Strategies for E-cadherin Recycling: A Computational Model

A PROJECT REPORT

Submitted as part of BBD451 BTech Major Project (BB1)

Submitted by:
Vaibhav Agarwal 2020BB10061

Guided by:
Professor Amit Das

DEPARTMENT OF BIOCHEMICAL ENGINEERING AND BIOTECHNOLOGY
INDIAN INSTITUTE OF TECHNOLOGY DELHI

September 2023

DECLARATION

I certify that

a) the work contained in this report is original and has been done by me under theguidance of my supervisor(s).

b) I have followed the guidelines provided by the Department in preparing the report.

c) I have conformed to the norms and guidelines given in the Honor Code of Conduct of the Institute.

d) whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text ofthe report and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Signed by:
Name of the Student

CERTIFICATE

It is certified that the work contained in this report titled "**Strategies for E-cadherin Recycling: A Computational Model**" is the original work done by **Vaibhav Agarwal** and has been carried out under my supervision.

Signed by:
Name of the Faculty Mentor

Date:

ABSTRACT

Adhesion receptors, such as E-Cadherin, are crucial proteins found on the cell surface that facilitate cell adhesion by mediating physical attractions between the cell and its surrounding environment, specifically the extracellular matrix (ECM). E-Cadherin plays a pivotal role in various biological functions, including cell attachment, mobility, and signal communication. A notable feature of E-Cadherin is its ability to form mechanical links between neighbouring cells in epithelial tissues, which underpin many multicellular processes. These molecules tend to cluster, providing stability to the mechanical links between cells. This paper delves into the mechanisms controlling E-Cadherin's recycling and eventual endocytosis. Using the Drosophila follicular epithelium as a model, this study uses computational modelling to delve into the physical processes behind E-Cadherin recycling and endocytosis. We utilize Langevin dynamics simulations to observe E-Cadherin molecules on a plasma membrane. Furthermore, we simulate the interactions among the proteins using conventional pair potentials and examine how E-Cadherin endocytosis influences the clustering mechanism.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Abbreviation	Description
ECM	Extracellular Matrix
PM	Plasma Membrane
ZA	Zonula Adherens
Rab11, RabX1, Rab5	Member of Rab family of GTPases
DE-cad	Drosophilla E-cadherin

Chapter 1
INTRODUCTION

Cells utilize adhesion molecules to interact with their surroundings [1-3]. These structural proteins emerge from non-covalent bonds between adhesion molecules on one cell surface and receptors on adjacent cells or mediator molecules within the Extracellular Matrix (ECM). Cell adhesions are pivotal for maintaining tissue cohesion and facilitating cell communication. Through these adhesions, cells can transmit forces, enabling interaction and sensing.

E-cadherin, a transmembrane protein, is paramount in orchestrating cell attachment in epithelial cells. The accumulation of E-cadherin at intracellular junctions is vital for preserving tissue integrity and modulating epithelial tissue activity. Cell-cell adhesion is fortified when E-cadherin receptors on neighbouring cells interact and cluster at the cellular membrane. The integrity of tissues and the regulation of epithelial tissue activities hinge on the aggregation of E-cadherin at these intracellular boundaries and the maturation of these cell-cell adhesions [6].

E-cadherin facilitates cell-cell adhesion by establishing homophilic contacts with E-cadherin molecules on adjacent cells. The intracellular domain of E-cadherin interacts with several cytoplasmic proteins, including beta-catenin, alpha-catenin, and p120 catenin. These interactions are essential for linking the E-cadherin adhesion complex to the cell's actin cytoskeleton. The formation of E-cadherin clusters at cellular interfaces is a pivotal step in the evolution of cell-cell adhesion. The lateral association of E-cadherin molecules, facilitated by calcium-dependent interactions and connections between the cytoplasmic domains of E-cadherin and catenin, leads to the clustering of E-cadherin [1].

Associations between E-cadherin and F-actin are mediated by beta-catenin, alpha-catenin, and other F-actin-binding proteins like Vinculin and EPLIN. As epithelia develop, significant restructuring of cell junctions occurs. This necessitates the regulation of cell-cell adhesion, such as by modulating cadherin levels or renewing them through endocytic recycling. A recycling mechanism redistributes E-cadherin from the lateral PM to the apicolateral PM, leading to its accumulation at the ZA. Recycling involves molecular interactions between

Rab11, the exocyst complex, and beta-catenin. The study identifies RabX1 as a critical new component for DE-cadherin recycling, placing its function between the early and the recycling endosome. In RabX1 mutants, endocytosed DE-cadherin protein is not properly recycled but accumulates together with Rab5 and Rab11 in a large compartment. This targeted recycling is essential for the maintenance of the ZA and cell shape.

Chapter 2

MOTIVATION

The cellular landscape is a complex web of interactions, processes, and dynamics. Within this intricate framework, the recycling of molecules, particularly e-cadherin, stands out as a pivotal process. E-cadherin, a cornerstone of cell-cell adhesion, plays a significant role in maintaining tissue integrity and orchestrating various cellular activities. Its recycling, specifically through endocytosis, is a managed interaction that influences its organization on the cell surface of numerous metazoan organisms.

Given the importance of e-cadherin and its recycling, understanding the nuances of this process is crucial. Dysregulation of E-cadherin endocytosis has been linked to various diseases, including cancer and developmental disorders. For instance, rapid e-cad acquisition by endocytosis and a significant reduction in E-cadherin concentrations are associated with the collapse of the columnar epithelial structure during epithelial-mesenchymal transition (EMT), a critical phase in cancer development [1]. Variations in the appearance of adhesion molecules have also been identified as a cause of the cell-sorting process in tissue culture.

However, despite its significance, several aspects of e-cadherin recycling remain enigmatic. The nature of specific interactions leading to e-cadherin clustering, the dynamics of its endocytosis, and its movement on the cell surface are areas that require deeper exploration. Traditional experimental methods, while invaluable, may not capture the full spectrum of molecular dynamics and interactions at play [1].

This is where computational modelling steps in. Computational modelling offers a powerful tool to simulate, analyse, and predict the behaviour of molecules in various scenarios. By leveraging computational tools, we can delve into the microscopic world of e-cadherin recycling, exploring the nature of interactions between E-cadherins and understanding the coupling between the dynamics of endocytosis and the movement of E-cadherin on the cell surface. Such models can provide insights that are challenging to obtain through experimental means alone.

Furthermore, the research paper highlighted the role of Rab11 in controlling the transport of newly synthesized E-cadherin from the Golgi to the plasma membrane, emphasizing the significance of recycling pathways in maintaining E-cadherin's presence at the ZA [13]. By integrating these insights into computational models, we can gain a holistic understanding of the recycling process, its regulatory mechanisms, and its implications in health and disease.

Chapter 3
LITERATURE SURVEY

Endocytosis, a key cellular process, plays a pivotal role in modulating the amount of E-cadherin on cellular interfaces. This recycling mechanism not only regulates the distribution of E-cadherin but also selectively targets and reduces the formation of large E-cadherin clusters. The underlying hypothesis suggests that larger clusters might be more susceptible to endocytosis due to their propensity to facilitate the assembly of endocytic machinery. Such macroscopic clusters, if unchecked, could potentially disrupt the actomyosin system, leading to a cessation of tissue movements. By regulating the size and distribution of these clusters through recycling, endocytosis ensures the smooth functioning of cellular processes. [13]

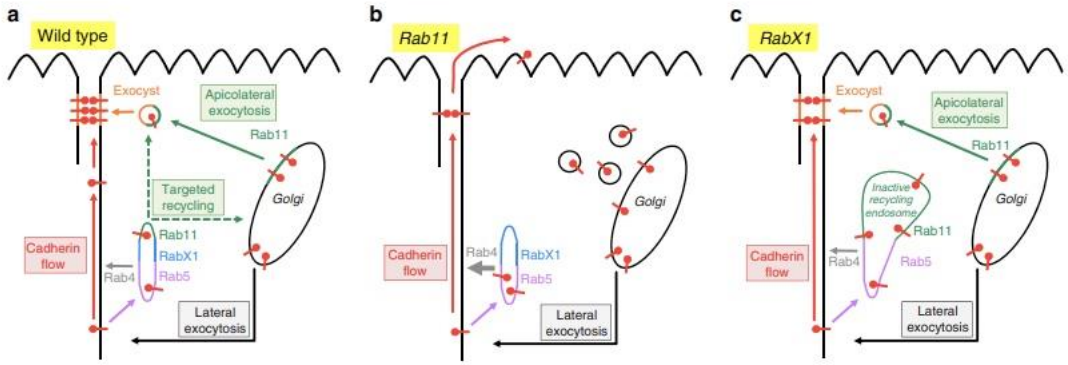

Figure 3.1 : Model displaying the mechanisms to control DE-cad localization

While E-cadherin can aggregate independently of actin, experimental data underscores the importance of E-cadherin's associations with actin for maintaining cluster stability in live epithelia. Recent computational models have proposed that E-cadherin might spontaneously aggregate under the influence of lateral forces [7]. Actin-based control plays a crucial role in preventing the disintegration of cadherin complexes. Despite evidence suggesting that E-cadherin mediates force transmission between the cytoskeleton and the cellular environment, the capacity of E-cadherin clustering to regulate this force transfer remains an area of active research [1]. The recycling of E-cadherin, especially through endocytosis, is thought to play a role in stabilizing these clusters and maintaining cellular adhesion.[7]

E-cadherin's role in cellular motility is multifaceted. While it restricts cell movement on matrices, its influence on cell movement through cell-rich tissues remains ambiguous. In-depth studies using in vivo mechanical stress sensors and other advanced techniques have revealed that E-cadherin-mediated adhesion between border cells and nurse cells stabilizes forward-directed protrusion, ensuring consistent mobility. This adhesion mechanism also facilitates cellular communication, with leading cells providing directional cues to follower cells. [2]

RAB5A, an essential endocytic protein, has been observed to stimulate the formation of distinct actin-based protrusions. These protrusions generate traction forces that are transmitted over extended distances through junctional contacts. This intricate interplay between mechanical coupling and polarity establishes a feedback loop, enabling cells to receive directional cues from neighboring cells. Such interactions enhance the dynamism of

multicellular organisms, optimizing junctional E-cadherin dynamics to accommodate changes in cellular proximity, volume, density, and stress. The recycling of E-cadherin, especially its redistribution from lateral to apicolateral regions, plays a crucial role in this dynamic process. [3]

Metastasis is a leading cause of cancer-related deaths. A notable observation is the inverse relationship between in vitro movement and E-cadherin concentrations. Despite the majority of breast cancers being invasive ductal carcinomas that express E-cadherin, its depletion has been linked to increased invasion. However, this also results in reduced tumor growth, survival, and metastatic spread. Strategies targeting E-cadherin-mediated survival pathways could offer potential therapeutic avenues for metastatic breast cancer.[5]

Chapter 4

MATERIAL AND METHODS

When two cells come into contact, the E-cadherin molecules on their surfaces become crucial. These molecules from each cell engage with each other, leading to the cells sticking together. The bond between the cells strengthens when E-cad molecules group together on one cell and connect with similar clusters on an adjacent cell.

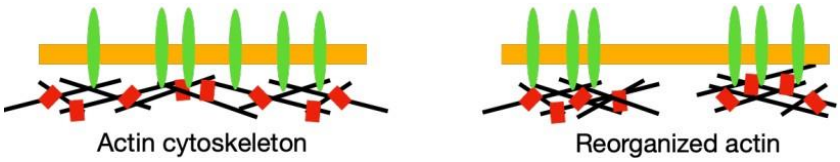

Figure 4.1 : E-cadherin clustering driven by F-actin

According to recent reports (Truong-Quang et al), F-actin plays a pivotal role in giving rise to the lateral movements of E-cadherin molecules at the cell surface. The E-cadherin molecules which come together then engage in cis-interactions which lead to cluster formation. These clusters are believed to regulate the maturation of cell-cell junctions where they engage in trans-interactions with clusters of E-cadherin from another nearby cell surface (Charras et al,

current ref 10). However, the principles that dictate these molecular groupings and interactions are yet to be defined. [1]

To delve deeper into these interactions, we base our analysis on certain presumptions. We use the Langevin equation to guide the movement of these protein molecules:

$$d^2X / dt^2 = - \Gamma \, dX/dt + F_{int} + F_{random} \qquad \text{Eq.(4.1)}$$

In this equation, the left side signifies a particle's acceleration in the x-direction, which also represents the inertial force. Γ stands for the friction coefficient, dX/dt indicates the velocity of the particle, F denotes the force of interaction between particles, and F random represents the force arising from the unpredictable motion of the surrounding fluid or environment [8].

We employ Computational Langevin dynamics simulations to capture the unpredictable behaviour of intricate systems like biological entities, materials, and molecular structures. This method can account for the impact of temperature variations and unforeseen forces, which are pivotal for a precise prediction of the actions of intricate systems.

The protein encounters an active random force. This force emulates the influence of F-actin found in the cell cortex, which momentarily links with the proteins, causing their movement [7]. Further resolution of this equation yields:

$$\Gamma \, dX/dt = F_{random} = v \, n \qquad \text{Eq.(4.2)}$$

v is the characteristic velocity of the protein molecules arising from F-actin, and n is a unit vector that points in a random direction at any instant. dX is the displacement with time t. [8]

To correctly capture the bulk movement of particles, we use the Periodic Boundary Condition. In a periodic simulation setup, particle positions are replicated periodically throughout space, forming an endless grid of identical cells. The characteristics of PBC allow us to simulate a segment of the cellular interface that's significantly larger than individual protein sizes. For implementing PBC, it's essential to adhere to the minimum image convention (MIC) when determining molecular interactions. According to MIC, when molecules move past the cell's boundary, they interact with the nearest replicated image of themselves.

Instead of using the positions of all particles across all cells – a method that would lead to numerous unnecessary calculations – the conventional method focuses on interactions. The minimum image convention streamlines this by only considering the closest periodic image relative to each particle. By eliminating the farthest periodic image from the target cell, each particle's position is effectively confined within the simulation cell. This approach not only cuts down on computational demands but also ensures precise distance measurements between particles.

To integrate this convention into the programming, we utilize the np.round function from Python's NumPy library. This function allows us to round an array or a scalar value to a designated number of decimal points. Thus,

$$x_{ij} = x_{ij} - \text{np.round}\,(x_{ij}/\,L)\,L$$

Beyond the MIC, we also define an interaction range for the protein molecules. We postulate that proteins only interact when they come within a distance of r_{cut} from each other. This r_{cut} denotes the threshold distance between the centers of two protein molecules, beyond which they can establish a harmonic bond in our simulations. We designate r_{cut} as 2x%, characterizing the protein-protein interaction as a short-range attractive potential. In essence, proteins only engage when in close proximity; for distances exceeding r_{cut}, the interaction energy becomes null.

We've now established a model for the active motion of E-cadherin molecules on a membrane segment. In this model, when two proteins come near each other, they have the potential to create a bond.

We have already modelled the E-cad recycling via a Black-Box method, in which we directly added the recycled particles back without differentiating between their origins in the cell mechanism. Now in accordance to our future plan we would design a computational model that would incorporate all the different recycling mechanisms discussed in the wild type, RabX1 mutant and Rab11 mutant.

For convenience and in accordance with our literature survey we can safely assume that the Rab4, RabX1 and Rab11 protein recycling mechanism to be different.

In endocytosis some of the protein molecules that are removed from the cell surface are going to be recycled and added back to the cell surface after some time delay. This time delay is to account for the recycling process that takes place. We are also currently assuming that all the 3 types of recycling processes that are present are the same and are in a sort of a Black Box. Let's assume that the number of molecules that were to be removed are nd via endocytosis and the number of molecules that were to be added are na calculated by using the random.gauss function, which is a function in the built-in random module of Python that generates Gaussian (normal) distributed random numbers with a specified mean and standard deviation. It has two parameters: μ (mean of the distribution) and σ (standard deviation of the distribution). This way we model both endocytosis and exocytosis as Gaussian random processes which is a reasonable simplification to develop fundamental understanding. Also to incorporate the recycling process, a certain fraction of nd are going to be recycled back, let that fraction be represented by "recycleRate" and the recycled particles that are going to be excytosised after the time delay would be equal to this fraction of nd. This time delay is represented by "recycleFreq" in the code.

This finally gives us :

endocytosis rate of $k_{endo} = < n_d > /(N_t \Delta t)$

and excocytosis rate of $k_{exo} = < n_a > /(N_t \Delta t)$.

In the simulation, we supply the following sets of parameters to the random.gauss function:

$\mu_{endo} = < n_d >$ and $\sigma_{endo} = \sqrt{-< n_{d'} > - < n_d >}$, and

$\mu_{exo} = < n_a >$ and $\sigma_{endo} = \sqrt{-< n_{a'} > - < n_a >}$,

We keep $\mu_{endo} \approx \mu_{exo}$ to ensure the number of molecules present in the system at any given time remains nearly unchanged. We use same values for σ_{endo} and σ_{exo}. The advantage of using a Gaussian distribution for describing such random processes is that it is one of the fundamental probability distributions commonly used in scientific and engineering applications, and are often a better model for natural processes than uniformly distributed

random numbers.

Chapter 5

RESULTS AND DISCUSSION

Since, we have already seen various protein interactions and their movements on the cell surface in Ms. Bagmare's work, and also with addition of recycled molecules back we are not seeing difference on the cell surface between time traces we take as it is qualitatively similar to Wild type. We would be focusing on the various cell types and effect of the effect of addition of recycling on the model. We would be considering three cases, one in which endocytosis rate is higher than exocytosis, another in which latter is higher and the last in which both are equal. All other variables are fixed. We would expect that there would be no change in the trend of total number of molecules for the three cases i.e in case 1 total number would decrease in both cases, in case 2 total number of molecules would increase and in the third it would remain around the same.

As we see from the above illustrations that the we are getting results in order with our hypothesis and the model works to great accuracy in representing cluster formation even when adding recycled molecules back with a time delay.

Chapter 6

SUMMARY AND CONCLUSIONS

We see from the observed graphs and timelines of E-cad molecules present on the cell surface

REFERENCES

1) Binh-An Truong Quang, Madhav Mani, Olga Markova,Thomas Lecuit, and Pierre-Francois Lenne – "Principles of E-Cadherin Supramolecular Organization In Vivo", Current Biology 23, pg. 2197-2207, November 18, 2013

2) Danfeng Cai, Shann-Ching Chen, Mohit Prasad, Li He, Xiaobo Wang, Valerie Choesmel-Cadamuro, Jessica K. Sawyer, Gaudenz Danuser, "Mechanical Feedback through E-Cadherin Promotes Direction Sensing during Collective Cell Migration", Cell, Volume 157, issue 5, May 2014, pg.1146-1159

3) Chiara Malinverno. et.al, "Endocytic reawakening of motility in jammed epithelia", Nature materials 16, 587, January 2017

4) Robert J. Tetley and Yanlan Mao, "The same but different: cell intercalation as a driver of tissue deformation and fluidity", Philosophical transactions of the royal society, Biological Sciences 373, September 2018

5) Veena Padmanaban, Ilona Krol, Yasir Suhail, Barbara M. Szczerba, Nicola Aceto, Joel S. Bader & Andrew J. Ewald, "E-cadherin is required for metastasis in multiple models of breast cancer", Nature 573, pages: 439–444 , September 2019

6) Diego A. Vargas, Hans Van Oosterwyck, "Cell Adhesion: Basic Principles and Computational Modeling", Encyclopedia of biomedical engineering, pg: 45-58, 2019

7) Yang Chen,Julia Brasch,Oliver J. Harrison and Tamara C. Bidone, "Computational model of E- cadherin clustering under force", Biophysical Journal 120, pg: 4944–4954, November 2021

8) Understnading molecular simulations --- from algorithms to applications, by Daan Frenkel and Berend Smit.

9) Timothy E Vanderleest, Celia M Smits, Yi Xie, Cayla E Jewett, J Todd Blankenship , Dinah Loerke , University of Denver, US, "Vertex sliding drives intercalation by radial coupling of adhesion and actomyosin networks during Drosophila germband extension", Computational and systems biology, developmental biology, July 2018

10) Guillaume Charras, Alpha S Yap, "Tensile Forces and Mechano-transduction at Cell-Cell Junctions", Current Biology, 28(8):R445-R457, April 2018

11) Amit Das, Abrar Bhat, Rastko Sknepnek, Darius Köster, Satyajit Mayor, Madan Rao, “Stratification relieves constraints from steric hindrance in the generation of compact actomyosin asters at the membrane cortex”, Science Advances. 6, eaay6093 (2020).
12) Collins C, Denisin AK, Pruitt BL, Nelson WJ. Changes in E-cadherin rigidity sensing regulate cell adhesion. Proceedings of the National Academy of Sciences. 2017 Jul 18;114(29):E5835- 44.
13) Woichansky I, Beretta CA, Berns N, Riechmann V. Three mechanisms control E-cadherin localization to the zonula adherens. Nat Commun. 2016 Mar 10;7:10834. doi: 10.1038/ncomms10834. PMID: 26960923; PMCID: PMC4792928.
14) Radhikha Bagmare Endterm Report for BBD451, Batch of 2019 Biotechnology and Biochemical Engineering https://drive.google.com/file/d/1e7-eAMUzLDibqPVemeewOvpuhIl3IuFm/view?usp=sharing
15) B. Ladoux & R.-M. Mège. *Nat. Rev. Mol. Cell Biol.* **18**, 743 (2017).
16) A. I. Bachir, A. R. Horwitz, W. J. Nelson & J. M. Bianchini. *Cold Spring Harb. Perspect. Biol.* **9**, a023234 (2017).
17) N. C. Heer & A. C. Martin. *Development* **144**, 4249 (2017).

APPENDICES

A) Python Program

Below is the python program for E-cadherin endocytosis and Recycling mechanism:

```
import numpy as np
import random
import matplotlib.pyplot as plt
import cv2
from datetime import datetime


def Rab4Recycle(removed_particles_entering_rab4, rab4_efficiency):
    return int(np.round(removed_particles_entering_rab4*rab4_efficiency))
def TubularRecycle(removed_particles_entering_tubular, tubular_efficiency):
    return int(np.round(removed_particles_entering_tubular*tubular_efficiency))
def Rab4Efficiency(n_part):
    if n_part < 40:
        return 0.5
    elif n_part > 100:
        return 0.2
    else:
        return -0.005 * n_part + 0.7
def wildType(n_iter, mechanism_type):
    currentDate = datetime.now().date()
    currentTime = datetime.now().time()
    currentTime = str(currentTime)[:8]
    part_arr = []
    numbers = []
    print("WILD TYPE CALLED")
    del_t = 0.01
    L=20.0
    v = 1.0
    k = 1.0
    arr = []
    n_part = 100
    vecx = [[] for i in range(n_part)]
    vecy = [[] for i in range(n_part)]
    for i in range(n_part):
        x = random.random()*L
```

```
        y = random.random()*L
        vecx[i].append(x)
        vecy[i].append(y)
        arr.append((x,y))
    endo_rate = 10
    exo_rate = 10
    sigma_rate = 5
    freq = 100
    recycleFreq = 30
    recycled_particles = 0
    rab4_efficiency = 0.2
    lamRecycleFreqRab4 = 20
    lamRecycleFreqTubular = 40
    recycleFreqTubular = 40
    recycleFreqRab4 = 20
    tubular_efficiency = 0.4
    recycleRate = 0.3
    l0=1.1 # touching distance of molecules
    img_names = []
    for t in range(n_iter):
        if t%freq==0:
            fig, ax = plt.subplots()
            to_printx =[]
            to_printy =[]
            for i in range(n_part):
                to_printx.append(vecx[i][-1])
                to_printy.append(vecy[i][-1])
            ax.plot(to_printx, to_printy,'o')
            fig.savefig(f"images_my/{mechanism_type}/graph_{t}.png")
            plt.close(fig)
            img_names.append(f"images_my/{mechanism_type}/graph_{t}.png")
            nd = int(np.round(random.gauss(endo_rate,sigma_rate)))
            while(nd>=n_part):
                print("DANGER number of removed particles are greater than particles present, bleak possibility")
                nd = int(np.round(random.gauss(endo_rate,sigma_rate)))
            recycleFreqTubular = np.random.poisson(lam=lamRecycleFreqTubular)
            recycleFreqRab4 = np.random.poisson(lam=lamRecycleFreqRab4)
            recycled_particles_rab4 = Rab4Recycle(removed_particles_entering_rab4= nd*0.5,
rab4_efficiency=Rab4Efficiency(n_part))
```

```
            recycled_particles_tubular = TubularRecycle(removed_particles_entering_tubular=nd*0.5,
tubular_efficiency=tubular_efficiency)
            removed_ind = set()
            while(len(removed_ind)<nd):
                x = random.randint(0, n_part-1)
                removed_ind.add(x)
            new_indices = []
            for i in range(n_part):
                if i not in removed_ind:
                    new_indices.append(i)
            tvecx=[]
            tvecy = []
            tarr=[]
            for ind in new_indices:
                tvecx.append(vecx[ind])
                tvecy.append(vecy[ind])
                tarr.append(arr[ind])
            vecx = tvecx
            vecy = tvecy
            arr=tarr
            na = int(np.round(random.gauss(exo_rate,sigma_rate))) - (recycled_particles_rab4 +
recycled_particles_tubular)
            for i in range(na):
                x = random.random()*L
                y = random.random()*L
                vecx.append([x])
                vecy.append([y])
                arr.append((x,y))
            n_part = len(vecx)
            part_arr.append(n_part)
        elif (t+recycleFreqRab4)%freq==0:
            print(f"recycle freq for Rab 4 is \n {recycleFreqRab4}")
            for i in range(recycled_particles_rab4):
                x = random.random()*L
                y = random.random()*L
                vecx.append([x])
                vecy.append([y])
                arr.append((x,y))
            n_part = len(vecx)
```

```
        elif (t+recycleFreqTubular)%freq==0:
            print(f"recycle freq for Tubular is \n {recycleFreqTubular}")
            for i in range(recycled_particles_tubular):
                x = random.random()*L
                y = random.random()*L
                vecx.append([x])
                vecy.append([y])
                arr.append((x,y))
            n_part = len(vecx)
        if t%10 == 0:
            numbers.append(n_part)
    total_sum = 0
    for part in range(0, len(part_arr)):
        total_sum += part_arr[part]
    print(total_sum/len(part_arr))
    plt.figure(figsize=(10,10))
    plt.plot(numbers)
    plt.ylim(bottom=0)
    plt.title('Line Graph of E-cad molecules on surface vs Time')
    plt.xlabel('Time (Every 10th iteration)')
    plt.ylabel('Number of Particles')
    plt.savefig(f"graphs/{mechanism_type}/graph_{currentDate} {currentTime}.png")
    return img_names
def Rab11(n_iter, mechanism_type):
    currentDate = datetime.now().date()
    currentTime = datetime.now().time()
    currentTime = str(currentTime)[:8]
    part_arr = []
    numbers = []
    print("RAB11 CALLED")
    del_t = 0.01
    L=20.0
    v = 1.0
    k = 1.0
    arr = []
    n_part = 100
    vecx = [[] for i in range(n_part)]
    vecy = [[] for i in range(n_part)]
    for i in range(n_part):
```

```
    x = random.random()*L
    y = random.random()*L
    vecx[i].append(x)
    vecy[i].append(y)
    arr.append((x,y))
endo_rate = 10
exo_rate = 10
sigma_rate = 5
freq = 100
recycleFreq = 30
lamRecycleFreqRab4 = 20
lamRecycleFreqTubular = 40
recycleFreqTubular = 40
recycleFreqRab4 = 20
rab4_efficiency = 0.2
tubular_efficiency = 0.4
recycled_particles = 0
recycleRate = 0.3
l0=1.1 # touching distance of molecules
img_names = []
for t in range(n_iter):
    if t%freq==0:
        fig, ax = plt.subplots()
        to_printx =[]
        to_printy =[]
        for i in range(n_part):
            to_printx.append(vecx[i][-1])
            to_printy.append(vecy[i][-1])
        ax.plot(to_printx, to_printy,'o')
        fig.savefig(f"images_my/{mechanism_type}/graph_{t}.png")
        plt.close(fig)
        img_names.append(f"images_my/{mechanism_type}/graph_{t}.png")
        nd = int(np.round(random.gauss(endo_rate,sigma_rate)))
        while(nd>=n_part):
            print("DANGER number of removed particles are greater than particles present, bleak possibility")
            nd = int(np.round(random.gauss(endo_rate,sigma_rate)))
        recycleFreqTubular = np.random.poisson(lam=lamRecycleFreqTubular)
        recycleFreqRab4 = np.random.poisson(lam=lamRecycleFreqRab4)
```

```
        recycled_particles_rab4 = Rab4Recycle(removed_particles_entering_rab4= nd*0.5,
rab4_efficiency=Rab4Efficiency(n_part))
        recycled_particles_rab4_supposed = Rab4Recycle(removed_particles_entering_rab4= nd*0.5,
rab4_efficiency=0.2)
        recycled_particles_tubular = TubularRecycle(removed_particles_entering_tubular=nd*0.5,
tubular_efficiency=tubular_efficiency)
        recycled_particles_tubular_supposed = TubularRecycle(removed_particles_entering_tubular=nd*0.5,
tubular_efficiency=0.4)
        removed_ind = set()
        while(len(removed_ind)<nd):
            x = random.randint(0, n_part-1)
            removed_ind.add(x)
        new_indices = []
        for i in range(n_part):
            if i not in removed_ind:
                new_indices.append(i)
        tvecx=[]
        tvecy = []
        tarr=[]
        if len(new_indices) <= 50:
            rab4_efficiency = 0.3
        elif len(new_indices) >= 75:
            print("len of new_indices greater than 75")
            rab4_efficiency = 0.1
        for ind in new_indices:
            tvecx.append(vecx[ind])
            tvecy.append(vecy[ind])
            tarr.append(arr[ind])
        vecx = tvecx
        vecy = tvecy
        arr=tarr
        print(f"after removing molecules \n {len(arr)}")
        na = int(np.round(random.gauss(exo_rate,sigma_rate))) - (recycled_particles_rab4_supposed +
recycled_particles_tubular_supposed)
        for i in range(na):
            x = random.random()*L
            y = random.random()*L
            vecx.append([x])
            vecy.append([y])
```

```
                arr.append((x,y))
            n_part = len(vecx)
            print(f"after adding molecules \n {len(arr)}")
            part_arr.append(n_part)
        elif (t+recycleFreqRab4)%freq==0:
            for i in range(recycled_particles_rab4):
                x = random.random()*L
                y = random.random()*L
                vecx.append([x])
                vecy.append([y])
                arr.append((x,y))
            n_part = len(vecx)
            print(f"after adding recycled molecules from rab4 back \n {len(arr)}")
        if t%100 == 0:
            numbers.append(n_part)
    total_sum = 0
    for part in range(0, len(part_arr)):
        total_sum += part_arr[part]
    print(total_sum/len(part_arr))
    plt.figure(figsize=(10,10))
    plt.plot(numbers)
    plt.ylim(bottom=0)
    plt.title('Line Graph of E-cad molecules on surface vs Time')
    plt.xlabel('Time (Every 10th iteration)')
    plt.ylabel('Number of Particles')
    plt.savefig(f"graphs/{mechanism_type}/graph_{currentDate} {currentTime}.png")
    return img_names
def RabX1(n_iter, mechanism_type):
    currentDate = datetime.now().date()
    currentTime = datetime.now().time()
    currentTime = str(currentTime)[:8]
    part_arr = []
    numbers = []
    print("RABX1 CALLED")
    del_t = 0.01
    L=20.0
    v = 1.0
    k = 1.0
    arr = []
```

```
n_part = 100
vecx = [[] for i in range(n_part)]
vecy = [[] for i in range(n_part)]
for i in range(n_part):
    x = random.random()*L
    y = random.random()*L
    vecx[i].append(x)
    vecy[i].append(y)
    arr.append((x,y))
endo_rate = 10
exo_rate = 10
sigma_rate = 5
freq = 100
recycleFreq = 30
lamRecycleFreqRab4 = 20
lamRecycleFreqTubular = 40
recycleFreqTubular = 40
recycleFreqRab4 = 20
rab4_efficiency = 0.4
tubular_efficiency = 0.08
recycled_particles = 0
recycleRate = 0.3
l0=1.1 # touching distance of molecules
img_names = []
for t in range(n_iter):
    if t%freq==0:
        fig, ax = plt.subplots()
        to_printx =[]
        to_printy =[]
        for i in range(n_part):
            to_printx.append(vecx[i][-1])
            to_printy.append(vecy[i][-1])
        ax.plot(to_printx, to_printy,'o')
        fig.savefig(f"images_my/{mechanism_type}/graph_{t}.png")
        plt.close(fig)
        img_names.append(f"images_my/{mechanism_type}/graph_{t}.png")
        nd = int(np.round(random.gauss(endo_rate,sigma_rate)))
        while(nd>=n_part):
            print("DANGER number of removed particles are greater than particles present, bleak possibility")
```

```
            nd = int(np.round(random.gauss(endo_rate,sigma_rate)))
        recycleFreqTubular = np.random.poisson(lam=lamRecycleFreqTubular)
        recycleFreqRab4 = np.random.poisson(lam=lamRecycleFreqRab4)
        recycled_particles_rab4 = Rab4Recycle(removed_particles_entering_rab4= nd*0.5,
rab4_efficiency=Rab4Efficiency(n_part))
        recycled_particles_rab4_supposed = Rab4Recycle(removed_particles_entering_rab4= nd*0.5,
rab4_efficiency=0.2)
        recycled_particles_tubular = TubularRecycle(removed_particles_entering_tubular=nd*0.5,
tubular_efficiency=tubular_efficiency)
        recycled_particles_tubular_supposed = TubularRecycle(removed_particles_entering_tubular=nd*0.5,
tubular_efficiency=0.4)
        removed_ind = set()
        while(len(removed_ind)<nd):
            x = random.randint(0, n_part-1)
            removed_ind.add(x)
        new_indices = []
        for i in range(n_part):
            if i not in removed_ind:
                new_indices.append(i)
        tvecx=[]
        tvecy = []
        tarr=[]
        if len(new_indices) <= 50:
            rab4_efficiency = 0.5
        elif len(new_indices) >= 75:
            print("len of new_indices greater than 75")
            rab4_efficiency = 0.3
        for ind in new_indices:
            tvecx.append(vecx[ind])
            tvecy.append(vecy[ind])
            tarr.append(arr[ind])
        vecx = tvecx
        vecy = tvecy
        arr=tarr
        print(f"after removing molecules \n {len(arr)}")
        na = int(np.round(random.gauss(exo_rate,sigma_rate))) - (recycled_particles_rab4_supposed +
recycled_particles_tubular_supposed)
        for i in range(na):
            x = random.random()*L
```

```
            y = random.random()*L
            vecx.append([x])
            vecy.append([y])
            arr.append((x,y))
        n_part = len(vecx)
        print(f"after adding molecules \n {len(arr)}")
        part_arr.append(n_part)
        print(f"after adding recycled molecules back \n {len(arr)}")
    elif (t+recycleFreqRab4)%freq==0:
        for i in range(recycled_particles_rab4):
            x = random.random()*L
            y = random.random()*L
            vecx.append([x])
            vecy.append([y])
            arr.append((x,y))
        n_part = len(vecx)
        print(f"after adding recycled molecules from rab4 back \n {len(arr)}")
    elif (t+recycleFreqTubular)%freq==0:
        print(recycleFreqTubular)
        for i in range(recycled_particles_tubular):
            x = random.random()*L
            y = random.random()*L
            vecx.append([x])
            vecy.append([y])
            arr.append((x,y))
        n_part = len(vecx)
    if t%10 == 0:
        numbers.append(n_part)
total_sum = 0
for part in range(0, len(part_arr)):
    total_sum += part_arr[part]
print(total_sum/len(part_arr))
plt.figure(figsize=(10,10))
plt.plot(numbers)
plt.ylim(bottom=0)
plt.title('Line Graph of E-cad molecules on surface vs Time')
plt.xlabel('Time (Every 10th iteration)')
plt.ylabel('Number of Particles')
plt.savefig(f"graphs/{mechanism_type}/graph_{currentDate} {currentTime}.png")
```

```
    return img_names
def modelECAD():
    currentDate = datetime.now().date()
    currentTime = datetime.now().time()
    print("Generating random value for V between 0 and 1 ")
    n_iter = int(input("Enter the number of iterations :"))
    mechanism_type = str(input("Which Type of Mechanism do you want to run? A. Wild Type B. Rab11 C. RabX1
"))
    img_names = []

    if mechanism_type == "A" or mechanism_type == "a":
        mechanism_type = "wild_type"
        img_names = wildType(n_iter, mechanism_type)
    elif mechanism_type == "B" or mechanism_type == "b":
        mechanism_type = "rab11"
        img_names = Rab11(n_iter, mechanism_type)
    elif mechanism_type == "C" or mechanism_type == "c":
        mechanism_type = "rabx1"
        img_names = RabX1(n_iter, mechanism_type)

    print("pre-processing done!")
    codec = cv2.VideoWriter_fourcc(*"mp4v")
    out = cv2.VideoWriter(f"videos_my/{mechanism_type}/output {currentDate} {currentTime}.mp4", codec, 18,
(640, 480))
    for i in range(len(img_names)):
        img = cv2.imread(img_names[i])
        out.write(img)
    plt.show()
    out.release()
    cv2.destroyAllWindows()
    print("done!")
modelECAD()
```