
396     return result
397
398
```

```
399 class Experiment(object):
400     arguments = []
401     runs = 1
402     writer = None
403 
404     def __init__(self):
405         self.setup()
406         self.run()
407 
408     def setup(self):
409         pass
410 
411     def get_name(self):
412         return self.__class__.__name__
413 
414     @track_time
415     def run(self):
416         self.arguments = sorted(self.arguments)
417         print '{} experiment'.format(self.get_name())
418         print 'arguments:', self.arguments
419         for arg in self.arguments:
420             data = []
421             for _ in range(self.runs):
422                 result = self.calculate(arg)
423                 data.append(result)
424             avg_result = get_average(data)
425             self.writer.write(*avg_result)
426 
427     def calculate(self, arg):
428         pass
429 
430 
431 class P2pCollisionTimeOnDiffusionExperiment(Experiment):
432     runs = 30
433     arguments = {0.001 * x for x in range(1, 101, 5)}
434 
435     def setup(self):
436         self.config = Config(MAX_TICKS=300, MICROTUBULES_NUMBER=0)
437         self.writer = ResultWriter(self.get_name())
438         self.writer.write('diffusion', 'speed', 'collision_time')
```

```
439
440     def calculate(self, arg):
441         config = copy.copy(self.config)
442         config.DIFFUSION_COEFFICIENT *= arg
443         config.DIFFUSION_COEFFICIENT_MT *= arg
444         env = Environment(config)
445         env.run_simulation(False, False)
446         result = (config.DIFFUSION_COEFFICIENT,
447                   env.event_tracker.get_average_speed(),
448                   env.event_tracker.get_average_event_time('p2p_collistion'))
449         return result
450
451
452 class P2pCollisionTimeOnConcetrationExperiment(Experiment):
453     runs = 9
454     arguments = {x for x in range(200, 601, 10)}
455
456     def setup(self):
457         self.config = Config(MAX_TICKS=300, MICROTUBULES_NUMBER=0)
458         self.writer = ResultWriter(self.get_name())
459         self.writer.write('number_of_particles', 'speed', 'collision_time')
460
461     def calculate(self, arg):
462         config = copy.copy(self.config)
463         config.NUMBER_OF_PARTICLES = arg
464         env = Environment(config)
465         env.run_simulation(False, False)
466         result = (config.NUMBER_OF_PARTICLES,
467                   env.event_tracker.get_average_speed(),
468                   env.event_tracker.get_average_event_time('p2p_collistion'))
469         return result
470
471
472 class P2mCollisionTimeOnLengthExperiment(Experiment):
473     runs = 5
474     arguments = {0.1 * x for x in range(1, 21)} | {0.3 * x for x in range(1, 21)}
475
476     def setup(self):
477         self.config = Config(MAX_TICKS=300, FUSE_PARTICLES=False)
478         self.writer = ResultWriter(self.get_name())
```

```
479         self.writer.write('microtubule_length', 'collision_time')
480
481     def calculate(self, arg):
482         config = copy.copy(self.config)
483         config.MICROTUBULE_LENGTH = arg
484         env = Environment(config)
485         env.run_simulation(False, False)
486         result = (config.MICROTUBULE_LENGTH,
487                   env.event_tracker.get_average_event_time('p2m_collistion'))
488         return result
489
490 class P2mCollisionTimeOnRadiusExperiment(Experiment):
491     runs = 50
492     arguments = {0.01 * x for x in range(1, 21)} |\
493             {0.002 * x for x in range(2, 11)}
494
495     def setup(self):
496         self.config = Config(
497                 MAX_TICKS=300,
498                 NUMBER_OF_PARTICLES=100,
499                 FUSE_PARTICLES=False,
500                 IMMOBILE_THRESHOLD=1,
501                 IMMOBILE_THRESHOLD_MT=1)
502         self.writer = ResultWriter(self.get_name())
503         self.writer.write('radius', 'collision_time')
504
505     def calculate(self, arg):
506         config = copy.copy(self.config)
507         config.INIT_PARTICLE_RADIUS = arg
508         config.MICROTUBULE_RADIUS = arg
509         env = Environment(config)
510         env.run_simulation(False, False)
511         result = (config.INIT_PARTICLE_RADIUS,
512                   env.event_tracker.get_average_event_time('p2m_collistion'))
513         return result
514
515 class CreatingLargeGranulesExperiment(Experiment):
516     runs = 10
517     arguments = {0.03 + 0.001 * x for x in range(31)}
518
```

```
519     def setup(self):
520         self.config = Config(
521                 MAX_TICKS=1000,
522                 IMMOBILE_THRESHOLD=1,
523                 IMMOBILE_THRESHOLD_MT=1)
524         self.writer = ResultWriter(self.get_name())
525         self.writer.write('desired_radius', 'time')
526 
527     def calculate(self, arg):
528         def stop_condition(env):
529             R = max(particle.r for particle in env.particles)
530             return R > arg
531         config = copy.copy(self.config)
532         env = Environment(config, stop_condition=stop_condition)
533         env.run_simulation(False, False)
534         result = (arg,
535                   env.event_tracker.get_last_event_time())
536         return result
537 
538 class P2pCollisionTimeOnVolumeExperiment(Experiment):
539     runs = 10
540     arguments = {0.1 * x for x in range(5, 21)}
541 
542     def setup(self):
543         self.config = Config(MAX_TICKS=300, MICROTUBULES_NUMBER=0)
544         self.writer = ResultWriter(self.get_name())
545         self.writer.write('board_size', 'number_of_particles', 'collision_time')
546 
547     def calculate(self, arg):
548         config = copy.copy(self.config)
549         config.NUMBER_OF_PARTICLES = int(config.NUMBER_OF_PARTICLES * arg ** 3)
550         config.BOARD_SIZE *= arg
551         env = Environment(config)
552         env.run_simulation(False, False)
553         result = (config.BOARD_SIZE,
554                   config.NUMBER_OF_PARTICLES,
555                   env.event_tracker.get_average_event_time('p2p_collistion'))
556         return result
557 
558 
```

```
559 class P2pCollisionTimeOnImmobilityExperiment(Experiment):
560     runs = 3
561     arguments = {0.02 * x for x in range(5, 21)}
562 
563     def setup(self):
564         self.config = Config(MAX_TICKS=300, MICROTUBULES_NUMBER=0)
565         self.writer = ResultWriter(self.get_name())
566         self.writer.write('immobile_threshold', 'collision_time')
567 
568     def calculate(self, arg):
569         config = copy.copy(self.config)
570         config.IMMOBILE_THRESHOLD = arg
571         env = Environment(config)
572         env.run_simulation(False, False)
573         result = (config.IMMOBILE_THRESHOLD,
574                   env.event_tracker.get_average_event_time('p2p_collistion'))
575         return result
576 
577 
578 if __name__ == "__main__":
579     random.seed(1)
580     run_visualizer()
581     # P2pCollisionTimeOnDiffusionExperiment()
582     # P2pCollisionTimeOnConcetrationExperiment()
583     # P2mCollisionTimeOnLengthExperiment()
584     # P2mCollisionTimeOnRadiusExperiment()
585     # CreatingLargeGranulesExperiment()
586     # P2pCollisionTimeOnVolumeExperiment()
587     # P2pCollisionTimeOnImmobilityExperiment()
```