Two mathematical approaches to a study of T cell motion and activation in the lymph node

by

Monica Delgado Carrillo

B. Mathematics, Universidad de Guanajuato, 2009

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Mathematics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

August 2012

Abstract

T cells are part of the immune system and as such play a very important role in keeping us healthy. One crucial step in the complex process which is the immune response to pathogens is T cell activation.

The general goal of my thesis is to mathematically describe the migration patterns followed by T cells while waiting to be activated in the lymph node. Insight into these migration patterns could lead to better knowledge of the strategies T cells take to make activation such an efficient process.

In order to fulfill my goal I have used two different approaches: one mainly computational and the other mainly theoretical.

On the computational side, I analyzed three-dimensional microscopic movies of mice lymph nodes inside of which labelled T cells are moving. From the movies I extracted the trajectories of the cells. I studied movies from two experimental frameworks, exogenous and endogenous. On the former, more frequent type of experiment, T cells are labelled outside the mouse and then transferred in. The endogenous experiments, on the contrary, involve genetically modified mice whose T cells are born labelled. I concluded that there is a significant difference in labelled T cell motion between the two experimental frameworks. This suggests that previous results from exogenous experiments should be treated with caution due to possible errors introduced by the methods specific to that type of experiment.

On the theoretical side I studied the time it takes for a model T cell to be activated under different scenarios regarding the characteristics of the lymph node as well as of the other cells in it. Since T cells become activated after establishing contact with a specific cell among many similar ones which also move within the lymph node, what I effectively computed was the mean first passage time for a model T cell to reach a defined target within the model lymph node.

Preface

My supervisor, Daniel Coombs has been involved in all areas of this thesis. Raibatak Das, Coombs' postdoctoral researcher, was involved in the development of Chapter 2. They provided guidance and feedback, edited manuscripts, assisted with computational difficulties and gave advice in interpreting results.

Chapter 2 is the result of work in collaboration with the laboratory of Dr. Pasquale Maffia from the Institute of Infection, Immunity and Inflammation at the University of Glasgow, in Scotland, UK. Dr. Maffia and his collaborators own the copy rights to the data. All experiments were performed by Ross McQueenie. McQueenie also helped in describing the experimental methods for Section 2.2.1. I did the analysis, aided by Das and Coombs. Some of the codes I used were taken from [13] either in full or in parts. I used MATLAB for all computational results from this chapter. The idea for the algorithm described in Section 2.4.3 was suggested to us by Khuloud Jaqaman. For Chapter 3 I received help and guidance from Professor Michael Ward.

Figure 1.1 is here reproduced by kind permission of the Nature Publishing Group (License Agreement number 2970990947701). Figures 2.1 and 2.2 are here reproduced by kind permission of Michael W. Davidson of the National High Magnetic Field Laboratory, the author, under the condition that this thesis will only be available online in PDF format. All other figures were designed and created by me.

Table of contents

List of tables

List of figures

List of abbreviations

2PM	Two-photon microscopy
APC	Antigen presenting cell
BC	Boundary conditions
COM	Center of mass
DC	Dendritic cell
FPT	First passage time
FRN	Fibroblastic reticular network
GFP	Green flourescent protein
HEV	High endothelial venules
LHS	Left hand side
LN	Lymph node
MAX	Local intensity maxima
MFPT	Mean first passage time
PLO	Peripheral lymphoid organs
RHS	Right hand side
SPT	Single particle tracking
TCR	T cell receptor
VFPT	Variance of the first passage time

Acknowledgements

I would like to thank my advisor, Daniel Coombs, who took me as his student when I was desperate because none of the UBC faculty I wrote to had replied to me. I really enjoyed being his student and part of his group.

A special thank you goes to Dodo Das who was always there to help. I feel his guidance was esential in the development of the first part of my thesis. Thanks to Isabell Graf who took the time to teach me the basics of homogenization theory, and to her advisor Peter Malte for listening to our ideas and giving his advice as a homogenization expert.

I would also like to acknowledge the financial support and learning opportunities provided by the International Graduate Training Center (by the Pacific Institute for the Mathematical Sciences) throughout my masters, as well as the financial support granted to me by Consejo Nacional de Ciencia y Tecnología from Mexico.

Finally, I would like to thank the city of Vancouver and Canada for welcoming me, and all my friends and family, here and in Mexico, who helped me through the good and bad moments.

Dedication

To my beloved Mexico.

Chapter 1

Introduction

1.1 Biological description of the problem

The immune system is protects the body from foreign agents and disease. In vertebrates the actions of the immune system can be classified in two main categories: innate immunity and adaptive immunity. The innate immune system acts first and provides a first line of defense. It is generic, meaning that it reacts in much the same way to similar pathogens. The adaptive immune system, on the other hand, identifies the pathogen and produces cells which will only attack that specific pathogen. The adaptive system keeps a copy of the cells created specifically to attack the pathogen even after it has been cleared out of the body. By doing this it provides long-lasting immunity to the host, so next time the same pathogen is found in the body the immune system will already have the specific tools to fight it. However, the initial adaptive response is slower and also relies on the innate system.

T cells are a type of lymphocyte (white blood cell) which play a very important role in the adaptive immune response. T cells originate in the bone marrow but then migrate to the thymus to undergo maturation, which is why they are called "T" cells. Mature T cells then enter the bloodstream and migrate to the peripheral lymphoid organs (PLO), organized tissues where adaptive immune responses are initiated and where lymphocytes are maintained. T cells are continually recirculating through these tissues. They enter the PLOs by squeezing between high endothelial venules (HEV) and exit through efferent lymph vessels. Antigen[1] is also carried from the site of infection to the PLOs. Specialized cells, referred to as Antigen Pressenting Cells (APC), carry antigen to the PLOs and display it to lymphocytes. [19]

The lymph nodes (LN) are one type of PLOs. They are highly organized and are located at the points of convergence of vessels of the lymphatic sys-

[1]Antigen - Term used to refer to any substance that can be recognized by the adaptive immune system.

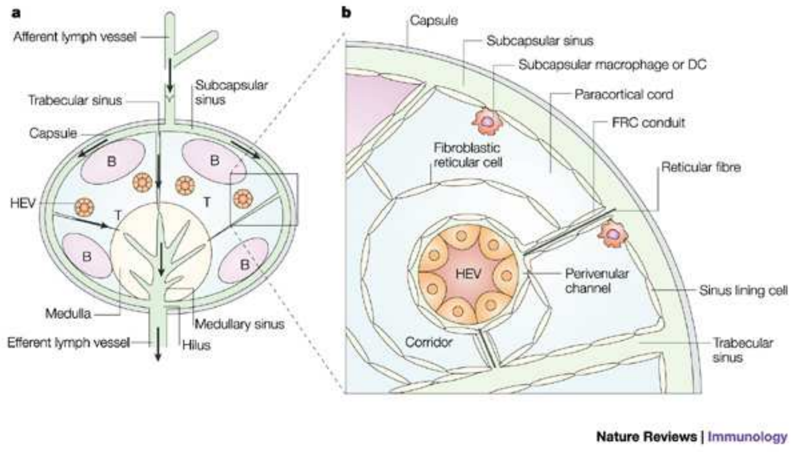


Figure 1.1: Schematic diagram showing the major structural components of a lymph node [24].

tem. One important structure inside the LN is the fibroblastic reticular network (FRN), a network formed by fibroblastic reticular cells and filaments between them. Figure 1.1 shows the architecture of the lymph node.

A key step in the cascade of actions which lead to the adaptive response is T cell activation. T cells are equipped with receptors which allow them to recognize antigen displayed by APCs. Hence T cell receptors (TCR) show an enormous variability (from one cell to other). While in the lymph node, T cells successively scan APCs and read the information presented by them until finding their cognate antigen. If the reading contact with the cognate APC lasts long enough then the T cell is activated and it undergoes proliferation and differentiation, the first steps of adaptive immunity. The descendants of an activated T cell are ready to fight the pathogens (by activating B cells to produce antibody, or by directly killing infected cells).

In particular, dendritic cells (DC) are an important class of APCs. They have long projections called dendrites, which gives them their name.

1.2 Current approaches

While interacting with APCs, T cells change their behaviour in different ways depending on whether they are bound or not, or if they have already been activated. Therefore, analyzing the movement of T cells in the LN is important, for it can lead to better knowledge of the processes that take place inside the LN.

Here, I present results from two different mathematical approaches to describing the behaviour of T cells in the LN. The first approach focused on describing T cell motion from data while the second one was about describing the time it takes for a T cell to find its corresponding APC under different hypotheses about T cell motion. The purpose of both approaches is to gain insight into the strategies of the immune system in making T cell activation an efficient process.

In Chapter 2 I present the work done using the first approach. For this part I analyzed imaging data of labelled T cells inside the LN of a healthy mouse. Thanks to the development of advanced imaging techniques (two-photon microscopy) we were able to acquire time series of quasi-three-dimensional images showing T cells inside a LN. From these I extracted cell paths and characterized them. This has been done in the past, however, we propose a new experimental frame which we think provides more accurate results. Here I present a comparison between the results obtained from the typical experiment and those from our new experimental frame.

The second approach is presented on Chapter 3. Here we studied the first passage time of a model T cell to a fixed target, the cognate APC. We considered different scenarios by varying the motility conditions for the T cell as well as the characteristics of the environment where it moves, i.e. the LN.

Chapter 2

Analysis of two-photon imaging data from T cells moving within living tissue

This chapter is the result of a collaboration with Dr. Pasquale Maffia's team at the University of Glasgow. The data they provided us with was a time series of 3-dimensional images showing T cells inside the LN. Each 3D image corresponding to a time point is in reality a stack of several 2D images, spaced by a couple of micrometers in height, which are taken rapidly and sequentially, and so are regarded as forming one 3D image at a single time point. I worked with several data sets, which varied in the number of time points as wells as the size of the stack. Details of each data set's specifications are given in Table 2.1.

2.1 Two-photon microscopy

Two-photon excitation microscopy [6] is a relatively new fluorescence imaging technique. It has the advantage of allowing living tissue to be imaged at high depth.

2PM relies on the fact that two photons of approximatedly half the energy needed for single photon excitation, can equally excite a fluorophore if the two photons hit it simultaneously. Because that is a lower probability event, fewer fluorophores will become excited, and more excitation events are likely to occur near the focal point, where more photons are concentrated (Figure 2.1). This is in part the reason why 2PM presents less background noise (see Figure 2.2).

Moreover, since lower energy means longer wavelength, the wavelength of the absorbed photons in 2PM is about two times the wavelength the absorbed photons in single photon microscopy. Because lower wavelength gets scattered a lot, this again means less noise in 2PM images, as well as less tissue damage by phototoxicity. [17]

Name	Exogenous/ Endogenous	Number of slices	Number of time points	Imaging rate (s/stack)	Pixels per side	μm per side	Height per slice (μm)
LN2	endogenous	17	120	29.025	512	531.37	2
LN7	endogenous	12	150	20.004	512	531.37	2
LN8	endogenous	12	60	20.816	512	531.37	2
LN9	endogenous	30	20	50.799	512	531.37	2
LN10	exogenous	22	30	45.666	512	425.1	2
Z1	exogenous	25	38	12.821	560	283.4	2.5

Table 2.1: Summary of the specifications of the experiments I analyzed data from. Only those specifications which influence the computational analysis of the data are included.

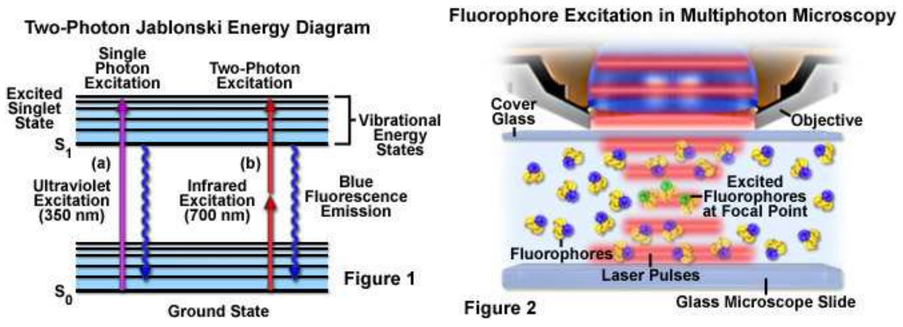

Two-Photon Jablonski Energy Diagram
Single Photon Excitation
Two-Photon Excitation
Excited Singlet State
S1
Vibrational Energy States
(a)
(b)
Ultraviolet Excitation (350 nm)
Infrared Excitation (700 nm)
Blue Fluorescence Emission
S0
Ground State
Figure 1
Fluorophore Excitation in Multiphoton Microscopy
Cover Glass
Objective
Excited Fluorophores at Focal Point
Fluorophores
Laser Pulses
Glass Microscope Slide
Figure 2

Figure 2.1: Two photon microscopy requires two simultaneous events of high wavelength [17].

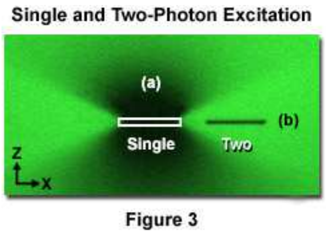


Figure 2.2: Two photon microscopy presents less background noise [17].

2.2 Experiments

Previous studies of T cell motion are based on experiments where cells are extracted from the mouse, then exogenously tagged with a fluorophore protein, and finally transfected back into the mouse. In these experiments only some of the transfected cells then home to the LN. Hence images from this type of experiment show low densities of labelled cells. Moreover, we hypothesize that cells which have been outside the mice show altered dynamics, and also those cells rediscovered in the LN may represent an unusual subset of the basic T cell population.

Our collaborators in the University of Glasgow have performed endogenous experiments on genetically modified mice whose cells express Green Fluorescent Protein (GFP). Because in these genetically modified mice all T cells express GFP, the density of cells shown in images obtained from endogenous experiments is higher. This is good in the sense that more data is available but it also makes the determination of trajectories more complicated.

2.2.1 Experimental methods

To image endogenous T cell populations, inguinal lymph nodes were excised from hCD2 GFP mice and placed in a perfusion chamber. Lymph nodes were maintained at 37°C by perfusion with heated CO_2 independent media. Following staining, approximately 2 million T cells were transferred by intravenous tail vein injection into CD11c YFP transgenic mice.

Multiphoton fluorescence was performed using a Zeiss 7MP multiphoton microscope (Carl Zeiss, Jena, Germany) after excitation by a Coherent Chameleon wavelength tuneable laser (Coherent, Glasgow, UK). Zeiss 10X/0.3 NA air and 20X/1.0 NA water immersion lenses (both Carl Zeiss, Germany) were used to focus light on the sample and to collect emitted light. Fluorescence emission was collected at $<$375nm (autofluorescence), 500-550nm (green) and 600-680nm (red). All samples were excited at 920nm.

2.3 Methods for the inference of cell paths

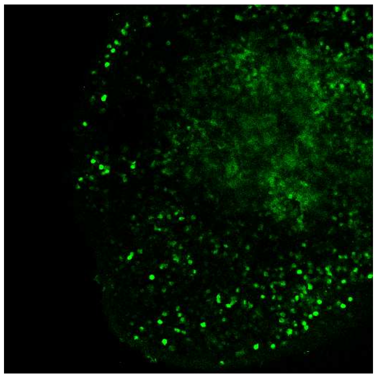
2.3.1 Single particle tracking

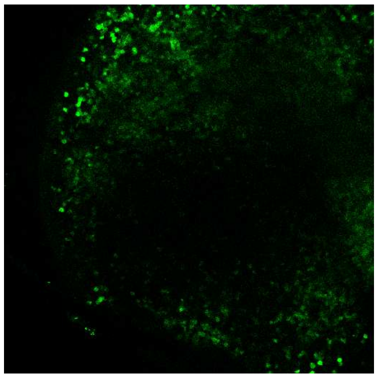
Single particle tracking (SPT) is a widely used technique whereby the motion of biological particles can be observed. The particle of interest is tagged using any of a variety of experimental techniques. Then a series of images is taken by means of a microscope at high temporal resolution. From these one can attempt to extract particle trajectories, a highly non-trivial task that requires joining the dots (cell centers) between frames in a sensible way. In typical SPT experiments, only a small number of particles are tagged, which makes the tracking somewhat easier [18].

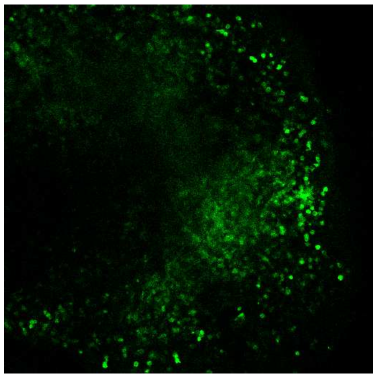
The endogenous experiments we worked with present three major difficulties as compared to typical SPT experiments:

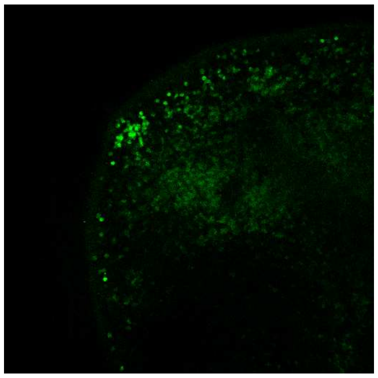
- First, in a strict sense, what I did was not SPT but rather single cell tracking. Cells are much bigger than what we generically call "a particle". This is not really a disadvantage per se, in fact I would say it is an advantage. The problem, however, is that many existing SPT techniques cannot be directly applied to cell tracking.

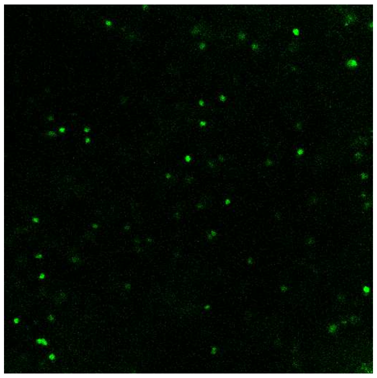
- Second, the high cell density is a big problem. Even by eye it is often difficult to tell which of two cells located close to each other on a


(a) LN2


(b) LN7


(c) LN8


(d) LN9


(e) LN10

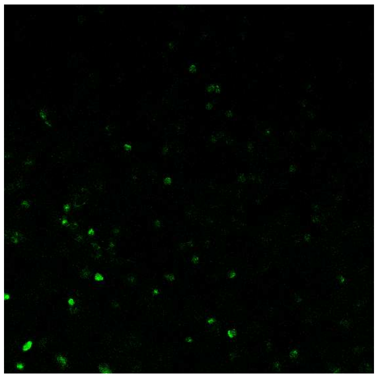

(f) Z1

Figure 2.3: A sample image from each data set. Notice that the bottom row shows the exogenous experiments. All images correspond to $t = 1$ and $z = 1$.

frame moved to where on the next frame. Moreover, background noise increases when having so many bright spots on an image.

- Third, SPT has only rarely been done in three dimensions. Having the third dimension is good because it provides more information about the phenomenon we are studying. However, we found that even once it is known which planes a specific cell lies on, determining its 3D center is a difficult task which has a great influence in the subsequent track analysis.

Hence we can conclude that our problem is how to deal with so much information.

2.3.2 Steps in the tracking process

I have distinguished three main stages in the process of extracting the particle trajectories. The stages were separated in this way because I tackled each stage separately, computationally speaking.

Denoising This is the first stage and consists of filtering the images so that the background noise does not interfere with the subsequent stages. In other words, in this stage a noisy image is transformed into one with better defined cell boundaries.

Detection The next step is to determine what makes a single cell in the image and find all single cells. To facilitate the last step once all cells in an image have been determined each of them is characterized by the 3D coordinates of its center.

Tracking At this stage one has a list of 3D positions corresponding to each time point. The tracking step consists of connecting these positions into discrete trajectories in a sensible way.

The following subsections describe each stage in more detail. A whole section (2.4) has been dedicated to the second stage, detection, because it was the step for which I developed new algorithms. One such algorithm actually uses the same method I used for the tracking step. This is why I left the detection section after the first and third steps' subsections, which basically used methods that already existed.

2.3.3 Denoising with wavelets

As I said before, first the images need to be treated so as to get rid of noise and extract positions of objects that are truly there. Further, one has to dismiss all those objects which seem too small, or otherwise “flawed”, to be real T cells (or whatever cells we are tracking).

For the denoising step I used an algorithm developed to extract bright spots from biological images [21]. The novelty about this algorithm is that it can detect bright spots at different levels of resolution, it is thus well suited to detect cells.

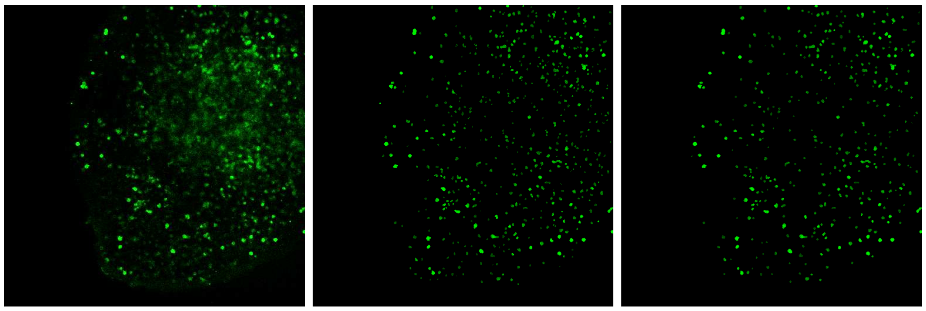
The algorithm was implemented in MATLAB by Henry Jaqaman and François Aguet. The code includes both the denoising and the detection part. However, they both only work in 2D, which is why I split the code in two parts. The denoising part of the algorithm is based on image processing with wavelets theory. I thought about extending it to three dimensions but it is not clear how much that would help. The problem is that the resolution of the images is lower in the third dimension (height) than in each x-y plane. I thought if I used a direct extension to 3D of the denoising algorithm a lot of information might be discarded as noise. Hence, I decided to use the denoising part to process all 2D images individually. Then the 2D denoised images are put together as 3D images for the next step.

At first I used the algorithm with the default parameter for the level of denoising and I noticed that after processing the images with this level of denoising there were many short tracks. So I decided to increase the level of denoising. The higher level of denoising uses more morphological operations to get rid of isolated pixels and other outliers. This is done by means of the MATLAB function *bwmorph*, the higher level additionally performing the processes *erode*, *spur*, *clean*, and *thicken*. By doing that I would get rid of dim spots in the central area, where most cells are located (see Figure 2.4).

I tried both levels and most results here use the higher level of denoising, unless otherwise stated. However, there are advantages and disadvantages to both levels. The higher level sometimes misses spots that are clearly visible by eye in some frames breaking a path or splitting a cell in the z direction.

2.3.4 Track extraction

U-track [13] is a well known SPT software. It is very flexible in that it allows the user to modify many parameters in order to adjust the algorithm to a



(a) Original (b) Lower level of denoising (c) Higher level of denoising

Figure 2.4: An example image in original and after denoising in the two levels. This image corresponds to data set "LN2" with $t = 1$ and $z = 8$.

specific problem.

In short, what the algorithm does is the following: For each pair of consecutive time points, match positions in the first frame with positions in the following frame. This is treated as a linear assignment problem [2]: there is a cost for each match and the goal is to minimize the total cost. To reduce the size of the combinatorial problem, a position can only find a match in a circular area around it (the radius of which is a user-assigned parameter). Not having a match is highly penalized, but it may happen, as each particle can only find a single match.

After the first step one is left with a bunch of generally short tracks. The second step is called "gap closing", here the short tracks previously obtained are matched, this time not on consecutive time steps. At this stage *u-track* offers the possibility of allowing for **merging** and **splitting**. It often happens that two particles in an image come so close to each other that they look as a single cell. The concepts of merging and splitting are related to such events. Merging refers to two cells which had separated tracks before but at a given time point appear as a single position. Splitting, on the other hand, refers to the case when a single track splits into two different ones. If one allows for merging and splitting then two segments' ends (starts) can be matched to the same segment's start (end). The problem with using the merge & split capability is that at the end one has groups of tracks instead of a single cell track. For example, if two tracks merge and later split again into two separate tracks, there is no way of knowing which of the first two tracks corresponds with which of the two last tracks. For that reason in the results shown here I did not allow for merging and splitting.

Next is a list of the *u-track* parameters I modified.

Gap closing time window This sets the maximum number of time steps that a track can be lost and still reappear. Because we have 3D images and the cells are large, bright objects, it is not often that a cell goes out of focus and then comes back. In fact, a cell can only get out of the imaging region if it is on the bottom or top planes, or near the edges of the other planes. Hence I set this parameter to 1.

Minimum track segment length Tracks obtained from the first step which are shorter than this parameter (units of number of frames) will not be considered for the second step (the gap closing). For this parameter I used the default, which is 2.

Search radius lower limit It refers to the search radius mentioned in the description of the first step of the tracking algorithm. Because many cells, specially in the periphery, stay static through out the video, I let this parameter be $0\mu m$.

Search radius upper limit This parameter was set to $10\mu m$. Its value was determined by noting how big were the steps taken by some of the cells whose movement was more apparent by eye. I also considered the fact that the average diameter of a T cell is $7\mu m$.

Flag for linear motion *U-track* allows for three motion models: Brownian motion, Brownian motion plus movement with constant velocity, and random motion plus movement along a straight line but with the possibility of immediate direction reversal. I tested all three and the differences did not appear to be significant (see Figures C.5 to C.7 in Appendix C). The results shown in the main text were obtained using the third motion model.

It is worth mentioning that during the detection step all calculations use units of pixels but before proceeding to the tracking step it is important to scale the coordinates appropiately so that the difference in x,y and z resolution does not introduce bias to the results. To that effect the results of detection were transformed to micrometers using the resolution specifications from each data set.

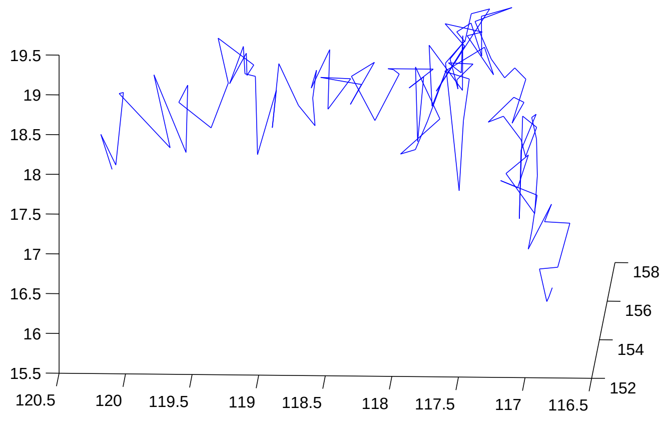


Figure 2.5: Sample trajectory computed by *u-track*. This corresponds to LN2. The scale is micrometers. Notice that it spans about $6\mu m$ on each dimension. By comparing with the video one realizes that this cell did not really move. Its apparent movement is a consequence of a general drift by the whole lymph node. The zigzagging pattern is very likely a consequence of intensity variations in the cell throughout the video.

2.4 The detection step

2.4.1 Spot detector

The detection method from [21] (hereafter called *spotDetector*) is relatively simple so it was not hard to adapt it to work on three dimensions. Here is a description of the original *spotDetector* algorithm:

1. Find local intensity maxima (MAX). In the original 2D version of SpotDetector, a 9x9 window is used to define "local".
2. Find connected regions. Two pixels are in the same region if they have one common vertex or side (hence each pixel has 8 neighbors).
3. For each connected region find its weighted centroid (COM) and the local maxima (computed in 1) that lies on the region. The program keeps a list of the COM of and highest local maximum in each region.
4. If no local maximum lies on the region then the index of the highest value on the region is recorded instead.
5. The program also keeps a list of all other (non highest) local maxima on the region, these are called secondary maxima. Secondary maxima whose distances to both the COM and the MAX do not excede a certain threshold (I used 5 pixels) are deleted from the list.
6. At the end, the COM and the remaining secondary maxima are returned as the centers of the detected cells comprised in the connected region or cluster. The program can easily be modified so the returned centers include the MAX instead of the COM.

The only tricky part in making this algorithm work for the 3D data was defining local in the first step. The 2D version uses a MATLAB built-in function called *ordfilt2* which replaces the value on each pixel with the maximum over the values of its neighboring pixels, where the neighborhood is defined by the notion of local (a 9x9 window). MATLAB does not have an analogous 3D function and although it does not seems like a complicated process to code, it turns out to be very computationally expensive. However, I found that someone had already coded a function *ordfilt3* [5] which nevertheless has the restriction of only working on cubic windows to define locality. This was inconvenient because the x-y scale is very different from the z scale on my images. Nevertheless, I used the 3D version of *spotDetector* with *ordfilt3* using a size 7 cubic window to define locality. Since scales vary from data set to data set (see Table 2.1) this might be more suitable for some data sets than for other.

One last detail from the 3D version is the definition of connected region. In 2D an 8-neighbor lattice was used. For 3D I chose 6-neighbors, i.e. two voxels are in the same region if they share a face. Like in the original *spotDetector*, the program offers the possibility of returning the COM or the MAX (plus the remaining secondary maxima).

My 3D version of *spotDetector*, which has large parts from the original, is given in Appendix B.1.

2.4.2 My detection technique

In the *spotDetector* algorithm, the detection of intensity local maxima and discrimination of them by distance is performed in order to distinguish the centers of several cells that may be very close together, forming a cluster. I proposed an alternative mechanism to separate cells using the 3D information. The idea behind my algorithm (hereafter referred to as *mySpotDetector*) is that clustered cells might appear as a single 2D connected region in some planes but not in all the planes spanned by all cells in the cluster. Hence one might take a 3D connected region and look for the planes in it where such region does not appear as a 2D connected region. The algorithm is explained next.

1. Find all 3D connected regions. I chose to do this defining connectedness in terms of a 6-neighbors matrix, but I also did trials using 18- and 26-neighbors.

2. Loop over all these connected regions.

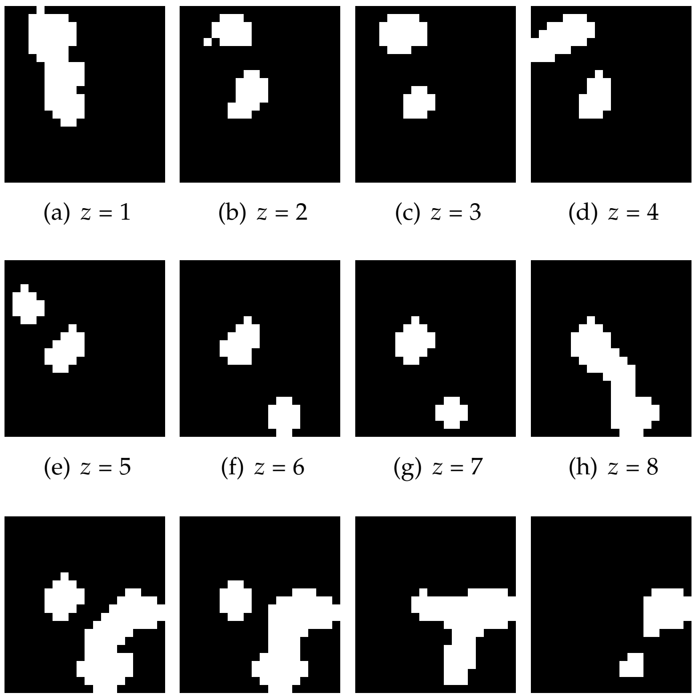
3. Look at the first (bottom most) plane that comprises the current region.

4. Find all 2D connected regions and save them. For these I used an 8-neighbors definition (but also did trials with 4-neighbors definition).

5. Look at the following plane that comprises the current 3D region, and also find all 2D connected regions.

6. Look at the intersections between 2D regions in the current plane and saved regions.

    - If a region in the current plane intersects none of the saved regions, append it to the list of saved regions.


(a) $z = 1$ (b) $z = 2$ (c) $z = 3$ (d) $z = 4$

(e) $z = 5$ (f) $z = 6$ (g) $z = 7$ (h) $z = 8$

(i) $z = 9$ (j) $z = 10$ (k) $z = 11$ (l) $z = 12$

Figure 2.6: Extract from one of the movies to show how my detection technique works. All the planes contained in one three dimensional connected region are shown. Intensity variations have been omitted in this example. See next figure for the resulting saved regions.

- If a region in the current plane intersects one or more saved regions, replace each of those saved regions with the corresponding intersection.

7. Repeat steps 5 and 6 until reaching the last plane that comprises the current 3D connected region.

8. A weighted centroid (depending on intensities) is then found for each saved region. These centers correspond to the cells that were so close to each other that appeared as the current single connected 3D region.

9. One could also start from the last (top most) plane comprising a given 3D connected region and sweep all planes down. This approach will be called "top down" whereas the one described before will be called

"bottom up". It does not appear to make much difference which approach is used.

Figures 2.6 to 2.8 show an example of how this detection method works.

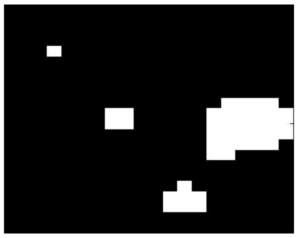


Figure 2.7: Resulting saved regions after processing the example shown in Figure 2.6 with *mySpotDetector*. According to step 8, the resulting detected cells would have centers determined as the intensity weighted centroids of the spots shown in this figure.

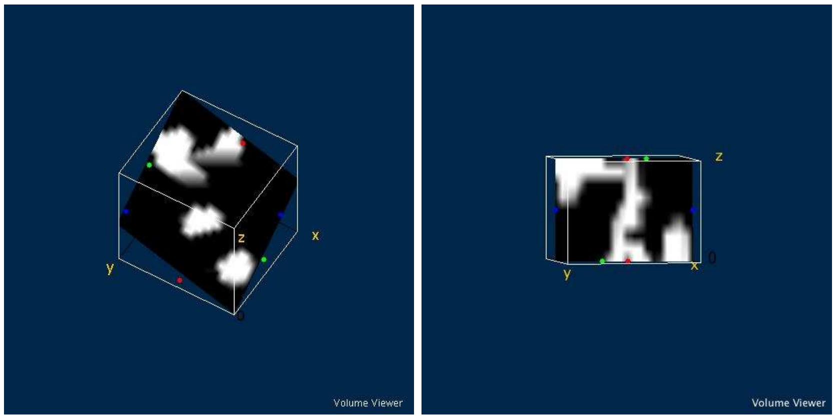


Figure 2.8: Two 3D views of the connected region from the example in Figure 2.6. These were plotted with the Volume Viewer plug-in from ImageJ [20].

2.4.3 Using u-track for detection

Although the results from both previous methods did not seem too bad at first, when feeding them to *u-track* for the tracking step, the program would have difficulty processing the information and often would show errors in computing the matrices to solve the linear assignment problem. Because of these errors we wanted another detection method. When consulting with the authors of *u-track*, it was suggested to us that we used *u-track* in 2D for the detection part. As it turned out, after doing this for the detection part the tracking step would then go smoothly.

In this algorithm, *u-track* is used separately on the stacks corresponding to each time point. Here the z-position is the analogous of time in the normal tracking process. *U-track* parameters are set to allow only for very short motion and no gaps (because one cell cannot disappear and then reappear skipping one intermediate plane). The parameters used were: gap closing time window 1, minimum track segment length 5 (to reduce the number of tracks included in the gap closing step), search radius 1 (except for "z1" which has higher resolution and so the search radius was allowed to be 1 or 2), and the linear motion model was assumed to be random.

The series of 2D positions input to *u-track* are obtained by using the original *spotDetector* on each plane individually.

The output of using *u-track* as a detection method is a series of positions in the planes spanned by each cell. Coordinates of each cell are then determined in the following way:

- For x and y, take the average of the x and y positions in each of the planes the spot spans.
- The z-coordinate is also an average of the vertical positions of the planes where the spot appears. This average is weighted with the intensity of the cell on that plane normalized by the total intensity of the cell.

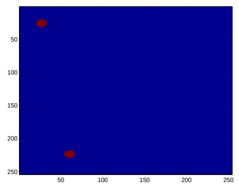
Abbreviated name	Program name	Section where it is described	Specific parameter
sd3D-MAX	B.1. *spotDetector3D*	2.4.1	choice = 0
sd3D-COM	B.1. *spotDetector3D*	2.4.1	choice = 1
mySD-bup	B.2. *mySpotDetector*	2.4.2	bt = 0
mySD-tdown	B.2. *mySpotDetector*	2.4.2	bt = 1
u-track	B.3. *detecttr*	2.4.3	

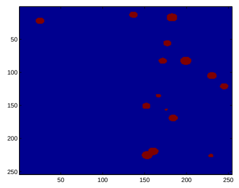
Table 2.2: Summary of the detection methods used. See Appendix B for the codes to which the second column refers. The specific parameter is used in the codes to distinguish between two variations of the same algorithm.

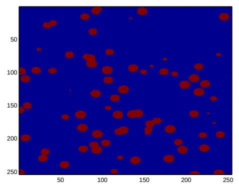
2.5 Testing

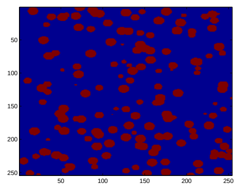
I mainly focused on testing the algorithm I created, that is *mySpotDetector*, although I did try a couple of tests on the existing methods (*u-track* and *spotDetector*).

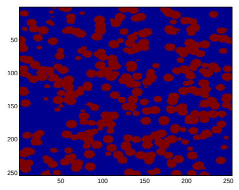
To test the detection methods I generated some 3-dimensional positions which were to be the centers of the cells. These centers are constrained to a box of size 254x254x64 and cells are assumed to be spherical of constant radius r. Then, the bottom plane of the box is "imaged": an image corresponding to that plane shows in black those pixels which fall inside a sphere. The next three planes are not imaged, but the fourth is. 17 images are generated in this manner and then processed with the detection methods. See Figure 2.9 for sample simulated planes. The interested reader can find the code for this test on Appendix B.4.


(a) 10


(b) 100


(c) 500


(d) 1000


(e) 2500

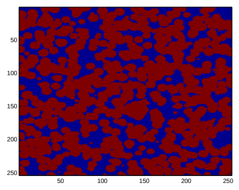

(f) 5000

Figure 2.9: Sample simulated images with $r = 6$. Caption of each subfigure indicates number of simulated cells.

It was difficult to come up with a way to measure error and this part still has to be improved. I computed two errors: Error 1 is the sum of the distance between each simulated cell and the closest center found by the detector, normalized by the number of cells simulated; Error 2 is the relative error between the estimated and true number of cells. Notice that Error 2 is not precisely the fraction of cells missed since many of the computed positions do not correspond to any of the real positions.

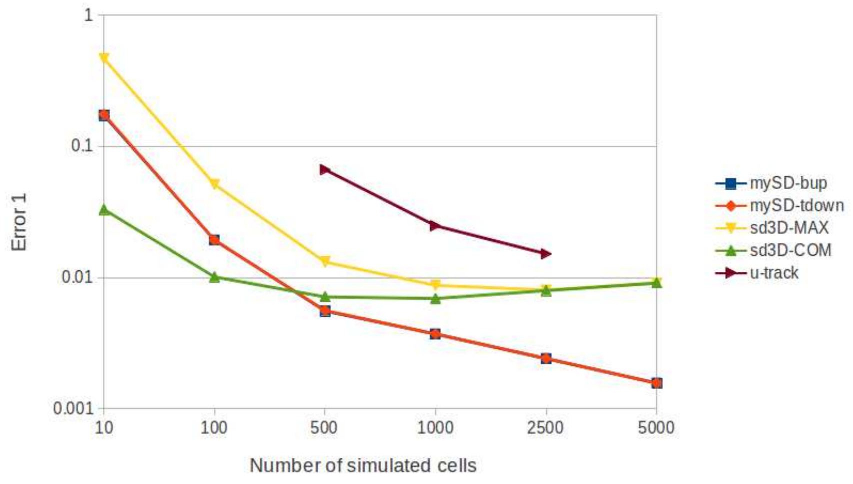


Figure 2.10: Error 1 as a function of the number of simulated cells. The vertical axis units are micrometers and it is shown in log scale. It was not possible to test *u-track* when there are only a few or too many simulated cells (cases with 10,100, and 5000), since in such cases the algorithm returns no found cells.

Figures 2.10 and 2.11 summarize the results obtained.

The large relative error between the estimated and true number of cells (Error 2) is due to the high percentage of cell overlapping (see Figure 2.9). Taking that into account, one can conclude that the fact that Error 1 is so small, especially when there are too many simulated cells, is also due to cell overlapping. It implies that cluster centers are quite close to the true centers of the simulated cells.

Notice that, at least in terms Error 2, the method that does best is *mySpot-Detector*. The reason is that this method attempts to distinguish individual cells in a cluster by using only shape information, while the other methods rely on intensity variations (which the simulated images do not have).

I tried to include the detection method which uses *u-track* in this testing procedure but I encountered several problems. When there are only a few cells or too many of them simulated (cases with 10,100, and 5000) the algorithm returns no found cells. For the cases where some cells were found, *u-track* returned several warning messages which indicate that it is not getting along well with the simulated data. So this method of detection

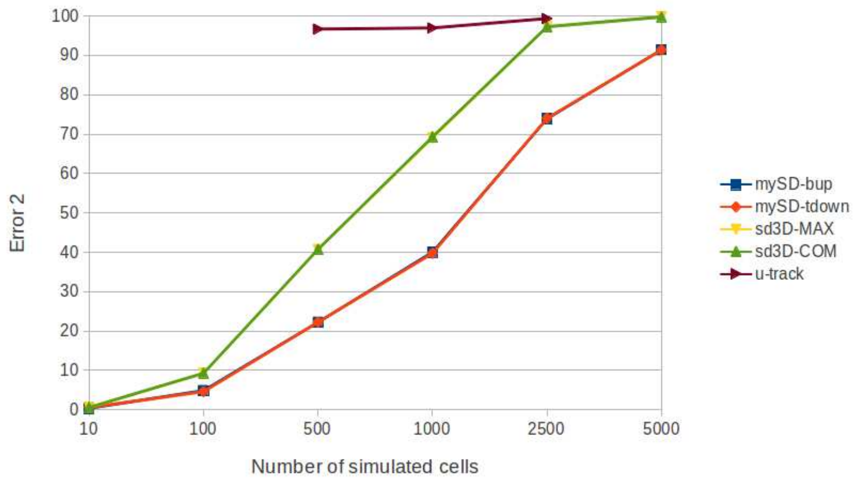


Figure 2.11: Error 2 as a function of the number of simulated cells, shown as percentage. It was not possible to test *u-track* when there are only a few or too many simulated cells (cases with 10,100, and 5000), since in such cases the algorithm returns no found cells.

does worse in the test than the other methods. The problem is that the 2D positions returned in the first step of the algorithm are not cell centers, but cluster centers. Clusters probably vary too much from one plane to the next one (in fact, mySpotDetector uses this variability to distinguish cells) and that is why *u-track* cannot build trajectories succesfully.

I think this does not necessarily mean that the methods are bad, but maybe the simulated data is too far from its experimental counterpart and this is why they fail. Indeed, the failure comes from the fact that there are no local maxima since there is no variation in intensity at all. More tests are needed on simulated data that includes noise and different shapes for the cells.

Even though *u-track* has been tested, results shown in [13] are only for 2D, so that it is not clear if the method was also tested in 3D. Therefore, I created a function (*testtrack*, see Appendix B.5) which generates a number of simulated Brownian paths with a chosen diffusion coefficient, saving their positions at certain time points. I wanted to build the simulation in such a way that positions were only recorded if the particle is inside the imaged box (i.e. a rectangular box of low height as before) but too much information was lost and the tracker did a very bad job, returning only a handful of trajectories even when I was simulating 1000 tracks. This issue already raises a warning, although we had already suspected that the low resolution in the z direction was the biggest problem we had with the data.

Nevertheless, I decided to go one step back and test *u-track* in a free diffusion situation, that is without the box constraints. Again I had trouble finding a good way of measuring the error. Just comparing the original number of tracks with those output by *u-track* does not seem to give a lot of information. In [13] the authors present their results from testing in terms of percentage of misses, but they do not explain how they assess whether a given trajectory was "missed" or not.

In summary, testing needs improving, especially in terms of better measuring whether or not the results are acceptable.

2.6 Analysis

From the results of the tracking process I analyzed some properties that can be rapidly computed and give an idea of the behaviour of the cells in terms of motion. These properties are

Turning angle Angle formed by three positions of a particle in consecutive time frames, as measured by the dot product.

Step size Distance travelled by a particle in one time step.

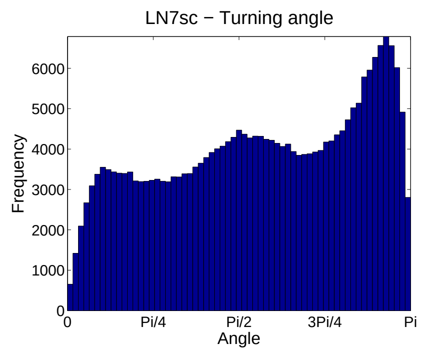
Average step size Average step size over all steps taken by a given particle.

Track diameter Maximum distance between any two positions of a given track.

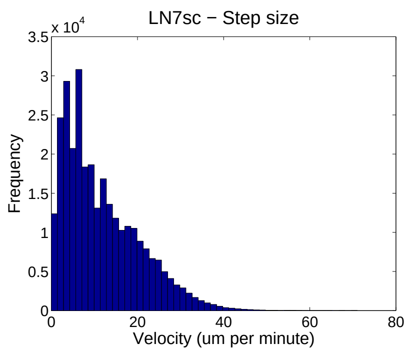
Track length (or life span) Number of time points a given track spans. Effectively computed as the difference between the last and the first time points where the particle is present.

For each set of results I got I computed these properties for every trajectory and then made histograms showing this information. For example, Figures 2.12 and 2.14 both show results from data set LN7 but the first one corresponds to the analysis using the detection algorithm *mySpotDetector* while the second one corresponds to using *u-track* as method of detection.

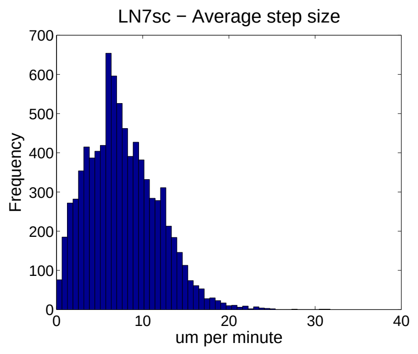
Notice from Figures 2.12(a), 2.13(a), and 2.14(a) that for the endogenous experiments there seem to be a preference for turning almost completely in



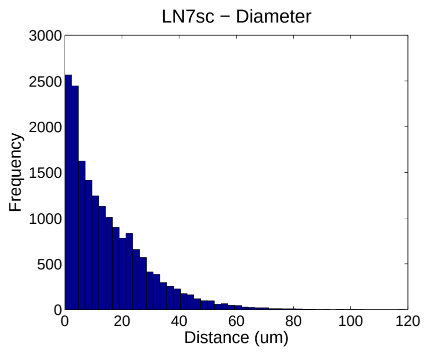
(a)



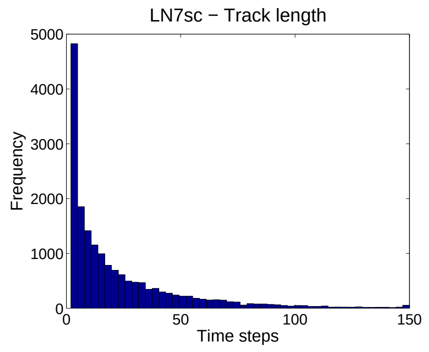
(b)



(c)

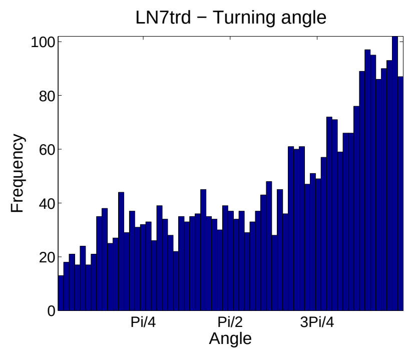


(d)

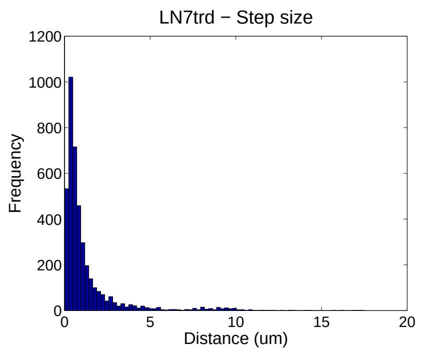


(e)

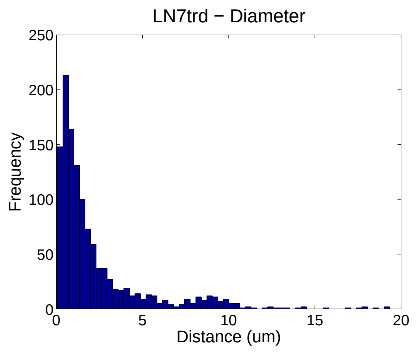
Figure 2.12: Histograms for data set LN7 using *mySpotDetector*-bup as detection method with the lower level of denoising.



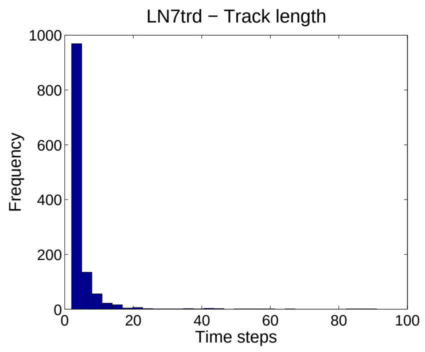
(a)



(b)



(c)



(d)

Figure 2.13: These histograms correspond to the analysis of the endogenous data set LN7, using *u-track* as method of detection and with the lower level of denoising.

the opposite direction (compare also with Figure 2.15 which shows results for one of the endogenous experiment). The peak at an angle of π would not be expected for random motion. In fact, no preferred angle would be expected for random motion.

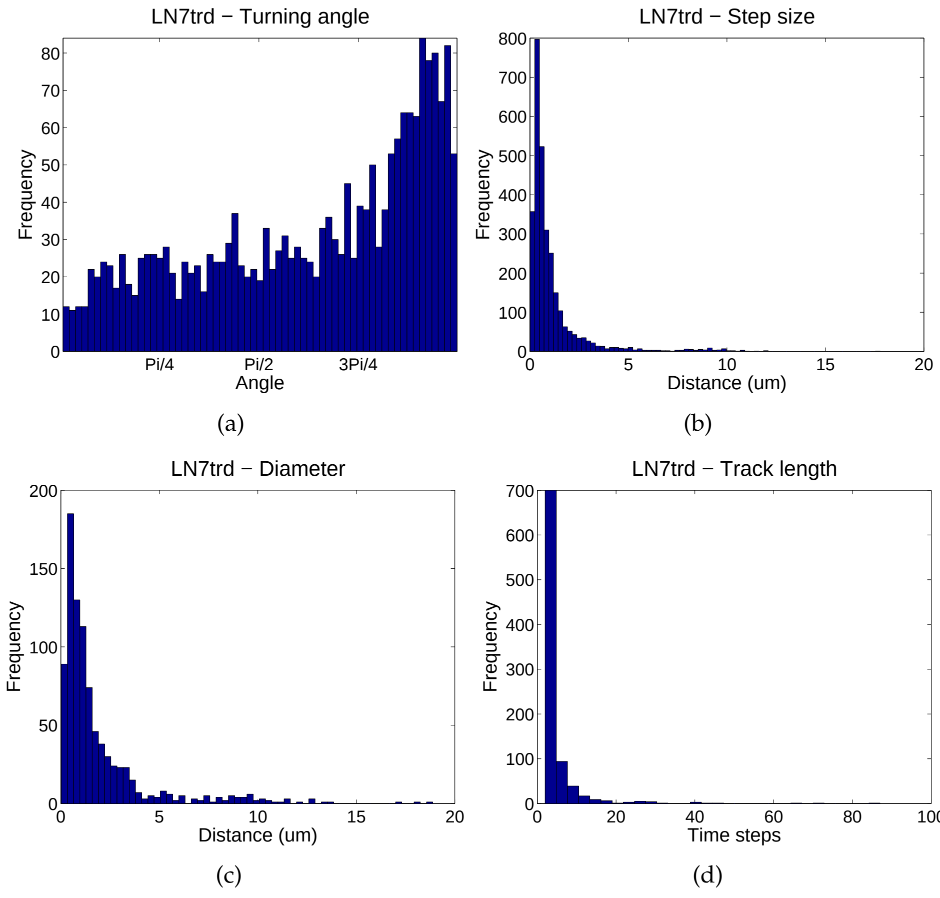
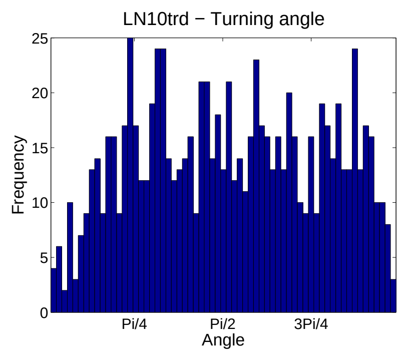


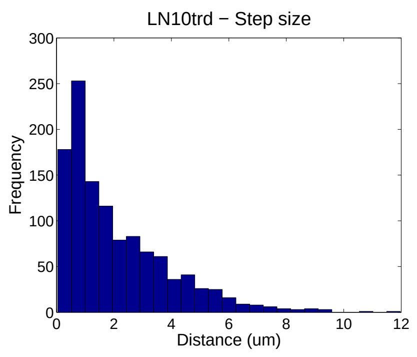
Figure 2.14: These histograms correspond to the analysis of the endogenous data set LN7, using *u-track* as method of detection and with the higher level of denoising.

2.6.1 Discrimination of tracks

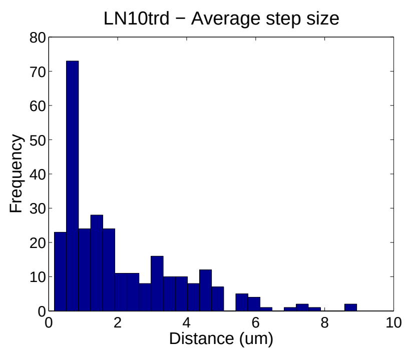
The mentioned bias in turning angle could be an important feature but we first needed to rule out the possibility that it was a consequence of the methods used and not of the true intrinsic cell dynamics. On Figure 2.16



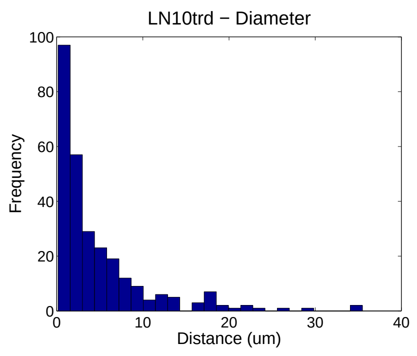
(a)



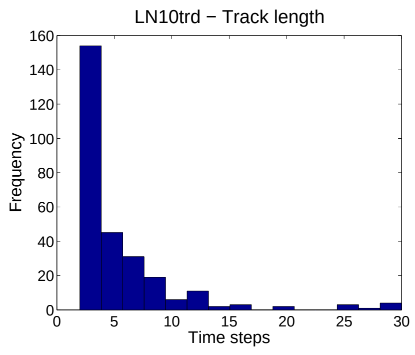
(b)



(c)



(d)



(e)

Figure 2.15: These histograms correspond to the analysis of exogenous data set LN10, using *u-track* as method of detection and with the higher level of denoising.

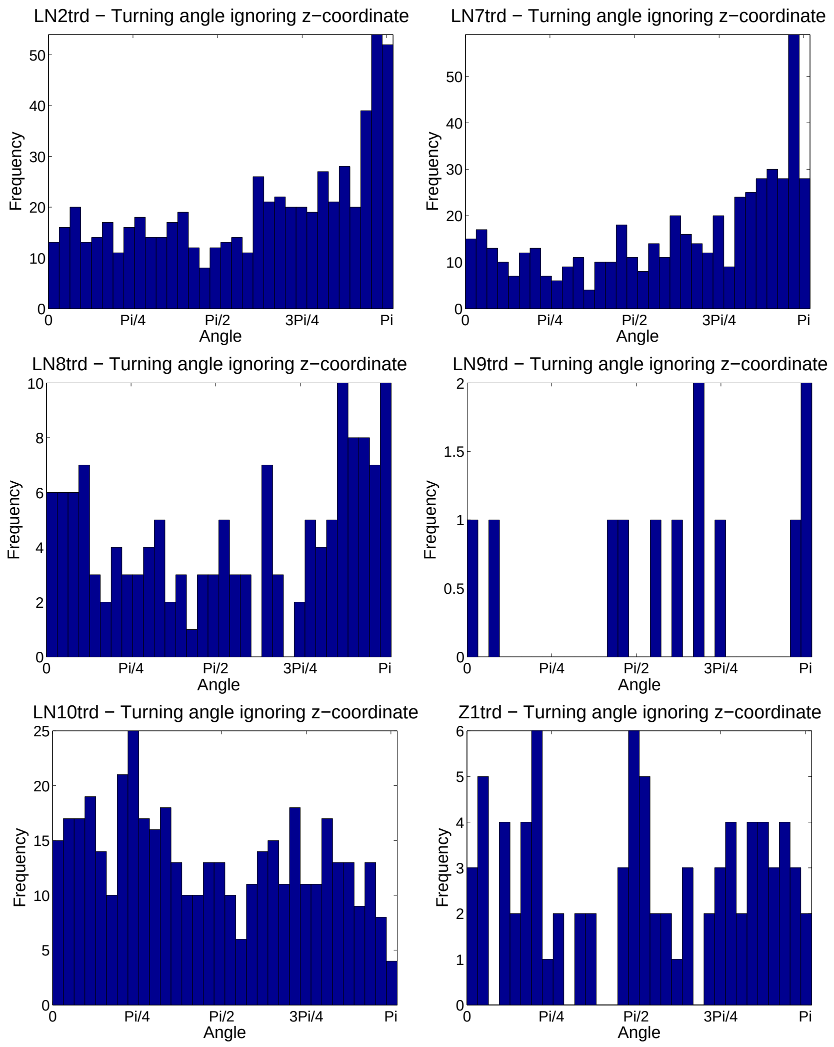


Figure 2.16: Histogram of turning angle ignoring the z-coordinate for all experiments with the higher level of denoising. Only tracks with average step $> 3\mu m$ and with at least 5 frames are included in these histograms. Compare the endogenous versus the exogenous experiments (bottom row).

we ignored the z-coordinate information. We did so because given the way this coordinate is assigned, there is a lot of oscillation in it, specially when the cell is only taking short steps (see Figures 2.17 and 2.18).

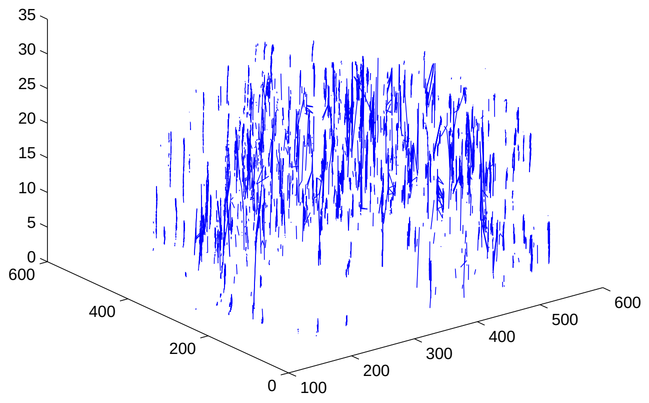


Figure 2.17: 3D representation of all tracks found for LN2 (higher level of denoising, *u-track* as method of detection). Scale is in micrometers.

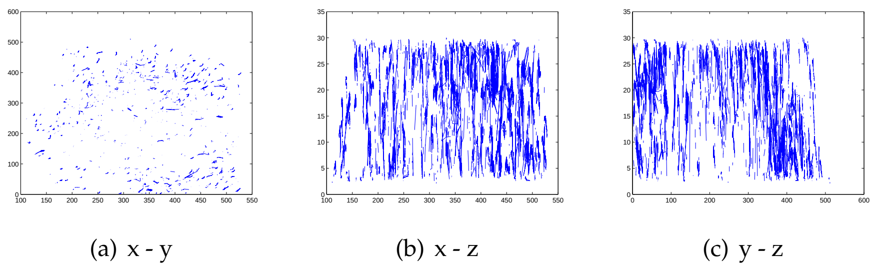


Figure 2.18: Planar projections of the tracks extracted from LN2. Scale is in micrometers.

Another approach was to separate the list of positions from a trajectory into two lists: odd and even time points. Then compute the histograms for each list separately. If the cell, or one of its coordinates, was oscillating in such a way that it was going back and forth from one "side" to another, then these histograms would show the true turning angle distribution. This

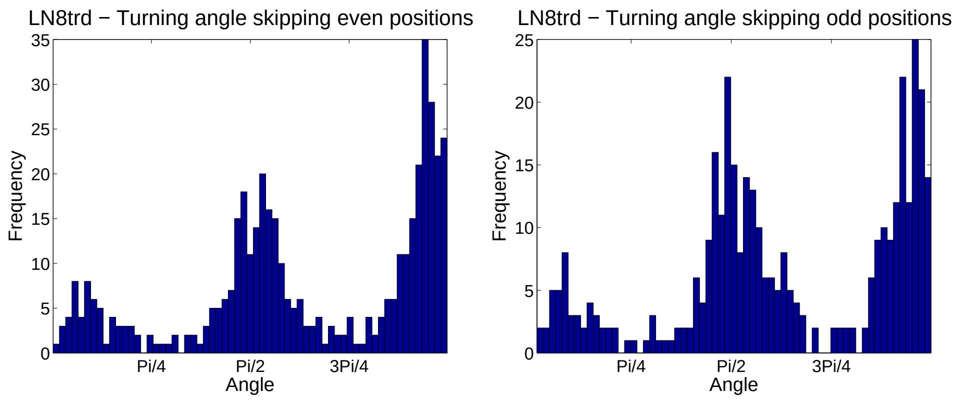


Figure 2.19: Turning angle histograms for odd and even time points separatedly. This results correspond to LN8 processed with the higher level of denoising and *u-track* as tracking method. The z-coordinate, unlike stated in Section 2.4.3, was assigned as the simple average of the planes spanned by a cell (i.e. no weighting).

was not the case however, as it can be seen from Figure 2.19.

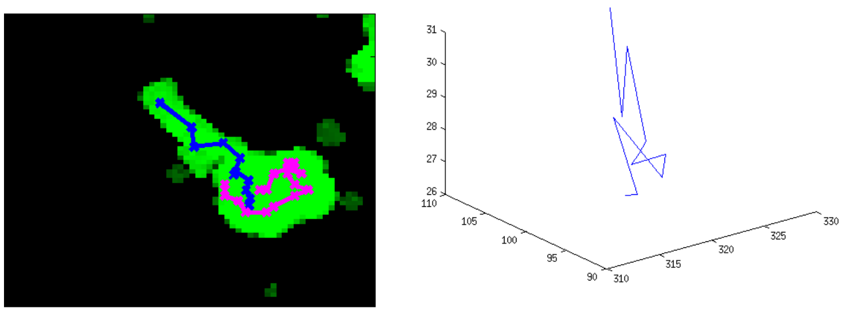
Because we observed many very short tracks (see insets (d) and (e) from Figure 2.15, and bear in mind that the diameter of a T cell is about $7\mu m$) we decided to restrict our analysis to the longer tracks. We also thought that the bias in turning angle could be being introduced by the short tracks as well.

Moreover, I noticed for a few examples that the average distance travelled during a time step is less than $1\mu m$ for tracks that correspond to stationary cells. For that reason I computed the number of tracks whose average step is greater than $3\mu m/min$, which is roughly equivalent to more than $1\mu m$ per frame. Results are shown on Table 2.3 as well as on Figure 2.16.

By comparing the tracking results with the videos I concluded that most of the short tracks are part of a bigger trajectory that the algorithm failed to connect (see Figure 2.20). Hence the fact there are so many short tracks does not mean, as one could imagine, that cells move so much that they disappear from the imaged region in just a few frames.

Name	Level of denoising	Tracks w/ big avg steps	Long tracks
LN2	higher	540	100
LN2	normal	753	140
LN7	higher	326	56
LN7	normal	530	90
LN8	higher	161	30
LN8	normal	248	34
LN9	higher	23	3
LN9	normal	43	9
LN10	higher	98	40
LN10	normal	111	41
Z1	higher	76	14
Z1	normal	158	23

Table 2.3: Number of tracks which are effectively moving, according with the criterion of having average step greater than $3\mu m/min$. The third column shows how many of the moving tracks are at least 5 steps long. Shorter tracks do not contribute significant information.



(a) (b)

Figure 2.20: (a) Example of how the process often misses part of the trajectory. The blue line shows what the computer found to be the full track, the magenta line is the rest of the track, which I found by eye. For each time point in the trajectory I chose the z plane where the cell looked bigger in area and then add all those images up to obtain the trace shown in the picture. The extract comes from experiment LN2. (b) 3D view of the part of the track which was detected by *u-track*.

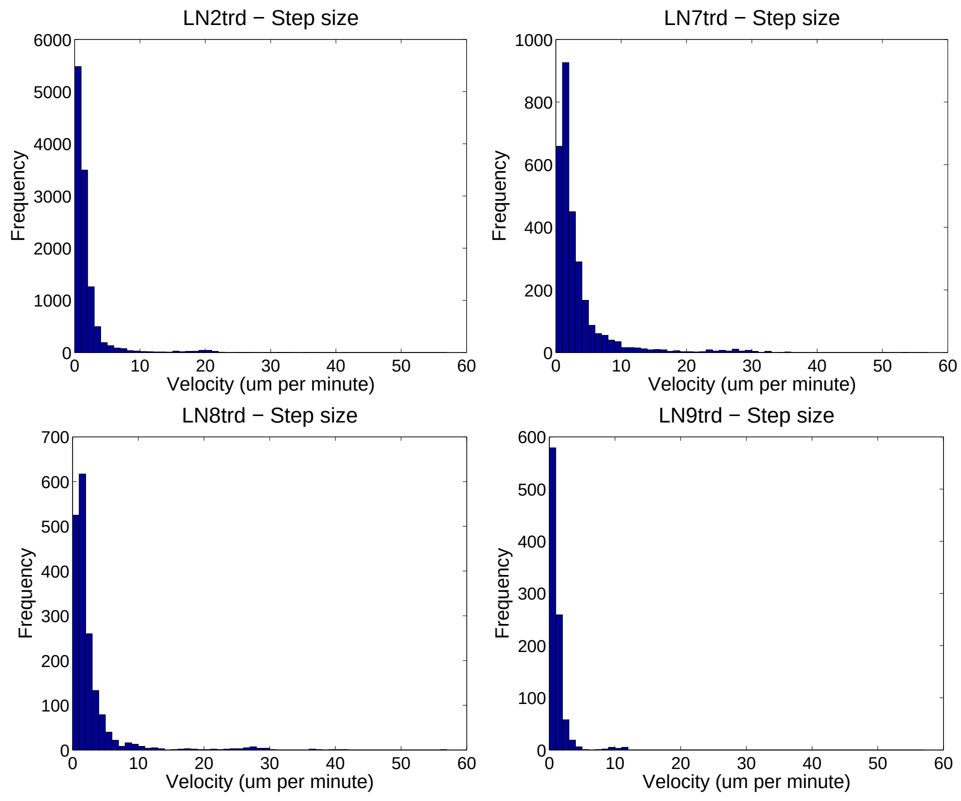
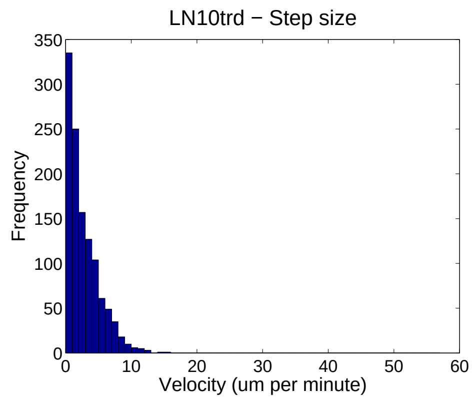


Figure 2.21: Histograms of step size corresponding to the endogenous experiments.

2.6.2 Exogenous vs endogenous

We thought that an interesting feature to compare between the exogenous and endogenous experiments was the cell velocity. Two of the properties described on Section 2.6 are related to cell velocity, namely step size and average step size. Step size refers to the distance traveled by a given cell in a given single time step, whereas average time step is the mean of the step sizes of all steps from a given track. Both quantities can be scaled to measure distance traveled per minute (which has actually been done in all the plots displayed here).

I used the step size information to test our hypothesis that cells present altered behaviour in exogenous experiments. We asked ourselves if, given the experimental results that we have, was there enough information to reject the hypothesis that the average velocity is the same for both types



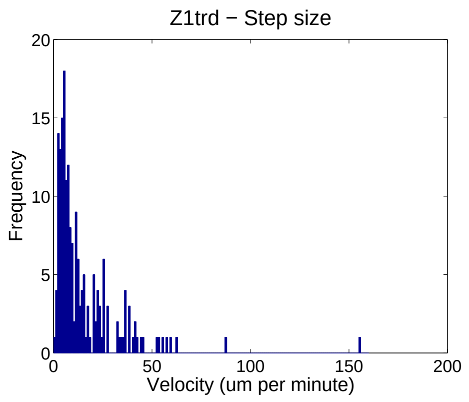


Figure 2.22: Histograms of step size corresponding to the exogenous experiments. The obvious outliers from experiment Z1 were deleted before bootstrapping.

(endogenous and exogenous) experiments. I found that we can reject that hypothesis with a confidence level of at least 95%.

To take advantage of all the data that we have, we used step size instead of average step size. In this way I have 17,261 data points from the endogenous experiments and 1,349 data points from the exogenous experiments. To obtain the distribution of mean velocity from this data, I took 1000 bootstrap samples from each type of experiment and computed the average of those samples. The bootstrap sampling was done in MATLAB.

The hypotheses to test are

$$H_0 : v_{exo} - v_{endo} = 0$$

$$H_a : v_{exo} - v_{endo} > 0$$

where v_{exo} refers to the average velocity of a cell from an exogenous experiment while v_{endo} refers to the average velocity of a cell from an endogenous experiment

Because the cells from one of the exogenous experiments, namely LN10, exhibit very rapid movement as compared to the others, I bootstrapped the data from the two exogenous experiments separately as well as all together. A high amount of movement is a natural consequence of the experimental conditions which is in general reduced by other methods but can be worse in certain experiments, like it is the case for this particular movie.

Tables 2.4 and 2.5 and Figure 2.23 show the results obtained. From Table 2.5 we can see that cells from the exogenous experiments move significantly faster than those from endogenous experiments. In fact, from Table 2.4 we see that the distributions of average velocity do not overlap, so technically our conclusion has a significance level of 0%. Although we would need to analyze more experiments before drawing such a strong conclusion, this result is highly suggestive of an important difference in labelled T cell motion between exogenous and endogenous experiments.

Similar results were obtained when using the normal level of denoising.

Variable	Minimum	Maximum
v_{endo}	2.0902	2.2662
v_{exo}	3.7452	5.0438
v_{Z1}	11.0362	16.1753
v_{LN10}	2.5323	2.9707

Table 2.4: Maximum and minimum values of bootstrapped average velocities. Units are μm/min. v_{endo} is the bootstrapped average of 1000 samples collected from all four endogenous experiments while v_{exo} is the corresponding bootstrapped average of 1000 samples from the two exogenous experiments. Because the cells from LN10 exhibit very rapid movement as compared to the others, the data from the two exogenous experiments was bootstrapped separately as well as all together.

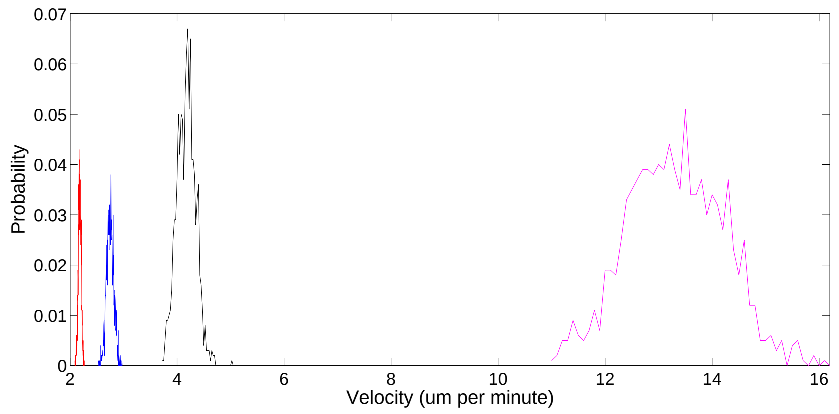


Figure 2.23: Distributions of bootstrapped average cell velocity: v_{endo} in red, v_{exo} in black, v_{Z1} in magenta, and v_{LN10} in blue.

Variable	Confidence interval
$v_{exo} - v_{endo}$	1.6772 - 2.3442
$v_{Z1} - v_{endo}$	9.3529 - 12.9622
$v_{LN10} - v_{endo}$	0.4314 - 0.7080

Table 2.5: Confidence intervals of 95% of $v_{exo} - v_{endo}$. Units are $\mu m/min$.

2.7 Discussion and future work

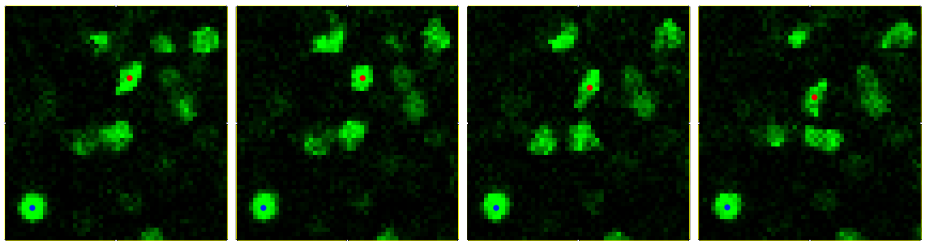
Although I did not obtain as many results from this project as I would have liked, I gained a lot of insight into the computational problem it poses. I think the reason why tracks are apparently easy to identify by eye is because our eyes and brain incorporate a lot of information, including shape and history. Unless a cell has too many neighbors, one can often determine unequivocally what is the position of a cell in the next frame by indirectly using information about its shape and its past trajectory.

Hence, shape plays an important role in this problem. It is true that all cells have more or less the same shape, and so it is not possible to perform the tracking by identifying the cells like one would with people. However, when cells move they change their cytoskeleton following more or less the same patterns which might be helpful in the tracking process. On the other hand, static cells look more round (see Figure 2.24).

I think one ought to consider, if not the shape, at least the whole area (or volume) of a cell instead of just attempting to form tracks out of a list of centers. From one time point to the next one the area occupied by a cell in the first frame often overlaps the area it occupies in the following frame. I am certain that by considering the whole volume of the cell and not just its center, the problems brought by the oscillations in coordinates, such as the strong peak in turning angle at π, would be overcome.

There is another piece of information that I think should be incorporated into *u-track*: the fact that cells do not simply disappear from one time step to the next. Because we are imaging in 3D they do not go out of the imaging region unless they are in the bottom/top planes or near the edges of the planar images. I predict that this would prevent the algorithm from breaking trajectories into several pieces, at least partially.

One other problem I did not manage to solve is the general drift that the whole lymph node undergoes. It is evident from the movies, when one looks at the static cells, that there is a general drift going on. Such drift is not obvious when the movie is played but only when one looks at the first



(a) $t = 6$ (b) $t = 7$ (c) $t = 8$ (d) $t = 9$

Figure 2.24: These images correspond to frames 6 to 9, plane $z = 1$, of experiment LN2. Notice how the shapes of the marked cells change from one frame to another. The cell marked with a blue dot is static while the one marked with a red dot is not.

and last frames. It is because of this drift that the diameters and average steps from the tracks corresponding to static cells are not so small.

For all the T cell images we have, we also have images where the boundary of the LN is tagged (Figure 2.25). We thought that from this images we were going to be able to estimate the size and direction of the drift. I tried to do so by computing the intensity weighted centroid of the LN's boundary for each time point, and then compute the vector of movement of this centroid from one frame to the next one. This vector would then be substracted from all cell positions in all future frames. I did this with one set but in doing so I noticed that some cells actually seem to be drifting in a very different direction than what the vector of weighted centroid displacement showed. Finding a better way to measure the drift is on the list of future work.

These and other factors are the reason why we think the tracking results need a lot of improvement yet.

Lack of adequate visualization tools is one more problem that, if overcome, could help improve the results. For example, at one point I engaged in the task of generating a considerable amount of trajectories by eye. I ended up not using these results simply because they were written on a piece of paper. I am convinced that it would have been different if the software we used to play the movies, which is only in 2D, had allowed me to store as part of the movie the trajectories I was detecting. What I was planning to do with these trajectories was train the parameters of the tracking, and maybe also the detection, algorithms. Visualization of the movies with the superimposed tracks in 3D is also important to validate the results and detect problems in the methods. I do not think I have the right training

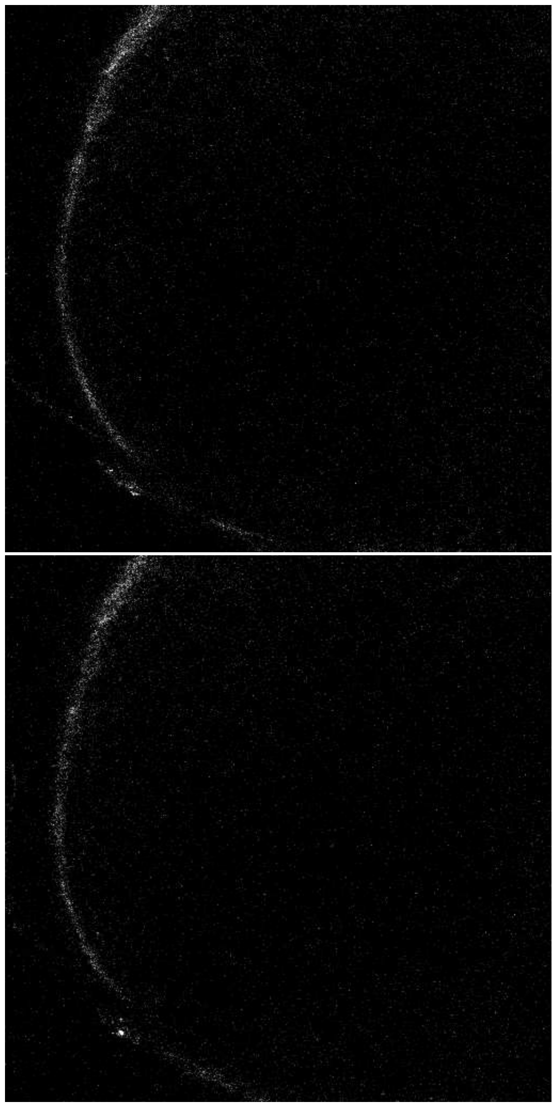


Figure 2.25: Microscopy images showing the contour of the LN. These images correspond to the first and last time points, plane $z = 1$, from data set LN7.

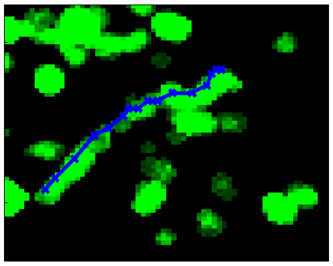


Figure 2.26: This figure shows a trajectory found by eye which the computational process missed. For each time point in the trajectory I chose the z plane where the cell looked bigger in area and then add all those images up to obtain the trace shown in the picture. The extract comes from experiment LN2.

to build my own software to solve this problem. However, I spend some time looking for existing solutions and even talked to some experts on visualization of biological images. Unfortunately, I did not find a satisfactory solution.

Besides improving the detection and tracking methods one natural extension to this work is to analyze more experiments, especially of the exogenous type, in order to confirm the results from Section 2.6.2.

Further extensions of this work include implementing some statistical method to infer the trajectory of a cell between time steps or once it has left the imaging region. Also, evaluating the relationship between movement patterns and zones in the LN, the influence of zones being not only suggested by theory but also by observing the videos.

Chapter 3

First passage time of a T cell to an APC

In this chapter I present some calculations of the mean time it takes for a naïve T cell to reach a specific APC, among many non-specific ones, inside the LN. First passage time calculations have been performed in the past for other problems related to T cell activation [27]. The T cell is taken to be a point in three dimensional space. The APCs, which in reality are bigger than T cells, are model as spheres, first with fixed finite radii and then with asymptotically small volume. While T cells are assumed to be mobile, APCs are static. The lymph node is in most cases a sphere, for the sake of simplicity in explicit calculations, but most results hold for any sort of convex shape.

The idea of analyzing this problem came from two papers [11, 12] which studied a similar problem using computational tools. The authors simulated the movement of T cells in the LN, including other cells in the model. On the first paper the goal was to find how population-level parameters relate to single-cell properties in the process of pathogen killing by T cells. It was found that the time it takes for a T cell to kill an infected cell has a bigger influence on the killing efficency than the time needed for a T cell to find the target cell. The second paper studied the influence of the FRN in the probability of two cells finding each other.

The results in this chapter are theoretical, that is, not based on nor compared with data. Hence, a predetermined model has to be assumed for cell motion. For the most part, I assumed that T cells perform isotropic Brownian motion. In Section 3.4 I considered the case of space dependant diffusion.

Another factor involved in the calculations is the influence of the lymph node's boundary. Because T cells enter the lymph node mainly through HEVs, located along veins that cross the LN, they could show up inside it at virtually any point. They however would always exit through the same spot, which could be modelled as a hole on the LN's boundary. Nevertheless, cells do not exit when they first "bump into" this hole, but they continue their search until they receive a signal to exit [24]. Hence I considered various different scenarios for the boundary conditions on the LN.

Since T cells are assumed to perform Brownian motion, then their concentration $u(x,t)$ at time t on the point x in space, satisfies the diffusion equation

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left[D(x)\frac{\partial u}{\partial x}\right]$$

The different boundary conditions considered apply to this equation and are the following

Dirichlet $u(x,t) = 0$ for $x \in \partial\Omega$, which means the particles get absorbed by the boundary (or exit) as they reach it.

Neumann $\partial_n u(x,t) = 0$ for $x \in \partial\Omega$, where ∂_n represents the derivative in the outward normal direction, means that there is no outward flux through the boundary. In other words, when a particle hits the boundary it is reflected back inside Ω.

Robin The Robin boundary condition is a combination of the Dirichlet and Neumann conditions, namely $\partial_n u(x,t) = -\lambda u(x,t)$ for $x \in \partial\Omega$. With this boundary condition there is some flux through the boundary.

3.1 First passage time concepts

An important concept in the theory of stochastic processes is that of the first passage time of a certain event. In the case of Brownian motion we usually think of the first time the particle reaches certain target. The probability moments of the first passage time are of interest in many areas of study [22].

In the next sections I will constantly refer to the following two concepts:

Splitting Probability When the absorbing (Dirichlet) boundary consists of disjoint sets $B = \cup_{i=1}^{N} B_i$, $\mathcal{P}_i$ is the probability of exiting through B_i.

Moments of the first passage time $\mathbb{T}_n$ is the n-th moment of the first passage time to the target(s) of interest, that is the APC. When referring to the mean first passage time (MFPT), that is the first moment, we drop the subscript and say simply $\mathbb{T}$.

3.2 Basic problem

On a first approach to the problem we consider Ω, the LN, to be the interior of a sphere of radius R_+, and Ω_1, an APC, to be a concentric sphere of radius $R_- < R_+$. For simplicity we center the spheres at the origin. For this problem, we considered two different conditions for the boundary of Ω. The boundary of the target, $\partial\Omega_1$, is always absorbing, as we regard the process as terminated once the particle has reached Ω_1. The T cell is a point with initial position $x \in \Omega/\Omega_1$.

3.2.1 Neumann boundary

Although in reality T cells are not strictly trapped inside the LN, as I already said, it is also not true that they exit as soon as they hit the boundary of the LN. Therefore, one might be interested in asking, if the T cell was indeed unable to leave the LN, how long will it take it on average, to find its cognate APC.

Mean first passage time

It is well known [22, 23] that the MFPT $\mathbb{T}$ satisfies the Poisson equation

$$\begin{aligned} \nabla^2\mathbb{T}(x) &= -\frac{1}{D}, \quad x \in \Omega/\Omega_1; \\ \mathbb{T}(x) &= 0, \quad x \in \partial\Omega_1; \\ \partial_n\mathbb{T}(x) &= 0, \quad x \in \partial\Omega, \end{aligned} \tag{3.1}$$

where D is the diffusion constant.

Because of radial symmetry $\mathbb{T}$ only depends on r so that the partial differential equation 3.1 reduces to the ordinary differential equation

$$\begin{aligned} \frac{\partial^2\mathbb{T}}{\partial r^2}(r) + \frac{2}{r}\frac{\partial\mathbb{T}}{\partial r}(r) &= -\frac{1}{D}, \quad r \in (R_-, R_+), \\ \mathbb{T}(R_-) &= 0, \\ \partial_n\mathbb{T}(R_+) &= 0. \end{aligned} \tag{3.2}$$

The solution to the homogeneous version of the Poisson equation, that is the Laplace equation, is

$$\frac{c_1}{r} + c_2, \tag{3.3}$$

and we find a particular solution for 3.2 to be $-\frac{r^2}{6D}$. Hence the general solution to 3.2 is

$$\mathbb{T}(r) = \frac{c_1}{r} + c_2 - \frac{r^2}{6D}. \tag{3.4}$$

Now the boundary conditions $\mathbb{T}(R_-) = 0$ and $\partial_n \mathbb{T}(R_+) = 0$ give the following equations for c_1 and c_2

$$\begin{aligned} \frac{c_1}{R_-} + c_2 - \frac{R_-^2}{6D} &= 0, \\ \mathbb{T}'(R_+) = -\frac{c_1}{R_+^2} - \frac{R_+}{3D} &= 0, \end{aligned}$$

from where

$$\begin{aligned} c_1 &= -\frac{R_+^3}{3D}, \\ c_2 &= \frac{R_-^2}{6D} - \frac{c_1}{R_-} = \frac{R_-^2}{6D} + \frac{R_+^3}{3DR_-} = \frac{R_-^3 + 2R_+^3}{6DR_-}. \end{aligned}$$

The final expression for $\mathbb{T}$ is

$$\mathbb{T}(r) = \frac{R_-^3 + 2R_+^3}{6DR_-} - \frac{R_+^3}{3Dr} - \frac{r^2}{6D}. \tag{3.5}$$

A plot of $\mathbb{T}$ is shown on Figure 3.1.

Variance of the mean first passage time

Since the equation for all the moments of the FPT are also well known, it is possible to also obtain an expresion for the Variance of the First Passage Time (VFPT) using that

$$Var = \mathbb{T}_2 - \mathbb{T}^2. \tag{3.6}$$

As in [23], the equation for the second moment is

$$\begin{aligned} \nabla^2 \mathbb{T}_2(x) &= -\frac{2}{D}\mathbb{T}(x), \quad x \in \Omega; \\ \mathbb{T}_2(x) &= 0, \quad x \in \Omega / \Omega_1; \\ \partial_n \mathbb{T}_2(x) &= 0, \quad x \in \partial\Omega. \end{aligned} \tag{3.7}$$

I found the following to be a particular solution for the previous equation:

$$\frac{r^4}{60D^2} + \frac{R_+^3 r}{3D^2} - \frac{\left(R_-^3 + 2R_+^3\right) r^2}{18D^2 R_-}.$$

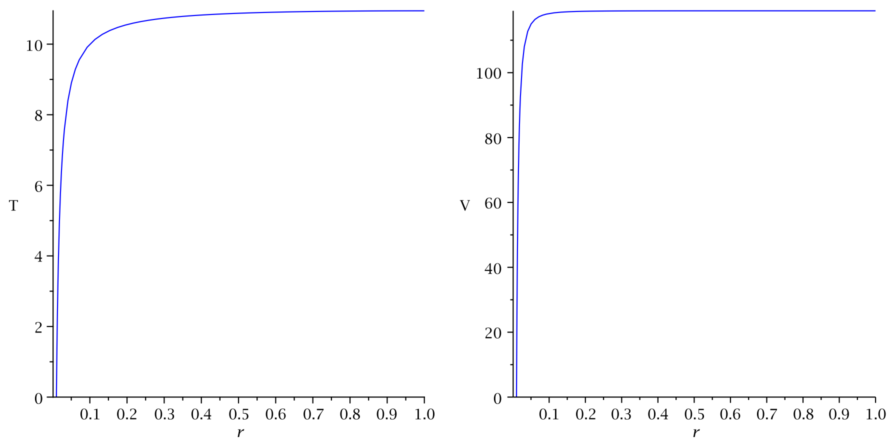


Figure 3.1: Plot of the mean (left) and variance (right) of the FPT to the target inner sphere Ω_1 as a function of the starting point, with reflecting boundary conditions. $R_+ = 1$, $R_- = 0.01$, and $D = 3$.

Hence the solution to 3.7 is

$$\mathbb{T}_2(r) = \frac{c_1}{r} + c_2 + \frac{r^4}{60D^2} + \frac{R_+^3 r}{3D^2} - \frac{\left(R_-^3 + 2R_+^3\right) r^2}{18D^2 R_-}.$$

After substituting the boundary conditions and solving for c_1 and c_2 we find a final expression for the second moment of the FPT

$$\begin{aligned}\mathbb{T}_2(r) = {} & \frac{18R_+^5 R_- - 5R_+^3\left(R_-^3 + 2R_+^3\right)}{45D^2 R_- r} \\ & + \frac{r\left(3R_- r^3 + 60R_+^3 R_- - 10r\left(R_-^3 + 2R_+^3\right)\right)}{180D^2 R_-} \\ & + \frac{40R_+^6 + 7R_-^6 - 72R_- R_+^5 - 20R_-^3 R_+^3}{180D^2 R_-^2},\end{aligned} \tag{3.8}$$

and so we also have an expression for the VFPT, which we denote by $\mathbb{V}$

$$\begin{aligned}\mathbb{V}(r) = {} & \frac{r\left(20R_+^3 - r^3\right)}{90D^2} + \frac{10R_+^6 + R_-^6 - 20R_-^3 R_+^3 - 36R_- R_+^5}{90D^2 R_-^2} \\ & + \frac{2R_+^5}{5D^2 r} - \frac{R_+^6}{9D^2 r^2}.\end{aligned} \tag{3.9}$$

A plot of $\mathbb{V}$ is shown on Figure 3.1.

3.2.2 Dirichlet boundary

Next we study what would happen if T cells would actually exit the LN as soon as they hit its boundary. If we only changed the Neumann boundary conditions to Dirichlet BC on equation 3.1 we would be computing the mean time it takes for the particle to be absorbed, which now can happen on any of the boundaries, i.e. $\partial\Omega$ or $\partial\Omega_1$. However, that is not what we want. We want the MFPT to the APC, that is the mean time it takes for the particle to hit $\partial\Omega_1$ before hitting the LN's boundary $\partial\Omega$. Hence, as in [22], we first need to know the probability of absorption at Ω_1, that is the splitting probability $\mathcal{P}_1$. Because we will only compute the splitting probability to Ω_1 we drop the subscript 1 and use $\mathcal{P}$ instead of $\mathcal{P}_1$ throughout this chapter.

Splitting probability and mean first passage time

According to [22] the splitting probability and the MFPT to Ω_1 satisfy the following equations:

$$\begin{aligned} \nabla^2\mathcal{P}(x) &= 0, \quad x \in \Omega / \Omega_1, \\ \mathcal{P}(x) &= 0, \quad x \in \partial\Omega, \\ \mathcal{P}(x) &= 1, \quad x \in \partial\Omega_1; \end{aligned} \tag{3.10}$$

$$\begin{aligned} D\nabla^2 [\mathcal{P}\mathbb{T}](x) &= -\mathcal{P}(x), \quad x \in \Omega / \Omega_1, \\ [\mathcal{P}\mathbb{T}](x) &= 0, \quad x \in \partial\Omega \cup \partial\Omega_1. \end{aligned} \tag{3.11}$$

Again by spherical symmetry the previous equations are reduced to

$$\begin{aligned} \mathcal{P}''(r) + \frac{2}{r}\mathcal{P}'(r) &= 0, \\ \mathcal{P}(R_-) &= 1, \\ \mathcal{P}(R_+) &= 0; \end{aligned} \tag{3.12}$$

$$\begin{aligned} D\left[\mathcal{P}''\mathbb{T} + 2\mathcal{P}'\mathbb{T}' + \mathcal{P}\mathbb{T}'' + \frac{2}{r}\left(\mathcal{P}'\mathbb{T} + \mathcal{P}\mathbb{T}'\right)\right] &= -\mathcal{P}, \\ \mathcal{P}(R_-)\,\mathbb{T}(R_-) &= 0, \\ \mathcal{P}(R_+)\,\mathbb{T}(R_+) &= 0. \end{aligned} \tag{3.13}$$

We already now the form of the solution to 3.12 (equation 3.3) so we just need to check boundary conditions and get the constants.

$$\begin{aligned}\mathcal{P}(R_-) &= \frac{c_1}{R_-} + c_2 = 1\\ \mathcal{P}(R_+) &= \frac{c_1}{R_+} + c_2 = 0\\ \implies c_2 &= -\frac{c_1}{R_+}\\ \implies R_+c_1 - R_-c_1 &= R_-R_+\\ \implies c_1 &= \frac{R_-R_+}{R_+ - R_-}\\ \implies c_2 &= -\frac{R_-}{R_+ - R_-}\end{aligned}$$

Hence

$$\mathcal{P}(r) = \frac{R_-R_+}{r(R_+ - R_-)} - \frac{R_-}{R_+ - R_-}. \tag{3.14}$$

Equation 3.13 then reduces to

$$\mathbb{T}'' + \frac{2c_2}{c_1 + c_2 r}\mathbb{T}' = -\frac{1}{D} \tag{3.15}$$

and we have that

$$\frac{2c_2}{c_1 + c_2 r} = -\frac{2}{R_+ - r},$$

so the solution is

$$\mathbb{T}(r) = -\frac{r(r - 2R_+)}{6D} - \frac{k_1}{r - R_+} + k_2.$$

Now we need to find k_1 and k_2 from the boundary conditions. We have

$$\begin{aligned}\mathcal{P}(r)\mathbb{T}(r) = {}& \frac{c_2R_+(r - 2R_+)}{6D} - \frac{c_2 r(r - 2R_+)}{6D}\\ &+ \frac{k_1c_2R_+}{r(r - R_+)} - \frac{k_1c_2}{r - R_+} - \frac{k_2c_2R_+}{r} + k_2c_2\end{aligned}$$

where we have written c_1 in terms of c_2 using that $c_1 = -R_+c_2$.

Notice that even though $\mathcal{P}(R_+) = 0$, $\mathcal{P}(R_+)\mathbb{T}(R_+)$ could be nonzero since $\mathbb{T}$ has a $\frac{1}{r-R_+}$ term. We have

$$0 = \mathcal{P}(R_+)\mathbb{T}(R_+) = \frac{k_1c_2(R_+ - R_+)}{R_+(R_+ - R_+)} = -\frac{k_1c_2}{R_+}$$

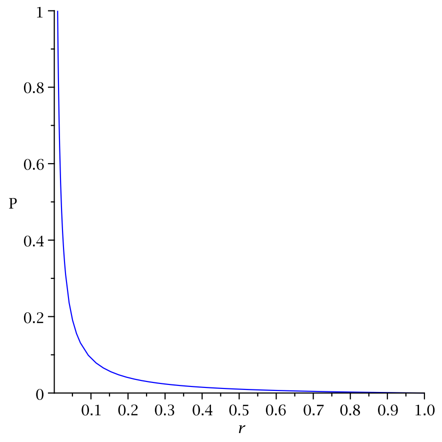


Figure 3.2: Typical shape of the probability of reaching Ω_1, $\mathcal{P}$, as given by equation 3.14. The parameters used are $R_- = 0.01$, $R_+ = 1$, and $D = 3$.

from where we must have $k_1 = 0$. On the other hand since $\mathcal{P}(R_-) = 1$ then to satisfy $\mathcal{P}(R_-)\,\mathbb{T}(R_-) = 0$ we need

$$0 = \mathbb{T}(R_-) = -\frac{R_-(R_- - 2R_+)}{6D} + k_2,$$

and thus

$$k_2 = \frac{R_-(R_- - 2R_+)}{6D}.$$

The final expresion for $\mathbb{T}$ is

$$\mathbb{T}(r) = -\frac{r(r - 2R_+)}{6D} + \frac{R_-(R_- - 2R_+)}{6D}. \tag{3.16}$$

Plots of $\mathcal{P}$ and $\mathbb{T}$ for the case with absorbing boundary are shown on Figures 3.2 and 3.3.

Variance of the first passage time

In [22], the author does not present equations for the higher order moments of the FPT. However, it is easy to deduce the equation for the second moment following the same arguments used to obtain the equation for the MFPT [see 22, Section 1.6.3].

We obtain $\mathcal{P}(x)$ by summing the probabilities for all paths that start at x and reach R_-, i.e.

$$\mathcal{P}(x) = \sum_{p \in paths} P_p(x). \tag{3.17}$$

Then, by their definitions, $\mathbb{T}$ and $\mathbb{T}_2$ can be written as

$$\mathbb{T}\left(x\right) = \frac{\sum_{p\in paths} P_p\left(x\right) t_p\left(x\right)}{\sum_{p\in paths} P_p\left(x\right)}, \tag{3.18}$$

$$\mathbb{T}_2\left(x\right) = \frac{\sum_{p\in paths} P_p\left(x\right) t_p^2\left(x\right)}{\sum_{p\in paths} P_p\left(x\right)}, \tag{3.19}$$

where $t_p\left(x\right)$ represents the time it takes for the particle to go from x to R_- following path p.

Now the sum overall the considered paths can be decomposed into the outcome after one step and the sum over all path remainders from the intermediate point x'. In this discretizing framework we assume that on a time step δt the particle can only take steps of size δx in one of six directions ($\pm e_1$, $\pm e_2$, $\pm e_3$), and each direction is chosen with equal probability. Thus x' can only be equal to $x \pm \delta x e_i$ with $i \in \{1,2,3\}$. Therefore

$$\begin{aligned}\mathcal{P}\left(x\right) &= \sum_p \left[\sum_{i=1}^{3} \frac{1}{6} P_p\left(x+\delta x e_i\right) + \sum_{i=1}^{3} \frac{1}{6} P_p\left(x-\delta x e_i\right)\right] \\ &= \frac{1}{6}\sum_{i=1}^{3}\left[\mathcal{P}_p\left(x+\delta x e_i\right) + \mathcal{P}_p\left(x-\delta x e_i\right)\right].\end{aligned}$$

Upon substituting $\mathcal{P}\left(x \pm \delta x e_i\right)$ by a three term Taylor expansion one obtains equation 3.10. If we repeat what we just did to equation 3.17, to equation 3.19 we obtain

$$\begin{aligned}
\mathcal{P}(x)\,\mathbb{T}_2(x) &= \sum_{p\in paths} P_p(x)\, t_p^2(x) \\
&= \sum_p \sum_{i=1}^{3} \left[\frac{1}{6} P_p(x+\delta x e_i)\left(t_p(x+\delta x e_i)+\delta t\right)^2 \right.\\
&\qquad\qquad \left. + \frac{1}{6} P_p(x-\delta x e_i)\left(t_p(x-\delta x e_i)+\delta t\right)^2\right] \\
&= \delta t^2 \mathcal{P}(x) + \frac{\delta t}{3}\sum_{i=1}^{3}\left[\sum_p P_p(x+\delta x e_i)\, t_p(x+\delta x e_i)\right.\\
&\qquad\qquad \left. + \sum_p P_p(x-\delta x e_i)\, t_p(x-\delta x e_i)\right] \\
&\quad + \frac{1}{6}\sum_{i=1}^{3}\left[\sum_p P_p(x+\delta x e_i)\, t_p^2(x+\delta x e_i)\right.\\
&\qquad\qquad \left. + \sum_p P_p(x-\delta x e_i)\, t_p^2(x-\delta x e_i)\right] \\
&= \delta t^2 \mathcal{P}(x) \\
&\quad + \frac{\delta t}{3}\sum_{i=1}^{3}\left[\mathcal{P}(x+\delta x e_i)\,\mathbb{T}(x+\delta x e_i) + \mathcal{P}(x-\delta x e_i)\,\mathbb{T}(x-\delta x e_i)\right] \\
&\quad + \frac{1}{6}\sum_{i=1}^{3}\left[\mathcal{P}(x+\delta x e_i)\,\mathbb{T}_2(x+\delta x e_i) + \mathcal{P}(x-\delta x e_i)\,\mathbb{T}_2(x-\delta x e_i)\right]
\end{aligned}$$

where the last expression is obtained using equations 3.18 and 3.19. Next we replace $[\mathcal{P}\mathbb{T}_n](x \pm \delta x e_i)$ by its three term Taylor expansion about x. Then

$$\begin{aligned}
\mathcal{P}(x)\,\mathbb{T}_2(x) &= \delta t^2 \mathcal{P}(x) + \frac{\delta t}{3}\sum_{i=1}^{3}\left[2\mathcal{P}(x)\,\mathbb{T}(x) + \delta x^2 \frac{\partial^2 \mathcal{P}\mathbb{T}}{\partial x_i^2}(x)\right] \\
&\quad + \frac{1}{3}\sum_{i=1}^{3}\left[\mathcal{P}(x)\,\mathbb{T}_2(x) + \frac{\delta x^2}{2}\frac{\partial^2 \mathcal{P}\mathbb{T}_2}{\partial x_i^2}(x)\right] \\
&= \delta t^2 \mathcal{P}(x) + 2\delta t \mathcal{P}(x)\,\mathbb{T}(x) + \frac{\delta t \delta x^2}{3}\nabla^2[\mathcal{P}\mathbb{T}](x) \\
&\quad + \mathcal{P}(x)\,\mathbb{T}_2(x) + \frac{\delta x^2}{2}\nabla^2[\mathcal{P}\mathbb{T}_2](x)
\end{aligned}$$

$$\begin{aligned}
&\implies -2\delta t \mathcal{P}(x)\,\mathbb{T}(x) \\
&= \delta t^2 \mathcal{P}(x) + \frac{\delta t \delta x^2}{3}\nabla^2[\mathcal{P}\mathbb{T}](x) + \frac{\delta x^2}{2}\nabla^2[\mathcal{P}\mathbb{T}_2](x)\,.
\end{aligned} \tag{3.20}$$

Now in this discretized diffusion process a relation between the time and space steps is enforced, namely that

$$D = \frac{\delta x^2}{2\delta t}.$$

Thus equation 3.20 turns into

$$\left(\frac{\delta x^2}{2D}\right)^2 \mathcal{P}(x) + \frac{\delta x^4}{6D}\nabla^2 [\mathcal{P}\mathbb{T}](x) + \frac{\delta x^2}{2}\nabla^2 [\mathcal{P}\mathbb{T}_2](x) = -\frac{\delta x^2}{D}\mathcal{P}(x)\,\mathbb{T}(x).$$

Finally, we divide the previous equation by δx^2 and then take the limit as δx goes to 0 to obtain

$$\nabla^2 [\mathcal{P}\mathbb{T}_2](x) = -\frac{2}{D}\mathcal{P}(x)\,\mathbb{T}(x). \tag{3.21}$$

Boundary conditions are analogous to those in 3.13.

Now that we have derived the equation we proceed to solving it. As in equation 3.15 the previous equation reduces to

$$\mathbb{T}_2'' + \frac{2}{r - R_+}\mathbb{T}_2' = \frac{1}{3D^2}\left(r\left(r - 2R_+\right) - R_-\left(R_- - 2R_+\right)\right) \tag{3.22}$$

which has solution given by

$$\begin{aligned}\mathbb{T}_2(r) = {} & \frac{3D^2 b_1 + 5R_+^3 R_-\left(R_- - 2R_+\right) + 2R_+^5}{45D^2\left(r - R_+\right)} \\ & - \frac{\left(R_- - R_+\right)^2\left(r - R_+\right)^2}{9D^2} + \frac{\left(r - R_+\right)^4}{20D^2} + b_2,\end{aligned}$$

where b_1 and b_2 are constants to be determined upon sustitution of boundary conditions. We start by checking the condition imposed on the outer boundary ($r = R_+$), again we must be careful here since $\mathbb{T}_2$ has a term $\frac{1}{r - R_-}$. Cancelling terms where applicable we are left with the following equation at $r = R_+$

$$\begin{aligned}0 &= [\mathcal{P}\mathbb{T}_2](r) \\ &= \frac{R_+\left(3D^2 b_1 + 5R_+^3 R_-\left(R_- - 2R_+\right) + 2R_+^5\right) - r\left(3D^2 b_1 + 5R_+^3 R_-\left(R_- - 2R_+\right) + 2R_+^5\right)}{45D^2 r\left(r - R_+\right)} \\ &= -\frac{\left(r - R_+\right)}{45D^2 r\left(r - R_+\right)}\left(3D^2 b_1 + 5R_+^3 R_-\left(R_- - 2R_+\right) + 2R_+^5\right).\end{aligned}$$

Hence

$$b_1 = -\frac{5R_+^3 R_- \left(R_- - 2R_+\right) + 2R_+^5}{3D^2}.$$

Now, since $\mathcal{P}\left(R_-\right) = 1$ at the other boundary we have

$$\begin{aligned} 0 &= \mathbb{T}_2\left(R_-\right) \\ &= -\frac{\left(R_- - R_+\right)^4}{9D^2} + \frac{\left(R_- - R_+\right)^4}{20D^2} + b_2 \end{aligned}$$

and thus

$$b_2 = \frac{11\left(R_- - R_+\right)^4}{180D^2}.$$

Finally, we arrive to an expression for $\mathbb{T}_2$

$$\mathbb{T}_2\left(r\right) = \frac{\left(11\left(R_- - R_+\right)^2 - 9\left(r - R_+\right)^2\right)\left(\left(R_- - R_+\right)^2 - \left(r - R_+\right)^2\right)}{180D^2}. \tag{3.23}$$

We just need to use this together with formula 3.6 and equation 3.16 to obtain the VFPT for the absorbing case:

$$\begin{aligned} \mathbb{V}\left(r\right) &= \frac{\left(11\left(R_- - R_+\right)^2 - 9\left(r - R_+\right)^2\right)\left(\left(R_- - R_+\right)^2 - \left(r - R_+\right)^2\right)}{180D^2} \\ &\quad - \frac{\left(\left(R_- - R_+\right)^2 - \left(r - R_+\right)^2\right)^2}{36D^2} \\ &= \frac{\left(\left(R_- - R_+\right)^2 - \left(r - R_+\right)^2\right)\left(3\left(R_- - R_+\right)^2 - 2\left(r - R_+\right)^2\right)}{90D^2}. \end{aligned} \tag{3.24}$$

Figure 3.3 shows the VFPT for the problem with Dirichlet boundary conditions.

3.2.3 Robin boundary

Finally, we examine the case of a Robin boundary condition for $\partial\Omega$. It is a good condition to consider since the T cell is not really unable to leave the LN, as in the reflecting case, but it does not necessarily exits the first time it hits the boundary, as in the absorbing case. Therefore, it makes sense to ask what would happen if, on reaching the boundary of the LN, the T cell would exit only with certain probability smaller than 1. With this assumption and following the same discretizing argument that was used

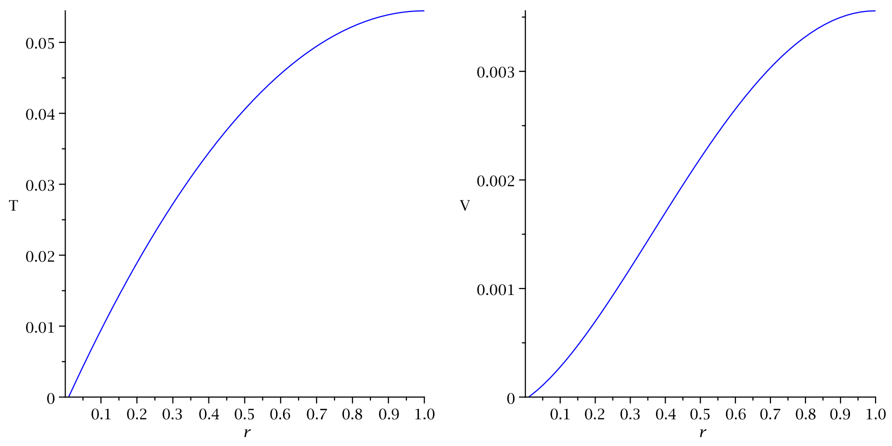


Figure 3.3: Plot of the mean (left) and variance (right) of the FPT to the target inner sphere Ω_1 as a function of the starting point, with absorbing boundary conditions (equations 3.16 and 3.24). The parameters used are $R_- = 0.01$, $R_+ = 1$, and $D = 3$.

on the previous section, we derived a boundary condition for the splitting probability. As it turns out, this is in agreement with assuming a Robin condition for the diffusion equation for the particle concentration u (as in the introduction fo this chapter) [8].

The same equations 3.10, 3.11, and 3.21 are still valid, only the boundary conditions will change. To derive the proper boundary conditions we go back to one dimension, the result is easily extendable to our 3-dimensional case. We consider a particle moving on the interval $[R_-, R_+]$.

Derivation of boundary conditions

Suppose that if the particle reaches R_+ then it either takes a step of size δx back into the interval with probability $1 - \lambda(\delta x)$, or the process ends with probability $\lambda(\delta x)$. In other words, the probability of getting to $R_+ - \delta x$ from R_+ in a time step of size δt, $p(R_+ - \delta x, \delta t | R_+)$, is $1 - \lambda(\delta x)$. Assume also that this probability function λ satisfies $\lambda(\delta x) = \lambda_0 \delta x$, where λ_0 is a constant. Hence $\lambda(\delta x) \to 0$ when $\delta x \to 0$. Then

$$\begin{aligned}\mathcal{P}(R_+) &= (1-\lambda)\,\mathcal{P}(R_+ - \delta x) \\ &\approx (1-\lambda)\left(\mathcal{P}(R_+) - \delta x \mathcal{P}'(R_+)\right),\end{aligned} \tag{3.25}$$

where we have used a Taylor approximation to obtain the second line. Therefore

$$(1-\lambda)\,\delta x\mathcal{P}'\,(R_+) = -\lambda\mathcal{P}\,(R_+)\,.$$

Now we divide by δx and obtain

$$(1-\lambda_0\delta x)\,\mathcal{P}'\,(R_+) = -\lambda_0\mathcal{P}\,(R_+)\,.$$

Taking the limit as δx goes to 0 we get the following condition for the splitting probability on the right hand side boundary:

$$\mathcal{P}'\,(R_+) = -\lambda_0\mathcal{P}\,(R_+)\,. \tag{3.26}$$

Following analogous steps, an equation can be derived for the MFPT to R_-, $\mathbb{T}\,(x)$. From equation 3.18 and by our boundary assumption we have

$$\begin{aligned}[\mathcal{P}\mathbb{T}]\,(R_+) &= \sum_p (1-\lambda)\,P_p\,(R_+-\delta x)\,\big(t_p\,(R_+-\delta x)+\delta t\big)\\ &= \delta t\,(1-\lambda)\,\mathcal{P}\,(R_+-\delta x) + (1-\lambda)\,\mathcal{P}\,(R_+-\delta x)\,\mathbb{T}\,(R_+-\delta x)\\ &= \delta t\mathcal{P}\,(R_+) + (1-\lambda)\,\big([\mathcal{P}\mathbb{T}]\,(R_+) - \delta x\,[\mathcal{P}\mathbb{T}]'\,(R_+)\big)\,,\end{aligned}$$

where we have subsituted equation 3.25 and again used a Taylor approximation of $\mathcal{P}\mathbb{T}$ about R_+ to obtain the last line. This yields

$$(1-\lambda)\,\delta x\,[\mathcal{P}\mathbb{T}]'\,(R_+) = \delta t\mathcal{P}\,(R_+) - \lambda\,[\mathcal{P}\mathbb{T}]\,(R_+)$$

Recall that we have established a relation between δt and δx, namely $\delta t = \delta x^2/2D$. Substituting that and the definition of $\lambda\,(\delta x)$ on the previous equation and then dividing by δx we get

$$(1-\lambda_0\delta x)\,[\mathcal{P}\mathbb{T}]'\,(R_+) = \frac{\delta x}{2D}\mathcal{P}\,(R_+) - \lambda_0\,[\mathcal{P}\mathbb{T}]\,(R_+)\,.$$

Again taking the limit as δx goes to 0 gives the boundary condition we were looking for:

$$[\mathcal{P}\mathbb{T}]'\,(R_+) = -\lambda_0\,[\mathcal{P}\mathbb{T}]\,(R_+)\,. \tag{3.27}$$

The same condition is satisfied by the second moment, as can be seen following the analogous argument, i.e.

$$\begin{aligned}[\mathcal{P}\mathbb{T}_2]\,(R_+) &= \sum_p (1-\lambda)\,P_p\,(R_+-\delta x)\,\big(t_p\,(R_+-\delta x)+\delta t\big)^2\\ &= \delta t^2\mathcal{P}\,(R_+) + 2\delta t\,(1-\lambda)\,[\mathcal{P}\mathbb{T}_2]\,(R_+-\delta x)\\ &\quad + (1-\lambda)\,\big([\mathcal{P}\mathbb{T}_2]\,(R_+) - \delta x\,[\mathcal{P}\mathbb{T}_2]'\,(R_+)\big)\end{aligned}$$

from where

$$[\mathcal{P}\mathbb{T}_2]'\,(R_+) = -\lambda_0\,[\mathcal{P}\mathbb{T}_2]\,(R_+)\,. \tag{3.28}$$

Splitting probability and mean first passage time

Now we are ready to solve the sytem given by equations 3.10, 3.11, and 3.21 changing the boundary conditions on Ω_1 to conditions 3.26, 3.27, and 3.28 respectively, which corresponds to the FPT problem with a partially absorbing boundary. We drop the subscript on parameter λ_0 in the boundary conditions.

We first look at the equation for $\mathcal{P}$. We already know the general solution to 3.10, namely

$$\mathcal{P}(r) = \frac{c_1}{r} + 1 - \frac{c_1}{R_-},$$

Then condition 3.26 gives

$$-\frac{c_1}{R_+^2} + \lambda\frac{c_1}{R_+} + \lambda\left(1 - \frac{c_1}{R_-}\right) = 0, \tag{3.29}$$

which then implies $-c_1R_- + \lambda c_1R_+R_- + \lambda R_+^2R_- - \lambda c_1R_+^2 = 0$ and so

$$c_1 = \frac{\lambda R_+^2R_-}{\lambda R_+^2 - \lambda R_+R_- + R_-}, \tag{3.30}$$

$$c_2 = \frac{R_-\left(1 - \lambda R_+\right)}{\lambda R_+^2 - \lambda R_+R_- + R_-}. \tag{3.31}$$

Hence the solution for $\mathcal{P}$ is

$$\mathcal{P}(r) = \frac{R_-\left(\lambda R_+^2 - \lambda R_+r + r\right)}{r\left(\lambda R_+^2 - \lambda R_+R_- + R_-\right)}. \tag{3.32}$$

See figures 3.4 and 3.5 for a visual representation of this equation.

We can now work the equation for $\mathbb{T}$. Again we know the equation simplifies by spherical symmetry to equation 3.15, which substituting the values of c_1 and c_2 we just computed yields

$$\mathbb{T}'' + \frac{2\left(1 - \lambda R_+\right)}{\lambda R_+^2 - \lambda R_+r + r}\mathbb{T}' = -\frac{1}{D}.$$

The general solution of the previous equation is

$$\mathbb{T}(r) = k_2 - \frac{\left(\lambda R_+^2 - \lambda R_+r + r\right)^2}{6D\left(\lambda R_+ - 1\right)^2} - \frac{k_1}{6D\left(\lambda R_+ - 1\right)\left(\lambda R_+^2 - \lambda R_+r + r\right)}. \tag{3.33}$$

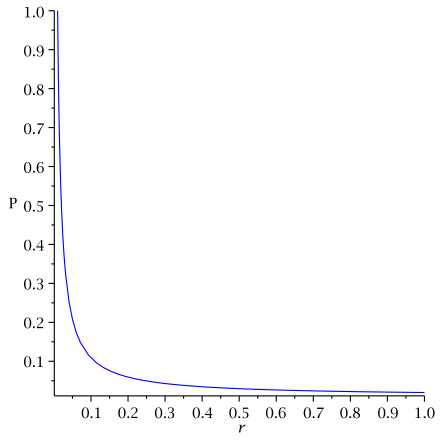


Figure 3.4: Solution for the splitting probability with a Robin BC on $\partial\Omega$. Parameters as before, plus $\lambda = 0.5$.

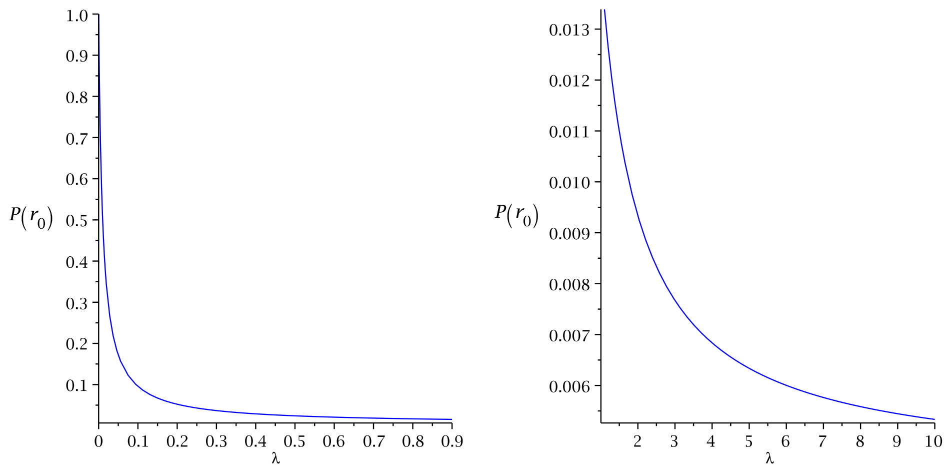


Figure 3.5: Splitting probability $\mathcal{P}$ as a function of the "leakyness" of the boundary, λ, for a fixed value of the starting point, $r_0 = 0.7$. All other parameters remain the same. The spectrum of λ has been split in two plots since $\mathcal{P}$ has a singularity at $\lambda = 1$ for the chosen value of R_+.

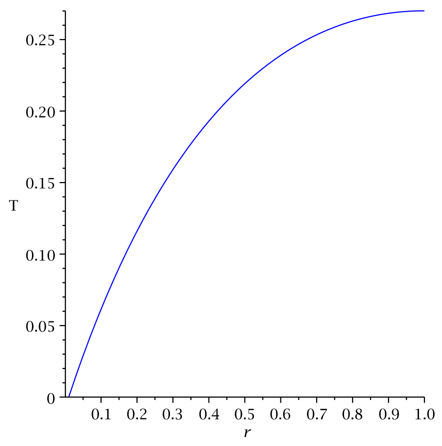


Figure 3.6: Solution for the MFPT with a Robin BC on $\partial\Omega$. Parameters as before, plus $\lambda = 0.5$.

After substituting both boundary conditions, $\mathbb{T}(R_-) = 0$ and $[\mathcal{P}\mathbb{T}]'(R_+) = -\lambda\,[\mathcal{P}\mathbb{T}](R_+)$, we obtain an expression for the MFPT:

$$\mathbb{T}(r) = \frac{1}{3D(\lambda R_+ - 1)}\left[\frac{\left(\lambda R_+^2 - \lambda R_+ R_- + R_-\right)^2 - \left(\lambda R_+^2 - \lambda R_+ r + r\right)^2}{2(\lambda R_+ - 1)} - \frac{R_+^3 (r - R_-)}{\left(\lambda R_+^2 - \lambda R_+ R_- + R_-\right)\left(\lambda R_+^2 - \lambda R_+ r + r\right)}\right]. \tag{3.34}$$

See figures 3.6 and 3.7 for a visual representation of this equation.

3.3 Multiple targets with asymptotically null volume

The real problem we are concerned with is how long it takes for a T cell to find its antigen specific APC, when it is surrounded by other non-specific APCs. The best model would be one where, upon reaching the non-specific APCs, the T cell in question would remain attached to it only for a short time and then it would continue its search in the LN. We start in this section with a model where the T cell stops its search when it hits any of the APCs.

We consider N spherical targets Ω_{ϵ_i} with centers $x_i \in \Omega$ with radii ϵa_i, $a_i > 0$, in such a way that these spheres are completely contained inside

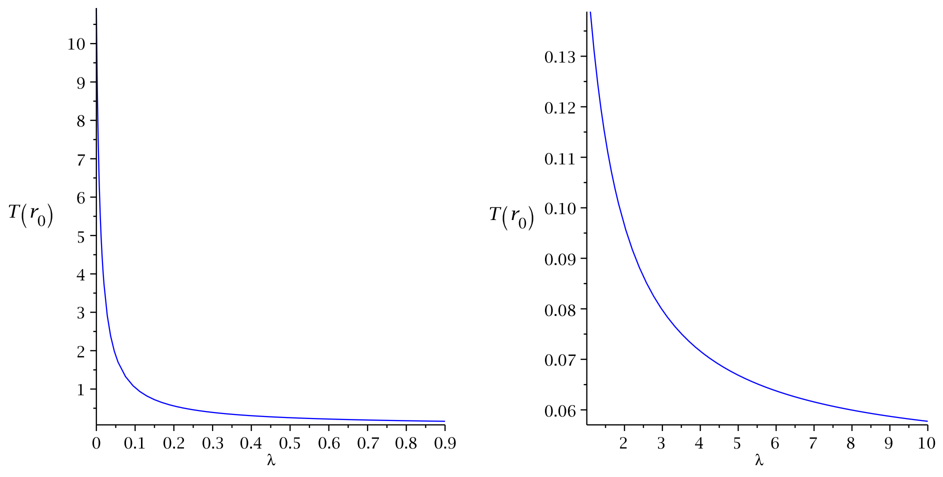


Figure 3.7: MFPT $\mathbb{T}$ as a function of the "leakyness" of the boundary, λ, for a fixed value of the starting point, $r_0 = 0.7$. All other parameters remain the same. The spectrum of λ has been split in two plots since $\mathbb{T}$ has a singularity at $\lambda = 1$ for the chosen value of R_+.

the LN (a sphere Ω of radius a centered at the origin). There are two reasons for making the radii asymptotically small. First, for simplicity, since having randomly located spheres with positive volumes would break the spherical symmetry of Ω that we have been taking a lot of advantage from. By keeping the symmetry we can use a matched asymptotics rather than a simulations approach to solve the problem. The second reason, is that, although APCs tend to be larger than T cells, it makes more sense to make APCs have asymptotically null rather than positive volume, given that the T cell is merely a point.

We again examine different boundary conditions for the LN.

3.3.1 Neumann boundary

This problem has already been studied in the past [3, 4, 15]. In this section I will outline the process of obtaining a two term expansion of the splitting probability $\mathcal{P}_1$, that is the probability of reaching Ω_{ϵ_1} before reaching any of the other APCs Ω_{ϵ_j} for $j = 2, \dots, N$, by using matched asymptotics. The starting point x is in $\Omega \backslash \Omega_p$ where $\Omega_p \equiv \cup_{i=1}^{N} \Omega_{\epsilon_i}$. I followed notes from Professor Michael Ward [25, 26]. For simplicity of notation, we again drop the subscript 1 on the splitting probability and use only $\mathcal{P}(x)$. Later we will use numeric subscripts again for the asymptotic expansions, then $\mathcal{P}_1$ will be used again, but it should not be confused with what we are now simply denoting by $\mathcal{P}$.

Again $\mathcal{P}$ satisfies Laplace's equation

$$\begin{aligned} \nabla^2 \mathcal{P}(x) &= 0, \quad x \in \Omega \backslash \Omega_p; \\ \partial_n \mathcal{P}(x) &= 0, \quad x \in \partial\Omega; \\ \mathcal{P}(x) &= 1, \quad x \in \partial\Omega_1; \\ \mathcal{P}(x) &= 0, \quad x \in \cup_{j=2}^{N} \partial\Omega_{\epsilon_i}. \end{aligned} \tag{3.35}$$

We first look at the outer region, away from the targets. As usual, we expand $\mathcal{P}$ as

$$\mathcal{P} = \mathcal{P}_0 + \epsilon\mathcal{P}_1 + \epsilon^2\mathcal{P}_2 + \ldots \tag{3.36}$$

Then

$$\begin{aligned} \nabla^2 \mathcal{P}_k &= 0, \quad x \in \Omega \backslash \{x_1, \ldots, x_N\} \\ \partial_n \mathcal{P}_k &= 0, \quad x \in \partial\Omega \end{aligned} \tag{3.37}$$

with certain singularity conditions as $x \to x_j$ for $j = 1, \ldots, N$, determined upon matching to the inner solution.

Then we look at the inner regions, we analyze separately the region near each target. Near the j-th APC, we expand the inner solution $w(y) \equiv \mathcal{P}(x_j + \epsilon y)$, with $y \equiv \epsilon^{-1}(x - x_j)$,

$$w = w_0 + \epsilon w_1 + \ldots \tag{3.38}$$

Now $\nabla_y^2 w = \epsilon^2 \nabla_x^2 \mathcal{P}$ and so

$$\nabla_y^2 w_0 = 0, \quad y \notin \Omega_j; \quad w_0 = \delta_{j1}, \quad y \in \partial\Omega_j \tag{3.39}$$

$$\nabla_y^2 w_1 = 0, \quad y \notin \Omega_j; \quad w_1 = 0, \quad y \in \partial\Omega_j \tag{3.40}$$

Here $\Omega_j = \epsilon^{-1}\Omega_{\epsilon_j}$. The far-field boundary conditions for w_0 and w_1 are determined by the matching condition as $x \to x_j$ between the inner and outer expansions 3.36 and 3.38, written as

$$\mathcal{P}_0 + \epsilon\mathcal{P}_1 + \epsilon^2\mathcal{P}_2 + \ldots \sim w_0 + \epsilon w_1 + \ldots \tag{3.41}$$

The first matching condition is that $w_0 \sim \mathcal{P}_0$ as $|y| \to \infty$. As before, the solution to 3.39 is

$$w_0 = \frac{c_1}{|y|} + c_2,$$

so that as $|y| \to \infty$, $w_0 = c_2$ and thus $\mathcal{P}_0 = c_2$ is a constant. The condition $w_0 = \delta_{j1}, \quad y \in \partial\Omega_j$, means $w_0\left(a_j\right) = \delta_{j1}$, and so $c_1 = a_j\left(\delta_{j1} - \mathcal{P}_0\right)$. Hence

$$w_0(y) \approx \mathcal{P}_0 + \frac{a_j\left(\delta_{j1} - \mathcal{P}_0\right)}{|y|} \tag{3.42}$$

as $|y| \to \infty$. From this and the matching condition 3.41 we conclude that as $x \to x_j$,

$$\mathcal{P}_1 \sim \frac{a_j\left(\delta_{j1} - \mathcal{P}_0\right)}{|x - x_j|}. \tag{3.43}$$

To be able to satisfy this condition we slightly perturb equation 3.37:

$$\nabla^2 \mathcal{P}_1(x) = S\delta(x),$$

where S is a constant and δ is the Dirac delta function. Transforming the previous equation to spherical coordinates we have

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left[r^2 \frac{\partial \mathcal{P}_1}{\partial r}\right] = S\delta(r),$$

which upon integration holds

$$\frac{\partial}{\partial r}\left[r^2 \frac{\partial \mathcal{P}_1}{\partial r}\right] = Sr^2\delta(r). \tag{3.44}$$

Now recall that by definition

$$\int_{\mathbb{R}^3} \delta(x)\,dx = 1.$$

We can transform this equation to spherical coordinates to obtain

$$\begin{aligned}
1 &= \int_0^{2\pi}\int_0^{\pi}\int_0^{\infty} \delta(r)\, r^2 \sin\theta\, dr d\theta d\varphi \\
&= \int_0^{\infty} \delta(r)\, r^2 \int_0^{2\pi}\int_0^{\pi} \sin\theta\, d\theta d\varphi dr \\
&= \int_0^{\infty} \delta(r)\, r^2 \int_0^{2\pi} \left[-\cos\theta\right]_0^{\pi} d\varphi dr \\
&= \int_0^{\infty} \delta(r)\, r^2 \int_0^{2\pi} 2 d\varphi dr \\
&= 4\pi \int_0^{\infty} \delta(r)\, r^2 dr.
\end{aligned} \tag{3.45}$$

Hence, using 3.45 on equation 3.44 we get

$$\begin{aligned} r^2\frac{\partial\mathcal{P}_1}{\partial r} &= \frac{S}{4\pi}+c_1' \\ \implies \quad \frac{\partial\mathcal{P}_1}{\partial r} &= \frac{S}{4\pi r^2}+\frac{c_1'}{r^2} \\ \implies \quad \mathcal{P}_1\left(r\right) &= -\frac{S+c_1''}{4\pi r}+c_2'. \end{aligned}$$

Now we can find out what S has to be to agree with the matching condition 3.43, namely

$$S=-4\pi\left(\delta_{j1}-\mathcal{P}_0\right)a_j.$$

Therefore the equation for $\mathcal{P}_1$ writes

$$\begin{aligned} &\nabla^2\mathcal{P}_1=-4\pi\sum_{j=1}^{N}\left(\delta_{j1}-\mathcal{P}_0\right)a_j\delta\left(x-x_j\right), \quad x\in\Omega; \\ &\partial_n\mathcal{P}_1=0, \quad x\in\partial\Omega. \end{aligned} \tag{3.46}$$

Next we will use the divergence theorem to state a solvability condition for $\mathcal{P}_1$ and thus determine $\mathcal{P}_0$. Indeed by the divergence theorem

$$\iiint_\Omega \nabla^2\mathcal{P}_1 dV=\iint_{\partial\Omega}\nabla\mathcal{P}_1\cdot\vec{n}dS=\iint_{\partial\Omega}\partial_n\mathcal{P}_1 dS$$

and the RHS is equal to zero by the boundary condition on 3.46. On the other hand

$$\iiint_\Omega \nabla^2\mathcal{P}_1 dV=-4\pi\sum_{j=1}^{N}\left(\delta_{j1}-\mathcal{P}_0\right)a_j$$

and so

$$\begin{aligned} &\left(1-\mathcal{P}_0\right)a_1-\mathcal{P}_0\textstyle\sum_{j=2}^{N}a_j=0 \\ \implies \quad &\mathcal{P}_0=\frac{a_1}{\sum_{j=1}^{N}a_j}. \end{aligned} \tag{3.47}$$

We write

$$\mathcal{P}_0=\frac{a_1}{N\bar{a}}, \quad \bar{a}\equiv\frac{a_1+\cdots+a_N}{N}, \tag{3.48}$$

to save space.

Now let us look at the Green's function for the system

$$\begin{aligned} \nabla^2 G &= \frac{1}{|\Omega|} - \delta(x-\xi), \quad x \in \Omega; \\ \partial_n G &= 0, \quad x \in \partial\Omega; \\ \int_\Omega G(x;\xi)\,dx &= 0. \end{aligned} \tag{3.49}$$

In terms of this function the solution to 3.46 can be written as

$$\mathcal{P}_1(x) = 4\pi \sum_{j=1}^{N} (\delta_{j1} - \mathcal{P}_0)\, a_j G(x;x_j) + \chi_1 \tag{3.50}$$

where χ_1 is a constant. By integrating the previous equation we obtain an expression for χ_1:

$$\begin{aligned} \int_\Omega \mathcal{P}_1 dx &= 4\pi \sum_{j=1}^{N} (\delta_{j1} - \mathcal{P}_0)\, a_j \int_\Omega G(x;x_j) + \int_\Omega \chi_1 \\ &= \chi_1 |\Omega| \end{aligned}$$

by equation 3.47. Then

$$\chi_1 = \frac{1}{|\Omega|}\int_\Omega \mathcal{P}_1 dx$$

We have finally arrived to a two term expansion of the splitting probability:

$$\mathcal{P}(x) = \frac{a_1}{N\bar{a}} + 4\pi\epsilon a_1 \left[G(x;x_1) - \frac{1}{N\bar{a}} \sum_{j=1}^{N} a_j G(x;x_j) \right] + \epsilon\chi_1 + \mathcal{O}(\epsilon^2). \tag{3.51}$$

The derivation of G and χ_1 can be found in references [15, 25, 26] and I have also rewritten it in Appendix A for the sake of completeness. See Figures 3.9 and 3.10 for some example plots of $\mathcal{P}$.

Mean first passage time

We now look at the MFPT.

The solution to

$$\begin{aligned} \nabla^2 \mathbb{T}_G(x) &= -\frac{1}{D}, \quad x \in \Omega \backslash \Omega_p, \\ \partial_n \mathbb{T}_G(x) &= 0, \quad x \in \partial\Omega, \\ \mathbb{T}_G(x) &= 0, \quad x \in \partial\Omega_p, \end{aligned} \tag{3.52}$$

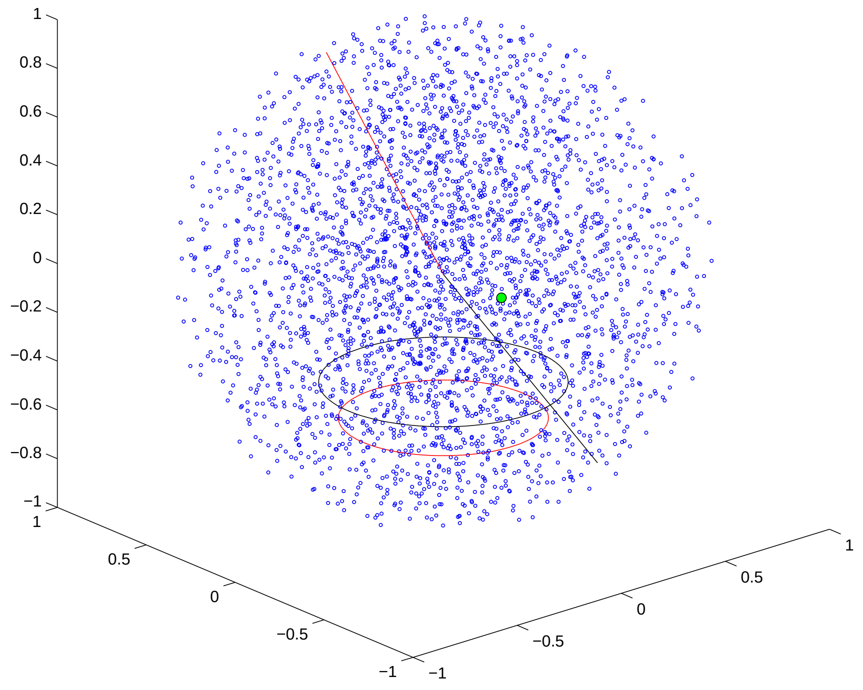


Figure 3.8: Sphere of radius 1 representing the lymph node, with 3000 randomly placed spherical targets (which represent dendritic cells). The one target filled in green is Ω_1, which represents the cognate APC the T cell is looking for. All targets are placed in such a way that there is at least 5ϵ between each pair and also separating them from the LN boundary. Since a spherical LN in mice is about 2mm in diameter [14], the scale here is in millimeters. I used $a_j = 0.35$ for all j, since for $\epsilon = 0.01$ the radius $\epsilon a_j = 3.5 \times 10^{-3}$mm would correspond to the average size of a dendritic cell. However, such a big value for ϵ made it imposible to compute positions for the 3000 targets without overlapping so I used $\epsilon = 0.001$ instead. See the following figures for more details.

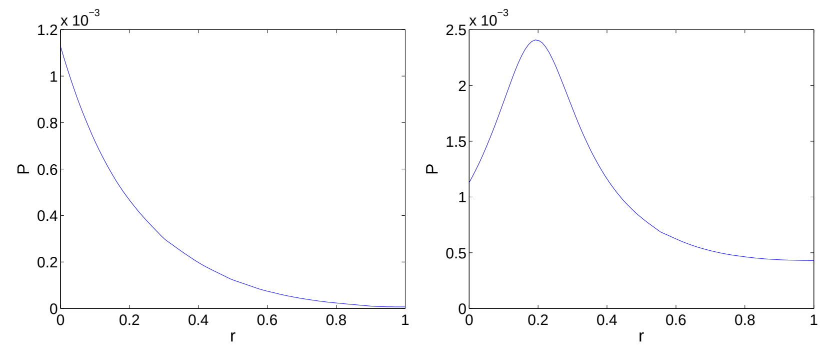


Figure 3.9: Splitting probability from equations 3.51, A.11, and A.15, with $\epsilon = 0.001$. $\mathcal{P}$ is plotted as a function of r along the radius shown in red (left) or black (right) in figure 3.8 (the angles which define these radii were chosen randomly).

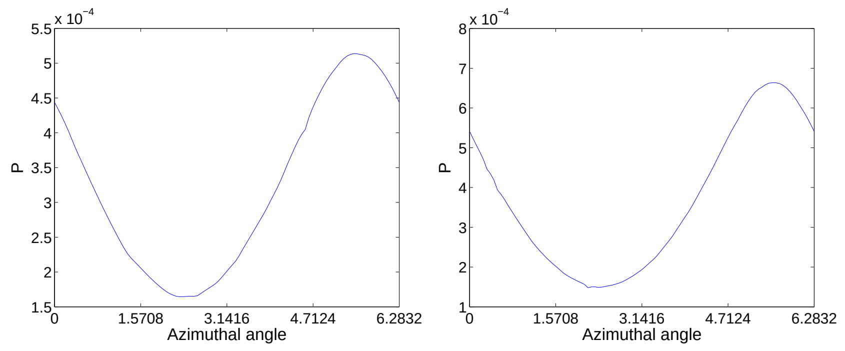


Figure 3.10: Splitting probability from equations 3.51, A.11, and A.15, with $\epsilon = 0.001$. $\mathcal{P}$ is plotted as a function of the azimuthal angle along the circle shown in red (left) or black (right) in figure 3.8 (the radius and the polar angle which define these circles were chosen randomly).

corresponds to the mean first passage time to hitting any of the traps. (The subscript G stands for "general".) This is readily calculated by using the same matched asymptotic approach as before. Thus $\mathbb{T}_G$ is given in the limit $\epsilon \to 0$ of small trap radius by

$$\mathbb{T}_G \approx \frac{a^3}{3N\bar{a}D\epsilon}\left[1 - 4\pi\epsilon \sum_{j=1}^{N} a_j G\left(x; x_j\right) + \frac{4\pi\epsilon}{N\bar{a}} a^T \mathcal{G} a + \mathcal{O}\left(\epsilon^2\right)\right] \quad (3.53)$$

[4]. See Figures 3.11 and 3.12 for examples.

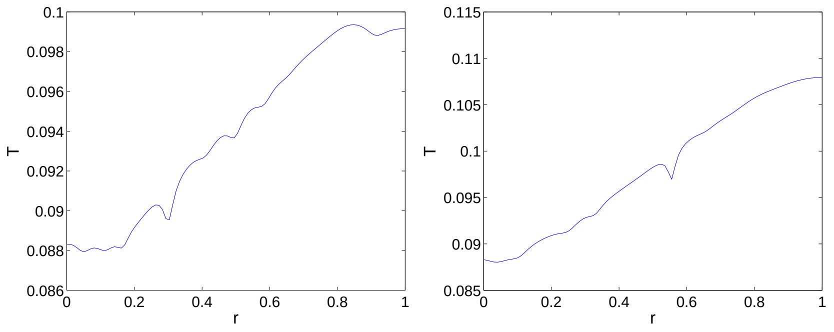


Figure 3.11: MFPT from equations 3.51 and A.11, with ϵ = 0.001 and D = 3. $\mathbb{T}_G$ is plotted as a function of r along the radius shown in red (left) or black (right) in figure 3.8 (the angles which define these radii were chosen randomly).

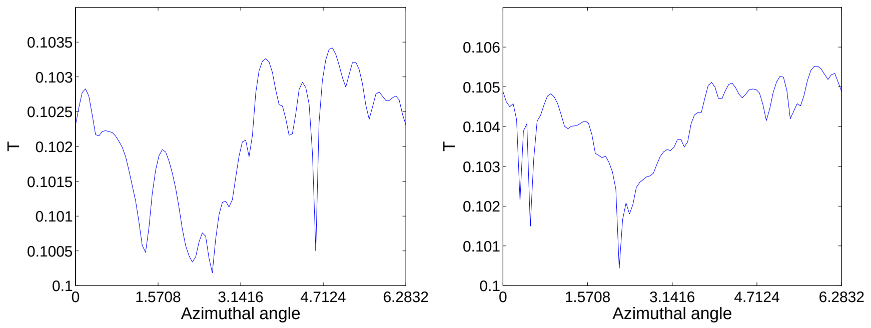


Figure 3.12: MFPT from equations 3.51 and A.11, with ϵ = 0.001 and D = 3. $\mathbb{T}_G$ is plotted as a function of the azimuthal angle along the circle shown in red (left) or black (right) in figure 3.8 (the radius and the polar angle which define these circles were chosen randomly).

Also, the average MFPT based on a uniform distribution of starting points, $\bar{\mathbb{T}}_G = |\Omega|^{-1} \int_\Omega \mathbb{T}_G dx$, satisfies

$$\bar{\mathbb{T}}_G \approx \frac{a^3}{3N\bar{a}D\epsilon}\left[1 + \frac{4\pi\epsilon}{N\bar{a}} a^T \mathcal{G} a + \mathcal{O}\left(\epsilon^2\right)\right]. \tag{3.54}$$

$\bar{\mathbb{T}}_G$ for parameters like those from Figure 3.11 is around 0.1075. Increasing the radii of the targets to 3.5 gives a $\bar{\mathbb{T}}_G$ of about 0.0125, while reducing ϵ to 0.0001 makes $\bar{\mathbb{T}}_G = 1.0885$. See also Figures 3.13 and 3.14.

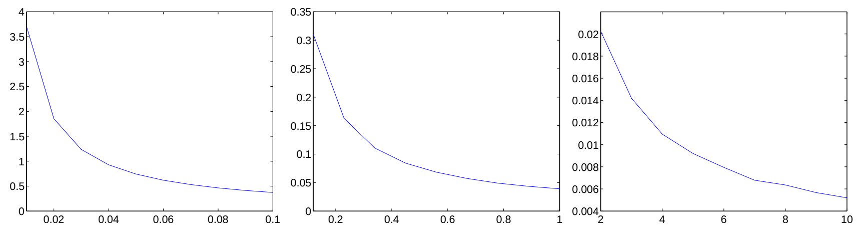


Figure 3.13: Plot of $\bar{\mathbb{T}}_G$ versus a_j. $N = 3000$, $a = 1$, $\epsilon = 0.001$ and $D = 3$. Due to the big variation in scale I separated the range of a_j in three plots.

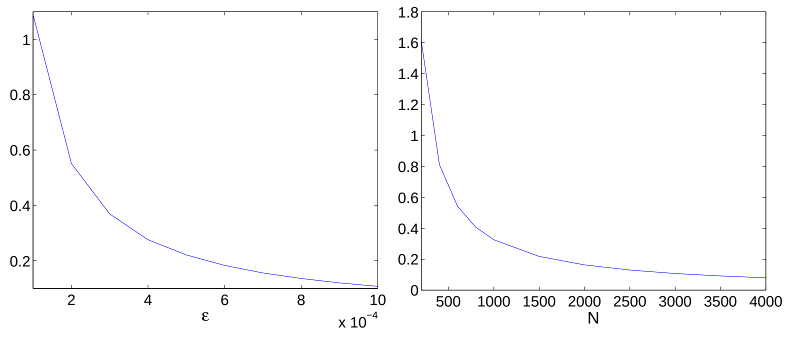


Figure 3.14: Plot of $\bar{\mathbb{T}}_G$ versus ϵ and N. Same parameters as previous figure.

To compute the MFPT to the specific cognate APC, Ω_1, when the T cell is unable to leave the LN we would instead need to solve

$$\begin{aligned} \nabla^2 \left[\mathcal{P}\mathbb{T}\right](x) &= -\frac{1}{D}\mathcal{P}(x), \quad x \in \Omega \backslash \Omega_p; \\ \partial_n \mathbb{T}(x) &= 0, \quad x \in \partial\Omega; \\ \left[\mathcal{P}\mathbb{T}\right](x) &= 0, \quad x \in \partial \cup_{j=2}^{N} \Omega_{\epsilon_j}; \\ \mathbb{T}(x) &= 0, \quad x \in \partial\Omega_{\epsilon_1}. \end{aligned} \tag{3.55}$$

Unfortunately, we have not as yet been able to solve this. Instead, we study the previously mentioned case where the T cell keeps its search after having contact with other non-cognate APCs. This is effectively done by imposing a Neumann boundary condition on all other targets ($j = 2, \dots, N$). Under this model, a contact between the T cell and a non-cognate APC is infinitesimaly short. The differential equation to solve is

$$\begin{aligned} \nabla^2 \mathbb{T}(x) &= -\frac{1}{D}, \quad x \in \Omega \backslash \Omega_p; \\ \partial_n \mathbb{T}(x) &= 0, \quad x \in \partial\Omega \bigcup \cup_{j=2}^{N} \Omega_{\epsilon_j}; \\ \mathbb{T}(x) &= 0, \quad x \in \partial\Omega_{\epsilon_1}. \end{aligned} \tag{3.56}$$

In the outer region we expand $\mathbb{T} = \epsilon^{-1}\mathbb{T}_0 + \mathbb{T}_1 + \epsilon\mathbb{T}_2 + \dots$. For $k = 0, 2$ the $\mathbb{T}_k$ satisfy

$$\begin{aligned} \nabla^2 \mathbb{T}_k(x) &= 0, \quad x \in \Omega \backslash \{x_1, \dots, x_N\}, \\ \partial \mathbb{T}_k(x) &= 0, \quad x \in \partial\Omega, \end{aligned} \tag{3.57}$$

while $\mathbb{T}_1$ satisfies

$$\begin{aligned} \nabla^2 \mathbb{T}_1(x) &= -\frac{1}{D}, \quad x \in \Omega \backslash \{x_1, \dots, x_N\}; \\ \partial_n \mathbb{T}_1(x) &= 0, \quad x \in \partial\Omega. \end{aligned} \tag{3.58}$$

Singularity conditions as $x \to x_j$ will be determined upon matching to the inner solution.

In the inner region near the j-th trap, like we did for $\mathcal{P}$ before, we let $v(y) \equiv \mathbb{T}(x_j + \epsilon y)$ with $y \equiv \epsilon^{-1}(x - x_j)$. We also expand $v = \epsilon^{-1}v_0 + v_1 + \dots$. The equations for the v_k are

$$\begin{aligned} \nabla^2 v_k(y) &= 0, \quad y \notin \Omega_j, \\ \partial_n v_k(y) &= 0, \quad y \in \partial\Omega_j, j \neq 1, \\ v_k(y) &= 0, \quad y \in \partial\Omega_1, \end{aligned} \tag{3.59}$$

and the matching condition reads

$$\epsilon^{-1}\mathbb{T}_0 + \mathbb{T}_1 + \epsilon\mathbb{T}_2 + \dots \sim \epsilon^{-1}v_0 + v_1 + \dots$$

By equations 3.57 and 3.59, $\mathbb{T}_0$ is a constant and v_0 is too in the inner region near the j-th trap for $j \neq 1$. The matching condition implies that $v_0 = \mathbb{T}_0$ in the j-th inner region $j \neq 1$ while

$$v_0(y) = \mathbb{T}_0 - \frac{\mathbb{T}_0 a_1}{|y|}$$

in the inner region near the first trap. By checking the matching condition again, we conclude that $\mathbb{T}_1$ satisfies equation 3.58 with singular behaviour

$$\mathbb{T}_1 \sim -\frac{a_1 \mathbb{T}_0}{|x - x_1|}$$

as $x \to x_1$ and $\mathbb{T}_1 \sim 0$ as $x \to x_j$. Therefore, in terms of the Dirac distribution, $\mathbb{T}_1$ satisfies

$$\begin{aligned} &\nabla^2 \mathbb{T}_1 (x) = 4\pi a_1 \mathbb{T}_0 \delta (x - x_1) - \frac{1}{D}, \quad x \in \Omega; \\ &\partial_n \mathbb{T}_1 (x) = 0, \quad x \in \partial\Omega. \end{aligned}$$

We use the divergence theorem on the previous equation to derive a solvability condition for $\mathbb{T}_1$, which yields

$$\mathbb{T}_0 = \frac{|\Omega|}{4\pi a_1 D}. \tag{3.60}$$

Now in terms of the Green's function, defined by equation 3.49, and a constant χ_T we find that

$$\mathbb{T}_1 = -4\pi a_1 \mathbb{T}_0 G (x; x_1) + \chi_T.$$

Next we expand the previous expression for $\mathbb{T}_1$ as $x \to x_j$ using the local behaviour of G found on Appendix A.1.1. We get

$$\begin{aligned} &\mathbb{T}_1 \sim -\frac{a_1 \mathbb{T}_0}{|x - x_1|} - 4\pi a_1 \mathbb{T}_0 R_{1,1} + \chi_T, \quad \text{as } x \to x_1; \\ &\mathbb{T}_1 \sim -4\pi a_1 \mathbb{T}_0 G_{j,1} + \chi_T, \quad \text{as } x \to x_j. \end{aligned}$$

Upon substituting into the matching condition we find that

$$v_1 \sim -4\pi a_1 \mathbb{T}_0 \mathcal{G}_{j,1} + \chi_T, \quad \text{as } |y| \to \infty,$$

where we are using the matrix notation from Appendix A.1.1. From equation 3.59 we note that v_1 is also a constant in the inner region near the j-th trap for $j \neq 1$, whereas $v_1(y) = c_1 \left(1 - \frac{a_1}{|y|}\right)$ in the inner region near the first trap, with c_1 a constant. Hence

$$v_1 (y) = \left(\chi_T - 4\pi a_1 \mathbb{T}_0 \mathcal{G}_{j,1}\right) \left(1 - \frac{\delta_{j,1} a_1}{|y|}\right).$$

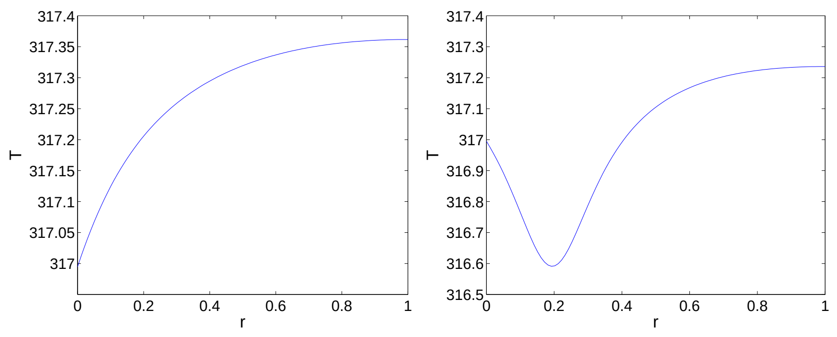


Figure 3.15: MFPT from equation 3.61 with $\epsilon = 0.001$ and $D = 3$. $\mathbb{T}$ is plotted as a function of r along the radius shown in red (left) or black (right) in figure 3.8 (the angles which define these radii were chosen randomly).

Once again we go back to the matching condition and obtain that $\mathbb{T}_2$ should also be approximatedly 0 as $x \to x_j$, $j \neq 1$, and

$$\mathbb{T}_2 \sim \frac{a_1 \left(4\pi a_1 \mathbb{T}_0 R_{1,1} - \chi_T\right)}{|x - x_1|}$$

as $x \to x_1$. Thus the equation for $\mathbb{T}_2$ is

$$\begin{aligned}&\nabla^2 \mathbb{T}_2(x) = -4\pi \left(4\pi a_1 \mathbb{T}_0 R_{1,1} - \chi_T\right) a_1 \delta(x - x_1), \quad x \in \Omega; \\ &\partial_n \mathbb{T}_2(x) = 0, \quad x \in \partial\Omega.\end{aligned}$$

Finally, the divergence theorem yields a solvability condition for $\mathbb{T}_2$ which gives $\chi_T = 4\pi a_1 \mathbb{T}_0 R_{1,1}$.

We have thus obtained a two-term expression for the MFPT:

$$\mathbb{T}(x) = \frac{a^3}{3\epsilon a_1 D} + \frac{4\pi a^3}{3D}\left(R_{1,1} - G(x; x_1)\right). \tag{3.61}$$

See Figures 3.15 and 3.16 for examples.

3.3.2 Robin boundary

In this section we want to study the asymptotic solution to 3.35 when the boundary condition on $\partial\Omega$, $\partial_n \mathcal{P} = 0$, is replaced by $\partial_n \mathcal{P} = \lambda \mathcal{P}$. Since the Robin Green's function for the Laplace equation is unknown, we instead consider the case when $\lambda = \epsilon\lambda_0$ for some constant λ_0 and let ϵ go to 0.

Notice that the solution in the inner region would be the same from the previous section, as only the outer boundary condition has changed.

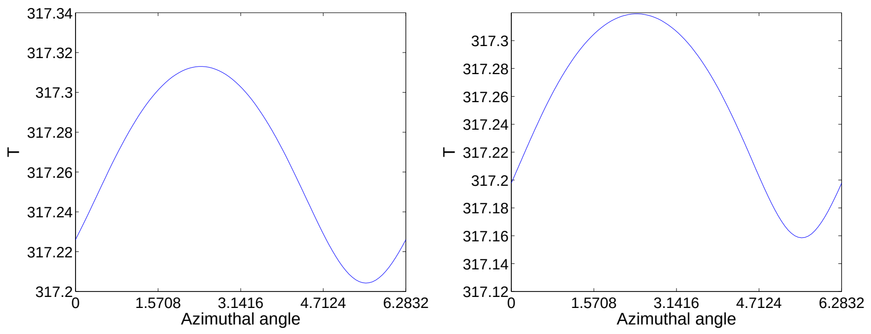


Figure 3.16: MFPT from equation 3.61 with $\epsilon = 0.001$ and $D = 3$. $\mathbb{T}$ is plotted as a function of the azimuthal angle along the circle shown in red (left) or black (right) in figure 3.8 (the radius and the polar angle which define these circles were chosen randomly).

Therefore, we need only solve the outer problem and be careful that the match is not lost.

Like before, in the outer region we expand $\mathcal{P} = \mathcal{P}_0 + \epsilon\mathcal{P}_1 + \epsilon^2\mathcal{P}_2 + \ldots$. Notice that the boundary condition for $\mathcal{P}_0$ is again $\partial_n \mathcal{P}_0 = 0$ so that $\mathcal{P}_0$ is a constant.

Now, near the j-th trap we again let $w(y) \equiv \mathcal{P}\left(x_j + \epsilon y\right)$ and so equations 3.39 and 3.40 hold. From 3.43 we find that $\mathcal{P}_1$ must satisfy

$$\begin{aligned} \nabla^2 \mathcal{P}_1 &= -4\pi \sum_{j=1}^{N} \left(\delta_{j1} - \mathcal{P}_0\right) a_j \delta\left(x - x_j\right), \quad x \in \Omega, \\ \partial_n \mathcal{P}_1 &= -\lambda_0 \mathcal{P}_0, \quad x \in \partial\Omega. \end{aligned} \tag{3.62}$$

The solvability condition for $\mathcal{P}_1$ obtained by the divergence theorem gives

$$\begin{aligned} -4\pi \sum_{j=1}^{N} \left(\delta_{j1} - \mathcal{P}_0\right) a_j &= \iint_{\partial\Omega} \lambda_0 \mathcal{P}_0 \\ &= -\lambda_0 \mathcal{P}_0 |\partial\Omega|, \end{aligned} \tag{3.63}$$

where $|\partial\Omega|$ is the surface area of the sphere Ω. Thus

$$\mathcal{P}_0 = \frac{4\pi a_1}{4\pi N\bar{a} + \lambda_0 |\partial\Omega|}. \tag{3.64}$$

Next we decompose $\mathcal{P}_1 = -\lambda_0 \mathcal{P}_0 P_{1P} + \mathcal{P}_{1H} + \chi_R$ where χ_R is a constant and $\mathcal{P}_{1P}$ and $\mathcal{P}_{1H}$ obey the following conditions

$$\begin{aligned} \nabla^2 \mathcal{P}_{1P} &= \frac{|\partial\Omega|}{|\Omega|}, \quad x \in \Omega, \\ \partial_n \mathcal{P}_{1P} &= 1, \quad x \in \partial\Omega, \\ \int_\Omega \mathcal{P}_{1P}(x)\, dx &= 0; \end{aligned} \tag{3.65}$$

$$\begin{aligned} &\nabla^2 \mathcal{P}_{1H} = \frac{\lambda_0 \mathcal{P}_0 |\partial\Omega|}{|\Omega|} - 4\pi \sum_{j=1}^{N} \left(\delta_{j1} - \mathcal{P}_0\right) a_j \delta\left(x - x_j\right), \quad x \in \Omega, \\ &\partial_n \mathcal{P}_{1H} = 0, \quad x \in \partial\Omega, \\ &\int_\Omega \mathcal{P}_{1H}(x)\, dx = 0. \end{aligned} \tag{3.66}$$

In terms of the Green's function of the previous section (equation 3.49), a solution to 3.66 is readily found to be

$$\mathcal{P}_{1H} = 4\pi \sum_{j=1}^{N} \left(\delta_{j1} - \mathcal{P}_0\right) a_j G\left(x; x_j\right), \tag{3.67}$$

by identity 3.63.

As for $\mathcal{P}_{1P}$, the divergence theorem yields

$$\int_\Omega \left(G(\xi; x)\, \nabla^2 \mathcal{P}_{1P}(\xi) - \mathcal{P}_{1P}(\xi)\, \nabla^2 G(\xi; x)\right) d\xi = \int_{\partial\Omega} G(\xi; x)\, d\xi.$$

The LHS is equal to

$$\frac{\lambda_0 \mathcal{P}_0 |\partial\Omega|}{|\Omega|} \int_\Omega G(\xi; x)\, d\xi - \int_\Omega \mathcal{P}_{1P}(\xi) \left(\frac{1}{|\Omega|} - \delta(\xi - x)\right) d\xi = \mathcal{P}_{1P}(x).$$

Hence

$$\mathcal{P}_{1P}(x) = \int_{\partial\Omega} G(\xi; x)\, d\xi. \tag{3.68}$$

Indeed, by reciprocity of G the third condition for $\mathcal{P}_{1P}$ is satisfied too since

$$\begin{aligned} \int_\Omega \mathcal{P}_{1P}(x)\, dx &= \int_{\partial\Omega} \left(\int_\Omega G(\xi; x)\, dx\right) d\xi \\ &= \int_{\partial\Omega} \left(\int_\Omega G(x; \xi)\, dx\right) d\xi \\ &= 0. \end{aligned}$$

For the specific case we are dealing with, namely Ω is a sphere of radius a, we can also find a explicit solution to 3.65. First notice that

$$\frac{|\partial\Omega|}{|\Omega|} = \frac{4\pi a^2}{\frac{4}{3}\pi a^3} = \frac{3}{a}.$$

Thus the general solution for $\mathcal{P}_{1P}$ is

$$\mathcal{P}_{1P}(r) = \frac{c_1}{r} + c_2 + \frac{r^2}{2a}$$

where c_1 and c_2 are constants. Then the boundary condition yields

$$1 = \mathcal{P}_{1P}'(a) = \left(-\frac{c_1}{r^2} + \frac{r}{a}\right)|_{r=a} = -\frac{c_1}{a^2} + 1$$

so that $c_1 = 0$. On the other hand, the integral condition imposed on $\mathcal{P}_{1P}$ gives

$$\begin{aligned} 0 &= \int_\Omega \mathcal{P}_{1P}(x)\,dx \\ &= 4\pi \int_0^a \left(c_2 + \frac{r^2}{2a}\right) r^2 dr \\ &= 4\pi \left(\frac{c_2 a^3}{3} + \frac{a^5}{10a}\right) \end{aligned}$$

by a change to spherical coordinates. Then

$$c_2 = -\frac{3}{10}a$$

and so

$$\mathcal{P}_{1P}(x) = \frac{|x|^2}{2a} - \frac{3}{10}a. \tag{3.69}$$

We have reached the analogous step to equation 3.51, a two term expansion for the splitting probability:

$$\begin{aligned} \mathcal{P}(x) = \frac{a_1}{N\bar{a} + \lambda_0 a^2} + \epsilon a_1 \Bigg[&-\frac{\lambda_0}{N\bar{a} + \lambda_0 a^2}\left(\frac{|x|^2}{2a} - \frac{3}{10}a\right) + 4\pi G(x; x_1) \\ &- \frac{1}{N\bar{a} + \lambda_0 a^2}\sum_{j=1}^{N} a_j G(x; x_j)\Bigg] + \epsilon\chi_R + \mathcal{O}(\epsilon^2). \end{aligned} \tag{3.70}$$

The derivation of χ_R has been left for Appendix A. See Figures 3.17 and 3.18 for some example plots of $\mathcal{P}$.

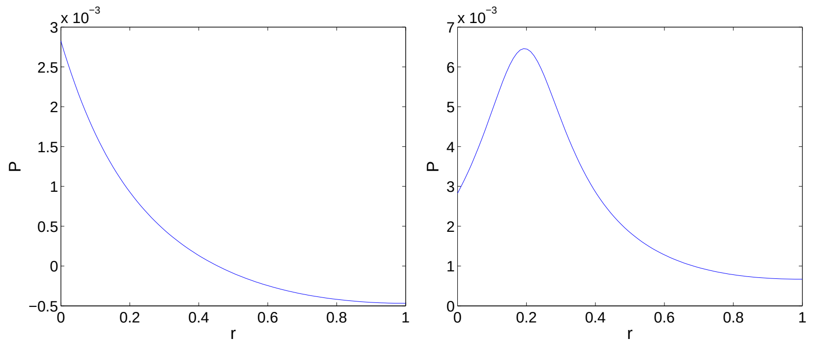


Figure 3.17: Splitting probability from equations 3.70, A.11, and A.19, with $\epsilon = 0.001$ and $\lambda_0 = 0.5$. $\mathcal{P}$ is plotted as a function of r along the radius shown in red (left) or black (right) in Figure 3.8 (the angles which define these radii were chosen randomly).

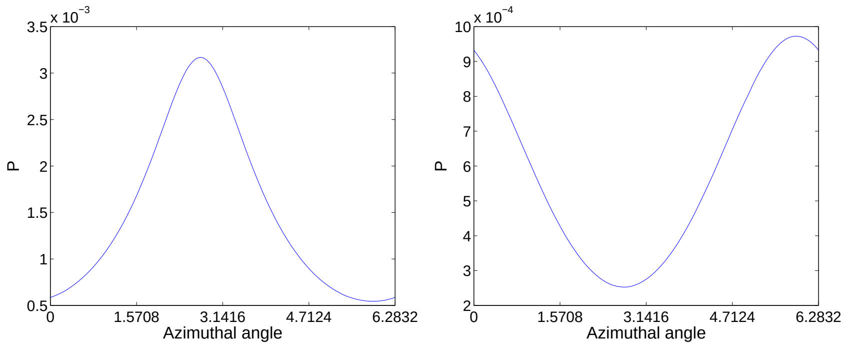


Figure 3.18: Splitting probability from equations 3.70, A.11, and A.19, with $\epsilon = 0.001$ and $\lambda_0 = 0.5$. $\mathcal{P}$ is plotted as a function of the azimuthal angle along the circle shown in red (left) or black (right) in Figure 3.8 (the radius and the polar angle which define these circles were chosen randomly).

An analogous derivation yields the MFPT to any absorbing boundary, that is the solution to equation 3.52 changing the Neumann boundary condition by the Robin boundary condition we proposed here $\partial_n \mathbb{T}_G = \epsilon\lambda_0 \mathbb{T}_G$. Because this general MFPT is not what we are interested in, I did not include that calculation.

3.4 Space dependant diffusivity

We now consider the problem of space dependant diffusion. As it has been said, lymph nodes have a highly organized architecture, and so it makes sense to think that T cells do not perform free diffusion, but that their diffusion coefficient depends on their location inside the LN. Moreover, it has been proposed [1] that the fibroblastic reticular network might be of aid in the search of a T cell for its cognate APC. In the FRN model dendritic cells are located on the network and T cells preferentially crawl along it with their velocities being higher on the FRN [12].

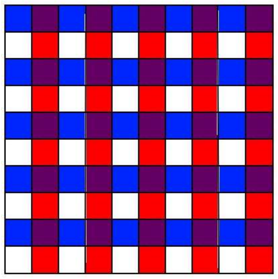


Figure 3.19: Visual representation of a two dimensional version of our FRN model (the FRN fibers shown in color). The diffusion coefficient on the x (respectively y) direction increases when the cell is on the blue (respectively red) layers. The purple squares mark the intersections between horizontal and vertical fibers of the FRN. On those regions diffusion is higher in both the x and y direction. White squares are areas of slower diffusion off the FRN. This image was adapted from [9].

To mimic the influence of the FRN in the time taken for a T cell to find its corresponding APC, we created a mathematical model where the lymph node is layered in all three dimensions. Diffusion on the x direction (respectively y, z directions) is higher when the cell is on the strips with y and z constant (respectively x and z constant, and x and y constant). (See Figure 3.19 for a visual representation of this model.)

We write the already familiar equation for the MFPT when there are several absorbing sections on the boundary, in the following way

$$L^{\epsilon}\left[\mathbb{T}^{\epsilon}\mathcal{P}\right] = \mathcal{P}\left(x\right), \tag{3.71}$$

with $\mathcal{P}$ satisfying the corresponding equation from the previous sections. Here ϵ represents the width of the layers.

Before we had $L = -D\nabla^2$. This time each entry of the Laplacian will be multiplied by a different diffusion coefficient, which will also depend on x.

We want to find $\mathbb{T}^{\epsilon}$ in the homogenization limit $\epsilon \to 0$, i.e. when the strips are so thin that every region, no matter how small, intersects both types of layers. Notice that L^{ϵ} can be written as $L^{\epsilon} = -A\left(x/\epsilon\right) : \mathcal{D}^2$ where $\mathcal{D}^2$ is the matrix of second partial derivatives, i.e. with i, j entry equal to

$$\left(\mathcal{D}^2\right)_{i,j} = \frac{\partial^2}{\partial x_i x_j},$$

and the colon operator acts in the following way

$$M : N = tr\left(M^T N\right) = \sum_{i,j=1}^{3} M_{i,j}N_{i,j}$$

for any matrices M and N.

To start with a simpler example, we take a step back to the case where there is only one target. Then the MFPT (to any absorbing part of the boundary) satisfies

$$L^{\epsilon}\mathbb{T}^{\epsilon} = 1 \tag{3.72}$$

[16], where

$$L^{\epsilon} = -\sum_{i,j=1}^{3} A_{i,j}\left(\frac{x}{\epsilon}\right)\frac{\partial^2}{\partial x_i x_j}, \tag{3.73}$$

A being the matrix of diffusion coefficients and ϵ the width of the strips.

According to [9], in the limit when $\epsilon \to 0$ solutions of the general equation $L^{\epsilon}u^{\epsilon} = f\left(x\right)$ converge to solutions of the averaged equation

$$-\bar{A} : \mathcal{D}^2 u = f\left(x\right) \tag{3.74}$$

Here the averaged coefficient matrix $\bar{A}$ is given by $\bar{A} = \langle\rho^{\infty}A\rangle$ where the angle brackets denote the average $\langle v\rangle = \int v$, and ρ is the invariant distribution of L, that is the solution to

$$L^{*}\rho^{\infty} = 0,$$
$$\langle\rho^{\infty}\rangle = 1.$$

Notice that the matrix A will be periodic since we want the diffusion coefficient to depend only on whether the particle is on or off the color strips and not of on which particular strip it is currently found. For the previous result to hold we need A to have period 1, so we will enforce this when defining the diffusion coefficents on the layered pattern. Thus to compute the average $\langle v \rangle$ we need only integrate over $[0, 1]$.

To define A, we first consider a symmetric function $g(x)$ such that g takes higher values on the interval $[0, 0.5]$, which would correspond to one of the FRN fibers, and lower values on the interval $[0.5, 1]$. One very simple example is shown in Figure 3.20.

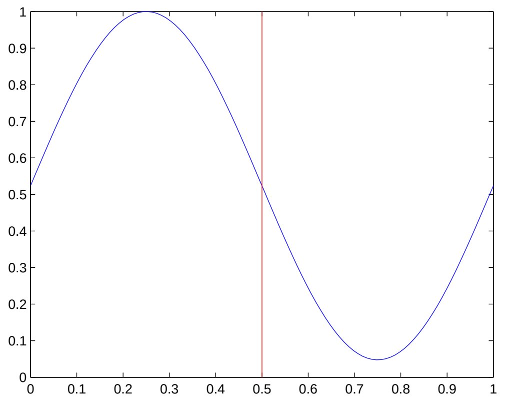


Figure 3.20: Plot of $g(z) = (\sin(2\pi z) + 1.1)/2.1$. The red line is the border line between layers like in Figure 3.19, the left side of the red line corresponding to a color strip.

In terms of such a function g we could define the diffusion coefficient matrix to be

$$A(x) = D \begin{pmatrix} g(x_2)g(x_3 - 0.5) & 0 & 0 \\ 0 & g(x_1 - 0.5)g(x_3 - 0.5) & 0 \\ 0 & 0 & g(x_1 - 0.5)g(x_2) \end{pmatrix},$$

where $x = (x_1, x_2, x_3)$ and D is a basal diffusion coefficient.

Clearly A is diagonal and has separable entries, that is it can be expressed as a product of diagonal matrices each depending on only one x_i:

$$A(x) = D \prod_{j=1}^{n} A^j(x_j).$$

In the particular case we are considering

$$A^1(x_1) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & g(x_1 - 0.5) & 0 \\ 0 & 0 & g(x_1 - 0.5) \end{pmatrix},$$

$$A^2(x_2) = \begin{pmatrix} g(x_2) & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & g(x_2) \end{pmatrix},$$

$$A^3(x_3) = \begin{pmatrix} g(x_3 - 0.5) & 0 & 0 \\ 0 & g(x_3 - 0.5) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Hence according to [9] the matrix of coefficients for the averaged equation has the simplified form

$$\begin{aligned} \bar{A} &= \left\langle \frac{1}{D\prod_{j=1}^{n} A_{jj}^{j}(x_j)} A \right\rangle \left\langle \frac{1}{D\prod_{j=1}^{n} A_{jj}^{j}(x_j)} \right\rangle^{-1} \\ &= \langle A \rangle . \end{aligned} \tag{3.75}$$

If we use g as in Figure 3.20 then

$$h = \int_0^1 g(z)\, dz = \int_0^1 g(z - 0.5)\, dz = \frac{1.1}{2.1}$$

so that the averaged equation ends up being

$$-Dh^2 \nabla^2 u = f(x) \tag{3.76}$$

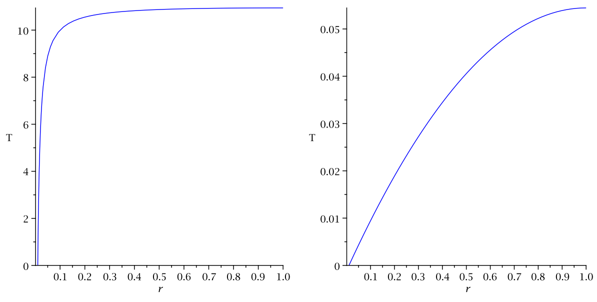
Notice that the case will be the same whenever the functions that describe the behaviour along each fiber integrate to the same constant, provided that A is diagonal and has separable entries.

Another interesting case to consider would be to only have these fast fibers along two of the three directions. Then we would have a different diffusion coefficient along one direction, but all coefficients would still be constant.

More complicated dynamics arise if diffusion changes from its basal state in every direction regardless of the orientation of the fiber the cell is at. Although these alternative model is plausible, the simple model presented here already incorporates the influence of the FRN so that there is no need to complicate it. Indeed, by increasing the diffusion coefficient of the same direction as the orientation of the fiber the cell is at, it would appear that it is crawling along the fiber.

3.5 Discussion and future work

Notice how the MFPT increases 10 fold from a Robin boundary (with $\lambda = 0.5$) and 100 fold from a Dirichlet boundary, to a Neumann boundary (see Figure 3.21). Also, in the case where the cell is unable to leave the LN, the MFPT increases very rapidly as the starting position of the T cell moves away from the target. In the other two cases, that is when the cell has some chance to escape from the LN, the increase is less steep. Recall that, in reality, T cells remain in the LN only for some time and if they do not get activated they recirculate through the body and return to the LN later.



(a) Neumann (b) Dirichlet (c) Robin

Figure 3.21: Mean first passage time with different conditions on the boundary of the LN $\partial\Omega$. Figures 3.1, 3.3 and 3.6 are here shown side by side for ease of comparison.

For the case where we considered multiple asymptotically small targets, notice how changing the Neumann for a Robin BC introduces a difference in the splitting probability already at first order. This is so despite the fact that the exit flux out of the LN imposed by the Robin condition is of order ϵ. (Compare equations 3.51 and 3.70.) Indeed, $\mathcal{P}_0$ for the Robin condition has an extra $\lambda_0 a^2$ in the denominator so that the probability of hitting Ω_1 before the process ends (by absorption by either the other targets or the LN boundary) is smaller in the Robin case.

To our knowledge, the splitting probability had never been computed for the multiple asymptotically small tagets with a Robin BC on the outer sphere. That is an important contribution from this thesis work, even if the case we considered is a Robin BC of order ϵ. Moreover, further extensions of this condition can be adapted from the results in [8]. For example, conditions could be derived for a model where the moving particle waits for some time on the boundary before deciding whether to exit or to return to the inside. Such a model might be relevant for other FPT biological problems, if not for the T cell activation problem here proposed.

Most results in this chapter can be improved to resemble more closely the dynamics of T cell activation. Many calculations were left in a sense incomplete, for example on Section 3.2.3 I did not compute the variance. Just the calculation of the second moment is very lengthy, complicated and error prone, which is why I did not continue to the variance. In the previous section one could also find explicit solutions for the case when the averaged diffusion coefficients are not equal in all directions.

Another interesting variation of the problems presented here would be to include more information about the antigen presenting cells. For example, since dendritic cells are a major class of APCs one could study how having long dendrites changes the time it takes for a T cell to find its corresponding DC.

Also all the models presented could be used with real data to estimate contact rates and other kinetic parameters.

As I said in the introduction, this mean first passage time approach to the immunological problem was inspired by the Graw-Regoes papers [11, 12]. Hence it is of high interest to compare our results with those presented in the paper. However, this work is not yet in a stage where such comparison can be done. On the 2009 paper, the authors studied the influence of target localization on the killing rates of cytotoxic T cells. Although we considered a different process, namely localization of the activating APC, the problems are very similar and many of the results here presented could be adapted to the other problem. The problem is, however, that our model ends when the target is found. To evaluate the simulated results from the paper under our theoretical framework, we would need to consider a model in which the bond remains for a certain time and after breakage the T cell goes back to the searching stage. I mentioned before that, given the serial engagement hypothesis, this model would also be interesting under the T cell activation problem.

Following the FRN model, one could also consider the MFPT to a specific node on a graph (the FRN) when the T cell is only moving along the edges of the graph. This has been studied in the past via simulations [7]. However, a MFPT calculation on a random graph (or other type of graph which might resemble the FRN) is very complicated.

In summary, there is a lot more work that can be done and many directions in which this research can be extended. Moreover, I believe the work here presented is significative for it sets the theoretical framework which can not only be used for the immunological problem I proposed, but for many other biological problems.

Bibliography

[1] Marc Bajénoff, Nicolas Glaichenhaus, and Ronald N. Germain. Fibroblastic reticular cells guide T lymphocyte entry and migration within the splenic T cell zone. *Journal of Immunology*, 181(6), September 2008.

[2] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems*. SIAM, 2009.

[3] Claire Chevalier, Olivier Bénichou, Bob Meyer, and Raphaël Voituriez. First-passage quantities of brownian motion in a bounded domain with multiple targets: a unified approach. *Journal of Physics A, Mathematical and Theoretical*, 44(2), December 2011.

[4] Alexei F. Cheviakov and Michael J. Ward. Optimizing the principal eigenvalue of the laplacian in a sphere with interior traps. *Mathematical and Computer Modeling*, 53(7-8), March 2010.

[5] Toby Collins. ordfilt3: Performs 3D order-statistic filtering on 3D volumetric data. From `http://www.mathworks.com/matlabcentral/fileexchange/22044-ordfilt3`, 2008.

[6] Winfried Denk, James H. Strickler, and Watt W. Webb. Two-photon laser scanning fluorescence microscopy. *Science*, 248(4951), April 1990.

[7] Graham M. Donovan and Grant Lythe. T cell movement on the reticular network. *Journal of Theoritical Biology*, 295, February 2012.

[8] William Feller. Diffusion processes in one dimension. *Transactions of the American Mathematical Society*, 77(1), May 1954.

[9] Brittany D. Froese and Adam M. Oberman. Numerical averaging of non-divergence structure elliptic operators. *Communications in Mathematical Sciences*, 7(4), August 2009.

[10] Izrail' M. Gradshteyn and Iosif M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, 1980.

[11] Frederik Graw and Roland R. Regoes. Investigating CTL mediated killing with a 3D cellular automaton. *PLoS Computational Biology*, 5(8), August 2009.

[12] Frederik Graw and Roland R. Regoes. Influence of the fibroblastic reticular network on cell-cell interactions in lymphoid organs. *PLoS Computational Biology*, 8(3), March 2012.

[13] Khuloud Jaqaman, Dinah Loerke, Marcel Mettlen, Hirotaka Kuwata, Sergio Grinstein, Sandra L. Schmid, and Gaudenz Danuser. Robust single-particle tracking in live-cell time-lapse sequences. *Nature Methods*, 5(8), 2008. Available at `http://lccb.hms.harvard.edu/software.html`.

[14] Yoshitsugu Kawashima, Makoto Sugimura, Yann-Ching Hwang, and Norio Kudo. The lymph system in mice. *The Japanese Journal of Veterinary Research*, 12(4), July 1964.

[15] Theodore Kolokolnikov, Michèle S. Titcombe, and Michael J. Ward. Optimizing the fundamental Neumann eigenvalue for the Laplacian in a domain with small traps. *European Journal of Applied Mathematics*, 16(2), April 2005.

[16] Hannah W. McKenzie, Mark A. Lewis, and Evelyn H. Merrill. First passage time analysis of animal movement insights into the functional response. *Bulletin of Mathematical Biology*, 71(1), September 2009.

[17] MicroscopyU. Fundamentals and applications in multiphoton excitation microscopy. From `http://www.microscopyu.com/articles/fluorescence/multiphoton/multiphotonintro.html`.

[18] Jennifer S. Morrison. Deciphering multi-state mobility within single particle trajectories of proteins on the plasma membrane. Master's thesis, The University of British Columbia, August 2010.

[19] Kenneth Murphy. *Janeway's Immunobiology*. Garland Science, 8th edition, 2012.

[20] National Institutes of Health. Image J. From `http://rsbweb.nih.gov/ij/`.

[21] Jean-Christophe Olivo-Marin. Extraction of spots in biological images using multiscale products. *Pattern Recognition*, 35(9), 2002.

[22] Sidney Redner. *A Guide to First passage processes.* Cambridge University Press, 2001.

[23] Hannes Risken. *The Fokker-Planck Equation. Methods of Solution and Applications.* Springer-Verlag, second edition, 1989.

[24] Ulrich H. von Andrian and Thorsten R. Mempel. Homing and cellular traffic in lymph nodes. *Nature Reviews Immunology*, 3:867–878, November 2003.

[25] Michael J. Ward. Math 551: Homework 2. From `http://www.math.ubc.ca/~ward/teaching/m551/hw2_11sol.pdf`.

[26] Michael J. Ward. Notes on strong localized perturbation theory. From `http://www.math.ubc.ca/~ward/teaching/m551/m550_hole.pdf`.

[27] Carla Wofsy, Daniel Coombs, and Byron Goldstein. Calculations show substantial serial engagement of T cell receptors. *Biophysical Journal*, 80(2), February 2001.

Appendix A

Additional derivations

A.1 From the Neumann splitting probability, equation 3.51

A.1.1 The Neumann Green's function for the sphere

We now calculate the Neumann Green's function $G(x, \xi)$ satisfying equation 3.49 when Ω is a sphere of radius a.

First let us look at the singular part of G, that is we want to solve

$$\nabla^2 u(x) = -\delta(x - \xi), \quad \text{in } \mathbb{R}^3. \tag{A.1}$$

Take a small sphere Ω_ϵ of radius ϵ about ξ. Then in a neighborhood of ξ we let $r = |x - \xi|$ and then

$$\begin{aligned} 0 &= \nabla^2 u \\ &= u_{rr} + \frac{2}{r} u_r, \quad \text{for } r > 0, \end{aligned}$$

so that $u(r) = \frac{\beta}{r}$.

Now, since

$$\int_{\Omega_\epsilon} \nabla^2 u(x)\, dx = -\int_{\Omega_\epsilon} \delta(x - \xi)\, dx = -1,$$

then by the divergence theorem

$$-1 = \int_{\Omega_\epsilon} \nabla u \cdot \vec{n} dS = 4\pi \left(r^2 u_r(r) \right) |_{r=\epsilon}, \tag{A.2}$$

again by a change to spherical coordinates as in equation 3.45. Since $u_r = -\beta r^{-2}$ then equation A.2 gives

$$4\pi \left(-\frac{\epsilon^2 \beta}{\epsilon^2} \right) = -1$$

which yields

$$\beta = \frac{1}{4\pi}.$$

Thus

$$u\left(x\right) = \frac{1}{4\pi|x-\xi|},$$

and so we can write

$$G\left(x;\xi\right) \approx \frac{1}{4\pi|x-\xi|} + R\left(x;\xi\right), \quad \text{as } x \to \xi. \tag{A.3}$$

To find R I followed the Appendix A from reference [15]. We first decompose G as follows

$$G\left(x;\xi\right) = u\left(x\right) + \frac{|x|^2+|\xi|^2}{6|\Omega|} + \frac{1}{4\pi}\phi\left(x;\xi\right), \tag{A.4}$$

with ϕ an unknown function we want to compute. Notice that in that case

$$\begin{aligned}\nabla^2 G\left(x;\xi\right) &= \nabla^2 u\left(x\right) + \frac{1}{6|\Omega|}\left(\frac{\partial^2}{\partial r^2} + \frac{2}{r}\frac{\partial}{\partial}\right)\left(|x|^2+|\xi|^2\right) + \frac{1}{4\pi}\nabla^2\phi\left(x;\xi\right)\\ &= -\delta\left(x-\xi\right) + \frac{1}{|\Omega|} + \frac{1}{4\pi}\nabla^2\phi\left(x;\xi\right).\end{aligned}$$

Because Ω is a sphere, $|\Omega| = 4\pi a^3/3$. Then

$$\partial_n G\left(x;\xi\right) = \frac{1}{4\pi}\left(\partial_n\left(\frac{1}{|x-\xi|}\right) + \frac{|x|}{a^3} + \partial_n\phi\right).$$

Substituting this onto the differential equation which defines G, equation 3.49, we find that ϕ must satisfy

$$\begin{aligned}&\nabla^2\phi = 0, \quad x \in \Omega,\\ &\partial_n\phi = -\frac{1}{a^2} - \partial_n\left(\frac{1}{|x-\xi|}\right), \quad x \in \partial\Omega.\end{aligned} \tag{A.5}$$

To solve A.5 we choose a coordinate system so that the source point $x = \xi$ is on the positive z axis. Then, since $\nabla^2\phi = 0$ and ϕ is axisymmetric, ϕ admits the series expansion

$$\phi\left(x;\xi\right) = \sum_{n=0}^{\infty} B_n P_n\left(\cos\theta\right)\left(\frac{|x||\xi|}{a^2}\right)^n, \tag{A.6}$$

where P_n is the Legendre polynomial of integer n and the B_n for $n = 0, 1, \ldots$ are coefficients to be determined. To simplify the problem, we have enforced that $\phi\left(x;\xi\right) = \phi\left(\xi;x\right)$ so that $G\left(x;\xi\right) = G\left(\xi;x\right)$, an assumption that agrees with the interpretation of $\mathcal{P}$. From A.6 we see that

$$\partial_n\phi|_{\partial\Omega}= \sum_{n=0}^{\infty} \frac{nB_n}{a^{n+1}} P_n\left(\cos\theta\right)|\xi|^n. \tag{A.7}$$

Now, from the generating function for Legendre polynomials,

$$\frac{1}{\sqrt{1-2tz+t^2}} = \sum_{k=0}^{\infty} t^k P_k\left(z\right), \tag{A.8}$$

we obtain the following equation, by setting $z = \cos\theta$ and $t = |\xi|/|x|$:

$$\begin{aligned} \frac{1}{|x-\xi|} &= \frac{1}{\sqrt{|\xi|^2 - 2|x||\xi|\cos\theta + |x|^2}} \\ &= \frac{1}{|x|\sqrt{t^2-2tz+1}} \\ &= \sum_{n=0}^{\infty} \frac{|\xi|^n}{|x|^{n+1}} P_n\left(\cos\theta\right). \end{aligned}$$

By differentiating this expression with respect to $r = |x|$ we get

$$\partial_r\left(\frac{1}{|x-\xi|}\right)|_{r=a} = -\sum_{n=0}^{\infty} \frac{\left(n+1\right)|\xi|^n}{a^{n+2}} P_n\left(\cos\theta\right). \tag{A.9}$$

Upon substituting A.7 and A.9 into the boundary condition given in A.5, we obtain

$$\sum_{n=0}^{\infty}\left(nB_n - \frac{n+1}{a}\right) P_n\left(\cos\theta\right)\frac{|\xi|^n}{a^{n+1}} = -\frac{1}{a^2}.$$

Since B_0 is arbitrary, we can choose it to be equal to $1/a$ for convenience. Then the other coefficients must satisfy

$$B_n = \frac{1}{a} + \frac{1}{na}, \quad \text{for } n \geq 1. \tag{A.10}$$

Back to the series expansion of ϕ, equation A.6, and replacing on it the previous expressions for the B_n's, we find that

$$\phi\left(x;\xi\right) = \frac{1}{a}\sum_{n=0}^{\infty} P_n\left(\cos\theta\right)\left(\frac{|x||\xi|}{a^2}\right)^n + \frac{1}{a}\sum_{n=1}^{\infty}\frac{1}{n}P_n\left(\cos\theta\right)\left(\frac{|x||\xi|}{a^2}\right)^n.$$

By using the generating function A.8 again we find that the first infinite sum reduces to

$$\begin{aligned}\frac{1}{a}\sum_{n=0}^{\infty} P_n(\cos\theta)\left(\frac{|x||\xi|}{a^2}\right)^n &= \frac{a}{|x|}\left(\frac{a^4}{|x|^2} - 2|\xi|\frac{a^2}{|x|}\cos\theta + |\xi|^2\right)^{-\frac{1}{2}} \\ &= \frac{a}{|x|r'},\end{aligned}$$

where $x' = xa^2/|x|^2$ is the image point to x outside the sphere and $r' = |x' - \xi|$.

To calculate the second infinite sum let us define the following function:

$$I(\beta) = \sum_{n=1}^{\infty} \frac{1}{n} P_n(\cos\theta)\,\beta^n.$$

In terms of it, the second infinite sum is equal to $a^{-1} I\left(|x||\xi|/a^2\right)$. Upon diferentiating and then using the generating function A.8 we get

$$\begin{aligned} I'(\beta) &= \sum_{n=1}^{\infty} P_n(\cos\theta)\,\beta^{n-1} \\ &= \frac{1}{\beta}\sum_{n=1}^{\infty} P_n(\cos\theta)\,\beta^n \\ &= \frac{1}{\beta}\left(\sum_{n=0}^{\infty} \beta^n P_n(\cos\theta) - P_0(\cos\theta)\right) \\ &= \frac{1}{\beta}\left(\frac{1}{\sqrt{\beta^2 - 2\beta\cos\theta + 1}} - 1\right),\end{aligned}$$

since $P_0(\cos\theta) = 1$. We can integrate the previous equation [see 10, page 95] to obtain an expression for I. Since $I(0) = 0$,

$$\begin{aligned} I(\beta) &= \int_0^{\beta} \frac{1}{s}\left(\frac{1}{\sqrt{s^2 - 2s\cos\theta + 1}} - 1\right) ds \\ &= \log\left(\frac{2}{1 - \beta\cos\theta + \sqrt{1 + \beta^2 - 2\beta\cos\theta}}\right).\end{aligned}$$

We are finally in a position to obtain a final expression for ϕ and so also for G.

$$\phi(x;\xi) = \frac{a}{|x|r'} + \frac{1}{a}\log\left(\frac{2a^2}{a^2 - |x||\xi|\cos\theta + |x|r'}\right),$$

$$
\begin{aligned}
G(x;\xi) = {} & \frac{1}{4\pi|x-\xi|} + \frac{|x|^2+|\xi|^2}{6|\Omega|} + \frac{a}{4\pi|x|r'} \\
& + \frac{1}{4\pi a}\log\left(\frac{2a^2}{a^2-|x||\xi|\cos\theta+|x|r'}\right) + B,
\end{aligned}
\tag{A.11}
$$

where B is a constant which will be chosen to satisfy $\int_\Omega G = 0$.

Now we want to show that $\int_\Omega G(x;\xi)$ is independent of ξ and so that we can compute B by looking at the simpler equation $\int_\Omega G(x;0) = 0$. We have:

$$
\begin{aligned}
0 &= \int_\Omega G(x;\xi')\left[\nabla^2 G(x;\xi) - \frac{1}{|\Omega|} + \delta(x-\xi)\right]dx \\
&= \int_\Omega G(x;\xi')\,\nabla^2 G(x;\xi)\,dx - \frac{1}{|\Omega|}\int_\Omega G(x;\xi') + G(\xi;\xi') \\
&= \int_\Omega \nabla\cdot\left[G(x;\xi')\,\nabla G(x;\xi)\right]dx - \int_\Omega \left[\nabla G(x;\xi)\cdot\nabla G(x;\xi')\right]dx \\
&\quad - \frac{1}{|\Omega|}\int_\Omega G(x;\xi') + G(\xi;\xi').
\end{aligned}
$$

Then the divergence theorem gives

$$
\begin{aligned}
\int_\Omega \nabla\cdot\left[G(x;\xi')\,\nabla G(x;\xi)\right]dx &= \int_{\partial\Omega} G(x;\xi')\,\nabla G(x;\xi)\cdot\vec{n}dx \\
&= 0,
\end{aligned}
$$

by the boundary condition on G, i.e. $\partial_n G = 0$ on $\partial\Omega$. Hence

$$
\frac{1}{|\Omega|}\int_\Omega G(x;\xi') = G(\xi;\xi') - \int_\Omega \nabla G(x;\xi)\cdot\nabla G(x;\xi')\,dx.
$$

Note that the RHS is symmetric in ξ and ξ'. It follows that $\int_\Omega G(x;\xi)\,dx = \int_\Omega G(x;\xi')\,dx$. With a change to spherical coordinates, like in equation 3.45, we compute the desired integral:

$$
\begin{aligned}
0 &= \int_\Omega G(x;0)\,dx \\
&= \frac{1}{4\pi}\int_0^{2\pi}\int_0^{\pi}\int_0^{\infty} r\sin\theta dr d\theta d\varphi + \frac{1}{4\pi a}\int_0^{2\pi}\int_0^{\pi}\int_0^{\infty} r^2\sin\theta dr d\theta d\varphi \\
&\quad + \frac{1}{6|\Omega|}\int_0^{2\pi}\int_0^{\pi}\int_0^{\infty} r^4\sin\theta dr d\theta d\varphi + B|\Omega| \\
&= \int_0^a r dr + \frac{1}{a}\int_0^a r^2 dr + \frac{4\pi}{6|\Omega|}\int_0^a r^4 dr + B|\Omega| \\
&= a^2\left(\frac{1}{2}+\frac{1}{3}+\frac{1}{10}\right) + \frac{4\pi a^3}{3}B.
\end{aligned}
$$

Here we have used the fact that when $\xi = 0$, $r' = |x'| = a^2/|x|$. Thus

$$B = -\frac{7}{10\pi a}. \tag{A.12}$$

A.1.2 Derivation of χ_1

In this section we compute the constant from the order ϵ function, $\mathcal{P}_1$, from the splitting probability with multiple targets and Neumann outer boundary.

Let us first write $\mathcal{P}_1(x)$ as $x \to x_j$ using what we found on the previous section (equation A.3).

$$\mathcal{P}_1(x) \approx \frac{(1-\mathcal{P}_0)\,a_1}{|x-x_1|} + 4\pi(1-\mathcal{P}_0)\,a_1R_{1,1} - 4\pi\mathcal{P}_0\sum_{i=2}^{N} a_iG_{1,i} + \chi_1, \quad \text{as } x \to x_1,$$

$$\mathcal{P}_1(x) \approx -\frac{\mathcal{P}_0a_j}{|x-x_j|} - 4\pi\mathcal{P}_0a_jR_{j,j} + 4\pi a_1G_{j,1} - 4\pi\mathcal{P}_0\sum_{i=1,i\neq j}^{N} a_iG_{j,i} + \chi_1,$$
$$\text{as } x \to x_j,\ j = 2,\dots,N,$$

where $G_{i,j} \equiv G(x_i;x_j)$ and $R_{i,j} \equiv R(x_i;x_j)$. We reduce the previous equations by introducing the notation

$$B_1 = 4\pi a_1R_{1,1} - 4\pi\mathcal{P}_0\left(a_1R_{1,1} + \sum_{i=2}^{N} a_iG_{1,i}\right),$$
$$B_j = 4\pi a_1G_{j,1} - 4\pi\mathcal{P}_0\left(a_jR_{j,j} + \sum_{i=1,i\neq j}^{N} a_iG_{j,i}\right), \quad j = 2,\dots,N.$$

Thus

$$\mathcal{P}_1(x) \approx \begin{cases} \dfrac{(1-\mathcal{P}_0)\,a_1}{|x-x_1|} + B_1 + \chi_1, & \text{as } x \to x_1 \\ -\dfrac{\mathcal{P}_0a_j}{|x-x_1|} + B_j + \chi_1, & \text{as } x \to x_j \end{cases}. \tag{A.13}$$

From the matching conditions 3.41 and 3.42 we get that w_1 must satisfy $w_1 \approx B_j + \chi_1$ as $|y| \to \infty$. On the other hand, equation 3.40 has solution

$$w_1(y) = c\left(1 - \frac{a_j}{|y|}\right),$$

where c is a constant which is readily found to be $c = B_j + \chi_1$, since $w_1 \to c$ as $|y| \to \infty$.

Going back again to the matching condition 3.41, it yields

$$\mathcal{P}_2 \approx -\frac{a_j\left(B_j + \chi_1\right)}{|x - x_j|}, \quad \text{as } x \to x_j,\, j = 1, \ldots, N.$$

By an argument like that led by equations 3.44 and 3.45 we find that $\mathcal{P}_2$ should satisfy

$$\nabla^2 \mathcal{P}_2 = 4\pi \sum_{j=1}^{N} a_j \left(B_j + \chi_1\right) \delta\left(x - x_j\right), \quad x \in \Omega, \tag{A.14}$$
$$\partial_n \mathcal{P}_2 = 0, \quad x \in \partial\Omega.$$

The divergence theorem yields a solvability condition for $\mathcal{P}_2$, just as it did for $\mathcal{P}_1$ on equation 3.47, namely

$$\sum_{j=1}^{N} a_j \left(B_j + \chi_1\right) = 0.$$

Hence

$$\chi_1 = -\frac{1}{N\bar{a}} \sum_{j=1}^{N} a_j B_j.$$

The expressions for B_j can be substituted back and so χ_1 can be written in matrix form as

$$\chi_1 = -\frac{4\pi a_1}{N\bar{a}} \left[\left(\mathcal{G}^T a\right)_1 - \frac{1}{N\bar{a}} a^T \mathcal{G} a \right], \tag{A.15}$$

where $a = \left(a_1, \ldots, a_N\right)^T$ and

$$\mathcal{G} \equiv \begin{pmatrix} R_{1,1} & G_{1,2} & \cdots & G_{1,N} \\ G_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & G_{N-1,N} \\ G_{N,1} & \cdots & G_{N,N-1} & R_{N,N} \end{pmatrix}.$$

Notice that with equations A.11 and A.12 we can compute all the entries of this matrix. In particular

$$R_{i,i} = \frac{|x_i|^2}{4\pi a^3} + \frac{a}{4\pi\left(a^2 - |x_i|^2\right)} + \frac{1}{4\pi a} \log\left(\frac{a^2}{a^2 - |x_i|^2}\right) - \frac{7}{10\pi a}. \tag{A.16}$$

A.2 From the Robin splitting probability, equation 3.70

A.2.1 Derivation of χ_R

In this section we compute the constant from the order ϵ function, $\mathcal{P}_1$, from the splitting probability with multiple targets and Robin outer boundary.

First we look at the limit as $x \to x_j$ of $\mathcal{P}_1(x)$. Like in Section A.1.2 we write

$$\mathcal{P}_1(x) \approx \begin{cases} \dfrac{(1-\mathcal{P}_0)\,a_1}{|x-x_1|} + B_1 + \chi_R, & \text{as } x \to x_1 \\ -\dfrac{\mathcal{P}_0 a_j}{|x-x_1|} + B_j + \chi_R, & \text{as } x \to x_j \end{cases} \tag{A.17}$$

with

$$B_1 = -\lambda_0 \mathcal{P}_0 \mathcal{P}_{1P}(x_1) + 4\pi a_1 R_{1,1} - 4\pi \mathcal{P}_0 \left(a_1 R_{1,1} + \sum_{i=2}^{N} a_i G_{1,i} \right),$$

$$B_j = -\lambda_0 \mathcal{P}_0 \mathcal{P}_{1P}\left(x_j\right) + 4\pi a_1 G_{j,1} - 4\pi \mathcal{P}_0 \left(a_j R_{j,j} + \sum_{i=1, i\neq j}^{N} a_i G_{j,i} \right), \quad j = 2, \dots, N.$$

Following an argument entirely analogous to that on A.1.2 we find that $\mathcal{P}_2$ must satisfy

$$\begin{aligned} \nabla^2 \mathcal{P}_2 &= 4\pi \sum_{j=1}^{N} a_j \left(B_j + \chi_R\right) \delta\left(x - x_j\right), \quad x \in \Omega, \\ \partial_n \mathcal{P}_2 &= -\lambda_0 \mathcal{P}_1, \quad x \in \partial\Omega. \end{aligned} \tag{A.18}$$

Upon using the divergence theorem on the previous equation we obtain a solvability condition for $\mathcal{P}_2$, namely

$$\begin{aligned} 4\pi \sum_{j=1}^{N} a_j \left(B_j + \chi_R\right) &= \int_\Omega \nabla^2 \mathcal{P}_2 dV \\ &= -\lambda_0 \int_{\partial\Omega} \mathcal{P}_1 dS \\ &= \lambda_0^2 \mathcal{P}_0 \int_{\partial\Omega} \mathcal{P}_{1P} dS \\ &\quad - \lambda_0 4\pi \sum_{j=1}^{N} \left(\delta_{j1} - \mathcal{P}_0\right) a_j \mathcal{P}_{1P}\left(x_j\right) - \lambda_0 \chi_R |\partial\Omega| \end{aligned}$$

by equation 3.68.

Now notice that $\mathcal{P}_{1P}$ is constant on $\partial\Omega$ since it is a function of only $|x|$. Its value is $a/2 - 3a/10 = a/5$. Hence

$$\int_{\partial\Omega} \mathcal{P}_{1P} dS = \frac{a|\partial\Omega|}{5} = \frac{4\pi a^3}{5}.$$

Substituting back the expressions for B_j we find that $\sum_{j=1}^{N} a_j B_j$ can be written with matrix notation as

$$\sum_{j=1}^{N} a_j B_j = -\lambda_o \mathcal{P}_0 \sum_{j=1}^{N} a_j \mathcal{P}_{1P}\left(x_j\right) + 4\pi \left[a_1 \left(\mathcal{G}^T a\right)_1 - \mathcal{P}_0 a^T \mathcal{G} a\right].$$

Recall that

$$\mathcal{P}_{1P}\left(x\right) = \frac{|x|^2}{2a} - \frac{3}{10}a.$$

Therefore

$$\begin{aligned}
\chi_R &= \frac{1}{4\pi\left(N\bar{a} + \lambda_0 a^2\right)} \left(\lambda_0^2 \mathcal{P}_0 \int_{\partial\Omega} \mathcal{P}_{1P} dS - \lambda_0 4\pi \sum_{j=1}^{N} \left(\delta_{j1} - \mathcal{P}_0\right) a_j \mathcal{P}_{1P}\left(x_j\right) \right. \\
&\qquad \left. - 4\pi \sum_{j=1}^{N} a_j B_j \right) \\
&= \frac{1}{N\bar{a} + \lambda_0 a^2} \left(\frac{1}{5}\lambda_0^2 \mathcal{P}_0 a^3 - \lambda_0 a_1 \mathcal{P}_{1P}\left(x_1\right) + \lambda_0 \mathcal{P}_0 \sum_{j=1}^{N} a_j \mathcal{P}_{1P}\left(x_j\right) \right. \\
&\qquad \left. + \lambda_o \mathcal{P}_0 \sum_{j=1}^{N} a_j \mathcal{P}_{1P}\left(x_j\right) - 4\pi \left[a_1 \left(\mathcal{G}^T a\right)_1 - \mathcal{P}_0 a^T \mathcal{G} a\right]\right) \\
&= \frac{1}{N\bar{a} + \lambda_0 a^2} \left(\frac{\lambda_0^2 \mathcal{P}_0 a^3}{5} - \frac{\lambda_0 a_1 |x_1|^2}{2a} + \frac{3\lambda_0 a_1 a}{10} + \frac{\lambda_0 \mathcal{P}_0}{a} \sum_{j=1}^{N} a_j |x_j|^2 \right. \\
&\qquad \left. - \frac{3\lambda_0 \mathcal{P}_0 a N\bar{a}}{5} - 4\pi \left[a_1 \left(\mathcal{G}^T a\right)_1 - \mathcal{P}_0 a^T \mathcal{G} a\right]\right).
\end{aligned} \tag{A.19}$$

Appendix B

Programs

In this appendix I present the most relevant codes written by me which I used for Chapter 2.

B.1 spotDetector3D

Extension to three dimensions of the program *spotDetector*, algorithm from [21], coded by François Aguet, which can be downloaded from http://lccb.hms.harvard.edu/software.html.

```
function movieInfo=spotDetector3D(path,Z,T,choice,dthreshold)
% Adapted for 3D from SpotDetector, http://lccb.hms.harvard.edu/doc/spotDetector_101410.zip
%
% Reads a denoised 3D video in the form of a collection of 2D images
% Detects centers of 3D cells in each snapshot
% Centers are determined in the following way:
% - First, find all 3D connected regions (considering only nearest
% neighbors).
% - Get the centers of mass of these regions.
% - Find local maxima (using a cubic window of side 7).
% - For each region, retain
%       - the first local maximum if choice=0
%       - the center of mass if choice=1
%   and the secondary local maxima that are farther away from both the
%   center of mass and the highest local maximum than dthreshold.
%
% INPUT     path is the path to the folder where the images are contained
%           the code assumes that such folder follows the pattern created by
%           imstotiff.m, i.e.
%            - such folder has the name of the data set, with which also the images
%              names start
%            - inside this folder, there's another one named "processed" where all the
%              denoised images are contained
%            - the images are named name_Tddd_Zdd.tif
%              Images are also assumed to be squared
%
%            Z is the number of planes from a 3D "snapshot"
%            T is the total number of time points imaged
%
% OUTPUT is movieInfo, appropiately formatted to be input to u-track
% For a movie with N frames, movieInfo is a structure array with N entries.
% Every entry has the fields xCoord, yCoord, zCoord (if 3D) and amp.
% If there are M objects in frame i, each one of these fields in
% moveiInfo(i) will be an Mx2 array, where the first column is the value
% (e.g. x-coordinate in xCoord and amplitude in amp) and the second column
% is all zero.
%
% In SpotDetector3D, flag is unused. Uncomment the block starting with
% "if flag==0", to split also regions where cells "look like" 3D 8's.

if nargin<5
    dthreshold = 5;
end
if nargin<4
    choice=0;
elseif isempty(choice)
    choice=0;
end
```

```
movieInfo=repmat(struct('xCoord',[],'yCoord',[],'zCoord',[],'amp',[]),T,1);
[~,base]=fileparts(path);
L=size(imread([path,filesep,'processed',filesep,base,'_T001_Z01.tif']),1);
img3d=zeros(L,L,Z); %img3d will contain the 3D image
ny=L;
nx=L;
nz=Z;

for t=1:T
    % save 2D images into 3D compund
    tname=['_T',num2str(t,'%3.3d')];
    for z=1:Z
        sufix=['_Z',num2str(z,'%2.2d'),'.tif'];
        name=[path,filesep,'processed',filesep,base,tname,sufix];
        img3d(:,:,z)=imread(name);
    end


%% 3D process connected components

    mask3d=img3d>0;
    localMax = locmax3d(img3d, 7);
    [labels, nComp] = bwlabeln(mask3d, 6);

    area = zeros(nComp, 1);
```

```
totalInt = zeros(nComp, 1);
nMaxima = zeros(nComp, 1);
xmax = zeros(nComp, 1);
ymax = zeros(nComp, 1);
zmax = zeros(nComp, 1);
xcom = zeros(nComp, 1);
ycom = zeros(nComp, 1);
zcom = zeros(nComp, 1);
labelVect = zeros(nComp, 1);

xmax2 = cell(nComp, 1);
ymax2 = cell(nComp, 1);
zmax2 = cell(nComp, 1);
area2 = cell(nComp, 1);
totalInt2 = cell(nComp, 1);
labelVect2 = cell(nComp, 1);

% Compute area and center of mass for each component
stats = regionprops(labels, img3d, 'Area', 'WeightedCentroid', 'PixelIdxList');

% Component labels of local maxima
maxLabels = labels .* (labels & localMax>0);
maxCoords(1:nComp) = struct('PixelIdxList', []);
mc = regionprops(maxLabels, 'PixelIdxList');
maxCoords(1:length(mc)) = deal(mc);
```

```
for n = 1:nComp
    [yi,xi, zi] = ind2sub([ny nx nz], stats(n).PixelIdxList);
    [ym,xm,zm] = ind2sub([ny nx nz], maxCoords(n).PixelIdxList);
    area(n) = stats(n).Area;
    com = stats(n).WeightedCentroid;
    xcom(n) = com(1);
    ycom(n) = com(2);
    zcom(n) = com(3);

    values = img3d(stats(n).PixelIdxList);
    totalInt(n) = sum(values);

    nMaxima(n) = length(xm);
    if nMaxima(n)==1
        xmax(n) = xm;
        ymax(n) = ym;
        zmax(n) = zm;
        nMaxima(n) = 1;
        labelVect(n) = labels(ym,xm,zm);
    elseif nMaxima(n)==0 % no maximum was detected for this cluster
        maxValueIdx = find(values == max(values));
        xmax(n) = xi(maxValueIdx(1));
        ymax(n) = yi(maxValueIdx(1));
        zmax(n) = zi(maxValueIdx(1));
        nMaxima(n) = 1;
    labelVect(n) = labels(ymax(n), xmax(n), zmax(n));
else % resolve multiple maxima cases
    maxValues = localMax(sub2ind(size(localMax), ym, xm, zm)); % highest local max
    maxIdx = find(maxValues == max(maxValues));
    xmax(n) = xm(maxIdx(1));
    ymax(n) = ym(maxIdx(1));
    zmax(n) = zm(maxIdx(1));
    labelVect(n) = labels(ymax(n), xmax(n), zmax(n));

    % remove highest max from list
    xm(maxIdx(1)) = [];
    ym(maxIdx(1)) = [];
    zm(maxIdx(1)) = [];

    % compute distance of secondary maxima to primary
    dist2max = sqrt((xmax(n)-xm).^2 + (ymax(n)-ym).^2 + (zmax(n)-zm).^2);
    dist2com = sqrt((xcom(n)-xm).^2 + (ycom(n)-ym).^2 + (zcom(n)-zm).^2);
    mindist = min(dist2max,dist2com);

    % retain secondary maxima where mindist > threshold
    idx2 = find(mindist > dthreshold);
    if ~isempty(idx2)
        xmax2{n} = xm(idx2);
        ymax2{n} = ym(idx2);
        zmax2{n} = zm(idx2);
        nSecMax = length(idx2);
```

```
            nMaxima(n) = nSecMax+1;

            % split area
            area2{n} = area(n)*ones(nSecMax,1)/nMaxima(n);
            area(n) = area(n)/nMaxima(n);
            labelVect2{n} = labels(sub2ind(size(labels), ymax2{n}, xmax2{n}, zmax2{n}));

            %intensity values
            totalInt2{n} = totalInt(n)*ones(nSecMax,1)/nMaxima(n);
            totalInt(n) = totalInt(n)/nMaxima(n);
        end
    end
end

xmax2 =  vertcat(xmax2{:});
ymax2 = vertcat(ymax2{:});
zmax2 = vertcat(zmax2{:});
totalInt2 = vertcat(totalInt2{:});
area2 = vertcat(area2{:});
labelVect2 = vertcat(labelVect2{:});

% assign results to output structure
frameInfo.xmax = [xmax; xmax2(:)];
frameInfo.ymax = [ymax; ymax2(:)];
frameInfo.zmax = [zmax; zmax2(:)];
frameInfo.xcom = [xcom; xmax2(:)];
frameInfo.ycom = [ycom; ymax2(:)];
frameInfo.zcom = [zcom; zmax2(:)];
frameInfo.totalInt = [totalInt; totalInt2(:)];
frameInfo.area = [area; area2(:)];

frameInfo.nMaxima = nMaxima; % maxima per component
frameInfo.labels = [labelVect; labelVect2(:)];
frameInfo.nComp = nComp;


% prepare fields for tracker
nObj = length(frameInfo.xmax);
movieInfo(t).amp = zeros(nObj,2);
movieInfo(t).xCoord = zeros(nObj,2);
movieInfo(t).yCoord = zeros(nObj,2);
movieInfo(t).zCoord = zeros(nObj,2);

movieInfo(t).amp(:,1) = frameInfo.totalInt;
if choice==0
    movieInfo(t).xCoord(:,1) = frameInfo.xmax;
    movieInfo(t).yCoord(:,1) = frameInfo.ymax;
    movieInfo(t).zCoord(:,1) = frameInfo.zmax;
elseif choice==1
    movieInfo(t).xCoord(:,1) = frameInfo.xcom;
    movieInfo(t).yCoord(:,1) = frameInfo.ycom;
    movieInfo(t).zCoord(:,1) = frameInfo.zcom;
    end

    frameInfo.path = [];
    frameInfo.maskPath = [];

end

parent=fileparts(pwd);
if choice==0
    sfx='max';
elseif choice==1
    sfx='com';
end
save([parent,filesep,'Results',filesep,base,'sd3D_',sfx,'.mat'],'movieInfo');
```

B.2 mySpotDetector

Alternative 3D detection algorithm of my own autorship.

```
function movieInfo=mySpotDetector(path,Z,T,bt)
% Algorithm by Monica Delgado
% Uses ideas from SpotDetector, http://lccb.hms.harvard.edu/doc/spotDetector_101410.zip
%
% Reads a denoised 3D video in the form of a collection of 2D images
% Detects centers of 3D cell-like connected regions in each snapshot
% The cell-like characteristic is obtained by splitting regions
% that look like two or more sphere-like objects pasted
%
% INPUT path is the path to the folder where the images are contained
%       the code assumes that such folder follows the pattern created by
%       imstotiff.m, i.e.
%        - such folder has the name of the data set, with which also the images
%          names start
%        - inside this folder, there's another one named "processed" where all the
%          denoised images are contained
%        - the images are named name_Tddd_Zdd.tif
%          Images are also assumed to be squared
%       Z is the number of planes from a 3D "snapshot"
%       T is the total number of time points imaged
%       bt is an indicator
%        - if bt=1 then sweeping will be done from top to bottom
%        - if bt=0 then sweeping will be done from bottom to top (default)
% Results will be saved in folder Results under the main folder
%
% Output is movieInfo, appropiately formatted to be input to u-track
% For a movie with N frames, movieInfo is a structure array with N entries.
% Every entry has the fields xCoord, yCoord, zCoord (if 3D) and amp.
% If there are M objects in frame i, each one of these fields in
% moveiInfo(i) will be an Mx2 array, where the first column is the value
% (e.g. x-coordinate in xCoord and amplitude in amp) and the second column
% is all zero.
```

```
if nargin<4
    bt=0;
end

movieInfo=repmat(struct('xCoord',[],'yCoord',[],'zCoord',[],'amp',[]),T,1);

[~,base]=fileparts(path);
L=size(imread([path,filesep,'processed',filesep,base,'_T001_Z01.tif']),1);
imagen=zeros(L,L,Z); %imagen will contain the 3D image

for t=1:T
    % save 2D images into 3D compund
    tname=['_T',num2str(t,'%3.3d')];
    for z=1:Z
        sufix=['_Z',num2str(z,'%2.2d'),'.tif'];
        name=[path,filesep,'processed',filesep,base,tname,sufix];
        imagen(:,:,z)=imread(name);
    end

    fprintf('Starting analysis of time point %d...\n',t);

    % begins analysis
    bimage=imagen>0; %make binary image corresponding to 3d image
    CC=bwconncomp(bimage,6); % find 3D connected regions
    R=CC.PixelIdxList;
    n=CC.NumObjects;
```

```
stats=regionprops(CC,imagen,'BoundingBox','WeightedCentroid','Image');

for i=1:n % sweep all connected regions

    bb=ceil(stats(i).BoundingBox);
    %the first 3 entries of bb denote the coordinates of the upper
    %right pixel of the bounding box, the following 3 denote the
    %width of the bounding box along each dimension

    if ((bb(4)==1) || (bb(5)==1) ||(bb(6)==1))
        % if the region is really of dim less than 3, then there's no
        % problem determining its center
        movieInfo(t).xCoord(end+1,:)=[stats(i).WeightedCentroid(1) 0];
        movieInfo(t).yCoord(end+1,:)=[stats(i).WeightedCentroid(2) 0];
        movieInfo(t).zCoord(end+1,:)=[stats(i).WeightedCentroid(3) 0];
        movieInfo(t).amp(end+1,:)=[sum(imagen(R{i})) 0];

    else
        flag=0; %this flag will turn on if in some plane of the current
        %3D region there's more than one 2D connected region

        s=size(stats(i).Image);
        img2d=zeros(s);
        img2d(stats(i).Image==1)=imagen(R{i});
        % img2d is the non-binary version of output Image from
        % regionprops
% initialize: find connected 2D regions inside the first
% plane of the current 3D region
if bt==1
    cc2d=bwconncomp(stats(i).Image(:,:,end),8); % start on top plane
elseif bt==0
    cc2d=bwconncomp(stats(i).Image(:,:,1),8); % start on bottom plane
end
if cc2d.NumObjects>1
    flag=1;
end
prevPIL=cc2d.PixelIdxList;
for m=1:length(prevPIL) %for each 2d connected region
    if bt==1
        prevplane{m}=bb(6); % start on top plane
    elseif bt==0
        prevplane{m}=1; % start on bottom plane
    end
end

% prevPIL=previous Pixel Indexes List,
% sweep all subsequent planes included in the 3D region
% this algorithm will find the intersections of 2D regions
% between planes
if bt==1
    ss=bb(6)-1;
    ff=1;
    inc=-1;
elseif bt==0
    ss=2;
    ff=bb(6);
    inc=1;
end
for z=ss:inc:ff
    %find 2D connected regions in each plane of the current
    %3D region
    cc2d=bwconncomp(stats(i).Image(:,:,z),8);
    if cc2d.NumObjects>1
        flag=1;
    end
    newlist=cc2d.PixelIdxList; % current plane list of pixels in each region
    flaglist=1:cc2d.NumObjects;

    m=1; % new regions counter
    for j=1:length(prevPIL)
        flagj=0;
        % if a given previous 2D region does not have
        % intersection with regions in the current plane,
        % don't intersect it (which would erase it), keep it, as
        % it is a region which has probably ended in the
        % previous plane, that's what this flagj is for
        % Same way, we must keep the 2D regions in current
    % plane which had empty intersections with all previous
    % regions, flaglist helps here

    %compare each region found so far with
    %current regions, keep non-empty intersections
    for k=1:cc2d.NumObjects
        set=intersect(prevPIL{j},newlist{k});
        if  ~isempty(set)
            flagj=1;
            flaglist=setdiff(flaglist,k);
            PIL{m}=set;
            plane{m}=[prevplane{j},z];
            m=m+1;
        end
    end

    if flagj==0
        PIL{m}=prevPIL{j};
        plane{m}=prevplane{j};
        m=m+1;
    end

end

% keep 2D regions in current plane that matched no
% previous regions
```

```
                for k=flaglist
                    PIL{m}=newlist{k};
                    plane{m}=z;
                    m=m+1;
                end

                prevPIL=PIL;
                prevplane=plane;

            end % end of the current plane of a given 3D region

%             if flag==0
%                 movieInfo(t).xCoord(end+1,:)=[stats(i).WeightedCentroid(1) 0];
%                 movieInfo(t).yCoord(end+1,:)=[stats(i).WeightedCentroid(2) 0];
%                 movieInfo(t).zCoord(end+1,:)=[stats(i).WeightedCentroid(3) 0];
%                 movieInfo(t).amp(end+1,:)=[sum(imagen(R{i})) 0];
%             else
            % convert PIL and plane to 3D linear indeces of Image
            label=zeros(s);
            mm=1;
            for m=1:length(PIL)
                if ((length(PIL{m})>1) || (length(plane{m})>1))
                    lidx{mm}=ind2sub2ind(s(1:2),PIL{m},plane{m});
                    label(lidx{mm})=mm;
                    mm=mm+1;
                end
            end
            % find the weighted centroids of the regions found
            aux=regionprops(label,img2d,'WeightedCentroid');
            cntr=zeros(length(aux),3);
            for m=1:length(aux)
                cntr(m,:)=aux(m).WeightedCentroid;
                movieInfo(t).amp(end+1,:)=[sum(img2d(lidx{m})) 0];
            end
            %recover corresponding coordinates in original image
            movieInfo(t).xCoord(end+1:end+m,:)=[cntr(:,1)+bb(1) zeros(m,1)];
            movieInfo(t).yCoord(end+1:end+m,:)=[cntr(:,2)+bb(2) zeros(m,1)];
            movieInfo(t).zCoord(end+1:end+m,:)=[cntr(:,3)+bb(3) zeros(m,1)];
%            end

            PIL={};
            plane={};
            lidx={};
        end %ends case more than one z-slice

    end % end of current 3D region

end %end of current time image

parent=fileparts(pwd);
if bt==0
    sfx='bup';
```

```
elseif bt==1
    sfx='tdown';
end
save([parent,filesep,'Results',filesep,base,'mysd_',sfx,'.mat'],'movieInfo');
```

B.3 detecttr

This function uses *u-track* on the denoised images both for detection of 3D cell centers and for the actual tracking.

```
function H=detecttr(path,Z,T,d)

if nargin<4
    d=0;
end

prnt=pwd;
[~,name]=fileparts(path);
if d==0
    mkdir(path,'lessnoise')
end
movieInfo=repmat(struct('xCoord',[],'yCoord',[],'zCoord',[],'amp',[],'height',[]),T,1);
H=[];

for t=1:T
    cd([prnt,filesep,'SpotDetector'])
    tstr=num2str(t,'%3.3d');
```

```
for z=1:Z
    if d==0
        frame = double(imread([path,filesep,'original',filesep,name,'_T',tstr,'_Z',...
                num2str(z,'%2.2d'),'.tif']));
        denoised=denoise(frame,4,2);
        imwrite(uint8(denoised-1),[path,filesep,'lessnoise',filesep,name,'_T',tstr,...
                '_Z',num2str(z,'%2.2d'),'.tif'],'tiff');
    elseif d==1
        denoised=imread([path,filesep,'lessnoise',filesep,name,'_T',tstr,'_Z',...
                 num2str(z,'%2.2d'),'.tif']);
    end
    mi(z) = spotDetector(denoised);
    %H=[H; mi(z).area];
    H=[H; mi(z).totalInt./mi(z).area];
end
cd([prnt,filesep,'u-track111221',filesep,'u-track'])
tfdet=scriptTrackGeneral(mi,0,2,1,5,0,1,2);
% last parameter is 1 for all LN except z1, in which case it's 2
for i=1:length(tfdet)
    if any(isnan(tfdet(i).tracksCoordAmpCG(1:8:end)))
        print('Gap found in track %d, t = %d\n',i,t)
        print('****************************\n')
    else
      movieInfo(t).xCoord=[movieInfo(t).xCoord; mean(tfdet(i).tracksCoordAmpCG(1:8:end)) 0];
      movieInfo(t).yCoord=[movieInfo(t).yCoord; mean(tfdet(i).tracksCoordAmpCG(2:8:end)) 0];
      zc=sum(tfdet(i).tracksCoordAmpCG(4:8:end).*[tfdet(i).seqOfEvents(1,1):...
             tfdet(i).seqOfEvents(2,1)])/sum(tfdet(i).tracksCoordAmpCG(4:8:end));
          movieInfo(t).zCoord=[movieInfo(t).zCoord; zc 0];
          movieInfo(t).height=[movieInfo(t).height; tfdet(i).seqOfEvents(2,1)...
                              -tfdet(i).seqOfEvents(1,1)+1];
          movieInfo(t).amp=[movieInfo(t).amp; max([std(tfdet(i).tracksCoordAmpCG(1:8:end)),...
           std(tfdet(i).tracksCoordAmpCG(2:8:end)),std(tfdet(i).tracksCoordAmpCG(3:8:end))]) 0];
        end
    end
end

for i=1:T
%movieInfo(i).xCoord(:,1)=movieInfo(i).xCoord(:,1)*530.33/512;
%movieInfo(i).xCoord(:,1)=movieInfo(i).xCoord(:,1)*424.27/512; %for LN10
movieInfo(i).xCoord(:,1)=movieInfo(i).xCoord(:,1)*283.4/560; %for z1
movieInfo(i).yCoord(:,1)=movieInfo(i).yCoord(:,1)*283.4/560;
movieInfo(i).zCoord(:,1)=2.5*(movieInfo(i).zCoord(:,1)-1);
%the scaling factor is 2 for most LN except for z1 in which case it is 2.5
end

tf=scriptTrackGeneral(movieInfo,0,3,1,2,2,0,10);

cd(prnt)
%mkdir('Results',[name,'trd'])
save(['Results',filesep,name,'trd',filesep,name,'trdlnzamp.mat'],'movieInfo');
save(['Results',filesep,name,'trd',filesep,'tr',name,'-trdlnzamp.mat'],'tf');
```

Function *scriptTrackGeneral* is called in the following way:

```
function tracksFinal=scriptTrackGeneral(movieInfo,tosave,probDim,TW,minTL,LM,minSR,maxSR)
% Copyright (C) 2011 LCCB
%
% This file is part of u-track.
%
% u-track is free software: you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation, either version 3 of the License, or
% (at your option) any later version.
%
% u-track is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with u-track.  If not, see <http://www.gnu.org/licenses/>.
%
%
%% Detection results: movieInfo
%
%For a movie with N frames, movieInfo is a structure array with N entries.
%Every entry has the fields xCoord, yCoord, zCoord (if 3D) and amp.
%If there are M features in frame i, each one of these fields in
%moveiInfo(i) will be an Mx2 array, where the first column is the value
%(e.g. x-coordinate in xCoord and amplitude in amp) and the second column
```

```
%is the standard deviation. If the uncertainty is unknown, make the second
%column all zero.
%
% INPUT     tosave is the full path and name where the results will be
%                  saved
%           TW = gapCloseParam.timeWindow
%           minTL = gapCloseParam.minTrackLen
%           LM = parameters.linearMotion
%           minSR = parameters.minSearchRadius
%           maxSR = parameters.maxSearchRadius
```

B.4 testdetect

```
function error=testdetect(density,r, dm)
% Generates a 3D image with spherical cells, samples it with 17 2D planes
% and then tries to compute the centers of the cells with SpotDetector.
% INPUT     density is the number of cells
%           r is the radius of a cell (in pixels); assuming they're all spherical
%               and have the same size
%           dm choose the detection method
%               - 0: my method, bottom up
%               - 1: my method, top down
%               - 2: adapted 3D method, with maximum
%               - 3: adapted 3D method, with center of max
%               - 4: detection with u-track
% OUTPUT    error(1) is the sum of the distance between each original cell
%                and the closest computed (by mySpotDetector) cell
%            error(2) is the difference between the new density of cells
%                (computed by mySpotDetector) and the original density

%% Parameters
bside=254; % number of pixels on each side of the 2D slices
uheight=4; % distance (in pixels) between each pair of z slices
nslices=17; % number of z slices
height=(nslices-1)*uheight+2*r;
imagen=uint8(zeros(bside,bside,nslices));

%% Generate cells (spheres)
% rng('shuffle'); %doesn't work in MatlabR2010a
C=rand(density,3);
C(:,1:2)=0.499+bside*C(:,1:2);
C(:,3)=height*C(:,3);

%% Take 2D images
for i=1:density % sweep all generated cells
    cx=C(i,1);
    cy=C(i,2);
    cz=C(i,3);
    % figure out which integer z are comprised in the sphere
    zstrt=max([ceil(cz-r),r]);
    zend=min([floor(cz+r),height-r]);
    % for each plane with integer z comprised in the sphere
```

```
        for z=zstrt:zend
            if mod(z-r,4)==0 % check which integer z correspond to z-slice (ie were imaged)
                d=z-cz; % distance between center of sphere and current plane
                R=sqrt(r^2-d^2); % radius of the intersection of the sphere with the plane
                k=(z-r)/4+1; % corresponding number of z-slice
                % figure out which integer y are comprised in the circular intersection
                y0=max([ceil(cy-R),1]);
                y1=min([floor(cy+R),bside]);
                % for each integer y comprised in the intersection
                for y=y0:y1
                    sec=sqrt(R^2-(y-cy)^2);
                    % x distance from y axis (of the circle) to circunference at height y
                    x0=max([round(cx-sec),1]);
                    x1=min([round(cx+sec),bside]);
                    imagen(y,x0:x1,k)=128;
                end
            end
        end
    end
    %rescale centers
    C(:,3)=(C(:,3)-r)/4+1;

    % add noise to original image
    % noise=uint8(64*randn(size(imagen)));
    % imagen=imagen+noise;
```

```
%% Spot Detector -- Image denoising
% for i=1:nslices
%     detectionMask = denoise(imagen(:,:,i));
% end

%% Spot Detector 3D -- My Algorithm
switch dm
    case 0
        movieInfo=mySpotDetector(imagen,0);
    case 1
        movieInfo=mySpotDetector(imagen,1);
    case 2
        movieInfo=spotDetector3D(imagen,0);
    case 3
        movieInfo=spotDetector3D(imagen,1);
    case 4
        movieInfo=detecttr(imagen);
end
newdens=size(movieInfo.xCoord,1);
output=zeros(newdens,3);
output(:,1)=movieInfo.xCoord(:,1);
output(:,2)=movieInfo.yCoord(:,1);
output(:,3)=movieInfo.zCoord(:,1);

%% Sort results
[~,ord2]=sort(C(:,3));C=C(ord2,:);
```

```
[~,ord1]=sort(output(:,3));output=output(ord1,:);

%% Compute error
dist=zeros(newdens,density);
for i=1:newdens
    for j=1:density
        dist(i,j)=norm(output(i,:)-C(j,:));
    end
end
[minbycol,midx]=min(dist,[],1);
error(1)=sum(minbycol)/density;
error(2)=newdens-density;

cd /home/monica/Thesisv2/simulations
% this function needs to be improved in the following ways:
% - find a better way to quantify error
% - check the scaling when measuring error, I think it would make more
% sense to rescale movieInfo instead of the centers, but one needs to be
% careful with the offsets (ie pixels start at 0.5, not 0)
```

B.5 testtrack

```
function tf=testtrack(no_of_tracks,no_of_steps,diffcoeff)
% This function still needs a way of measuring the errors.
% This function tests u-track on no_of_tracks simulated tracks which follow
% Brownian motion with diffusion coefficient diffcoef. Tracks are followed
% for no_of_steps seconds.

curr=pwd;
[prnt]=fileparts(curr);

movieInfo=repmat(struct('xCoord',[],'yCoord',[],'zCoord',[],'amp',[]),no_of_steps,1);

%% Specify simulation parameters
dim = 3; % Dimensionality
tau = 6; % Sampling interval in s (0.001 = 1 ms)
x= 250;
z= 24;

%% Code for generating single-state random walks
%% (i.e. pure diffusion with a single diffusion coefficient)
displacements = normrnd(0, sqrt(diffcoeff*tau), [no_of_tracks,no_of_steps-1, dim]);
%% Calculate position at each step
startingpoint=cat(3,x*rand(no_of_tracks,1,2),z*rand(no_of_tracks,1,1));
position = cumsum(cat(2,startingpoint,displacements),2);

% This part 'registers' only those positions of the simulated tracks which
% are inside a box of size x by x by z (which represents the imaged area).
% for i=1:no_of_steps
%     xidx=intersect(find(position(:,i,1)>=0),find(position(:,i,1)<=x));
%     yidx=intersect(find(position(:,i,2)>=0),find(position(:,i,2)<=x));
%     zidx=intersect(find(position(:,i,3)>=0),find(position(:,i,3)<=z));
%     idx=intersect(xidx,yidx);
%     idx=intersect(idx,zidx);
%     movieInfo(i).xCoord=[position(idx,i,1),zeros(length(idx),1)];
%     movieInfo(i).yCoord=[position(idx,i,2),zeros(length(idx),1)];
%     movieInfo(i).zCoord=[position(idx,i,3),zeros(length(idx),1)];
%     movieInfo(i).amp=[ones(length(idx),1),zeros(length(idx),1)];
% end

% This part keeps full tracks even if they wander off the 'imaged' box
for i=1:no_of_steps
    movieInfo(i).xCoord=[position(:,i,1),zeros(no_of_tracks,1)];
    movieInfo(i).yCoord=[position(:,i,2),zeros(no_of_tracks,1)];
    movieInfo(i).zCoord=[position(:,i,3),zeros(no_of_tracks,1)];
    movieInfo(i).amp=[ones(no_of_tracks,1),zeros(no_of_tracks,1)];
end

cd([prnt,filesep,'u-track110523',filesep,'u-track'])
tf=scriptTrackGeneral(movieInfo,[prnt,filesep,'Results',filesep,'testtr-',...
   num2str(no_of_tracks),'-',num2str(diffcoeff,'%1.2f'),'.mat'],dim);
cd(prnt)
```

Appendix C

Additional plots

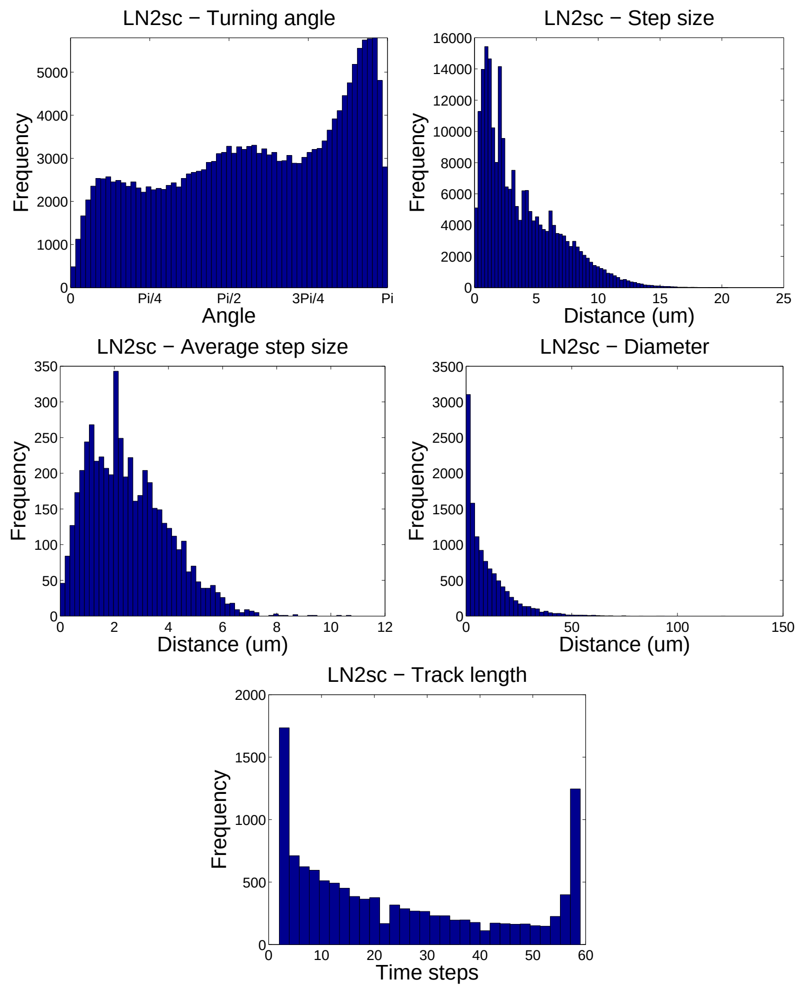


Figure C.1: Histograms for data set LN2 using *mySpotDetector*-bup as detection method with the lower level of denoising.

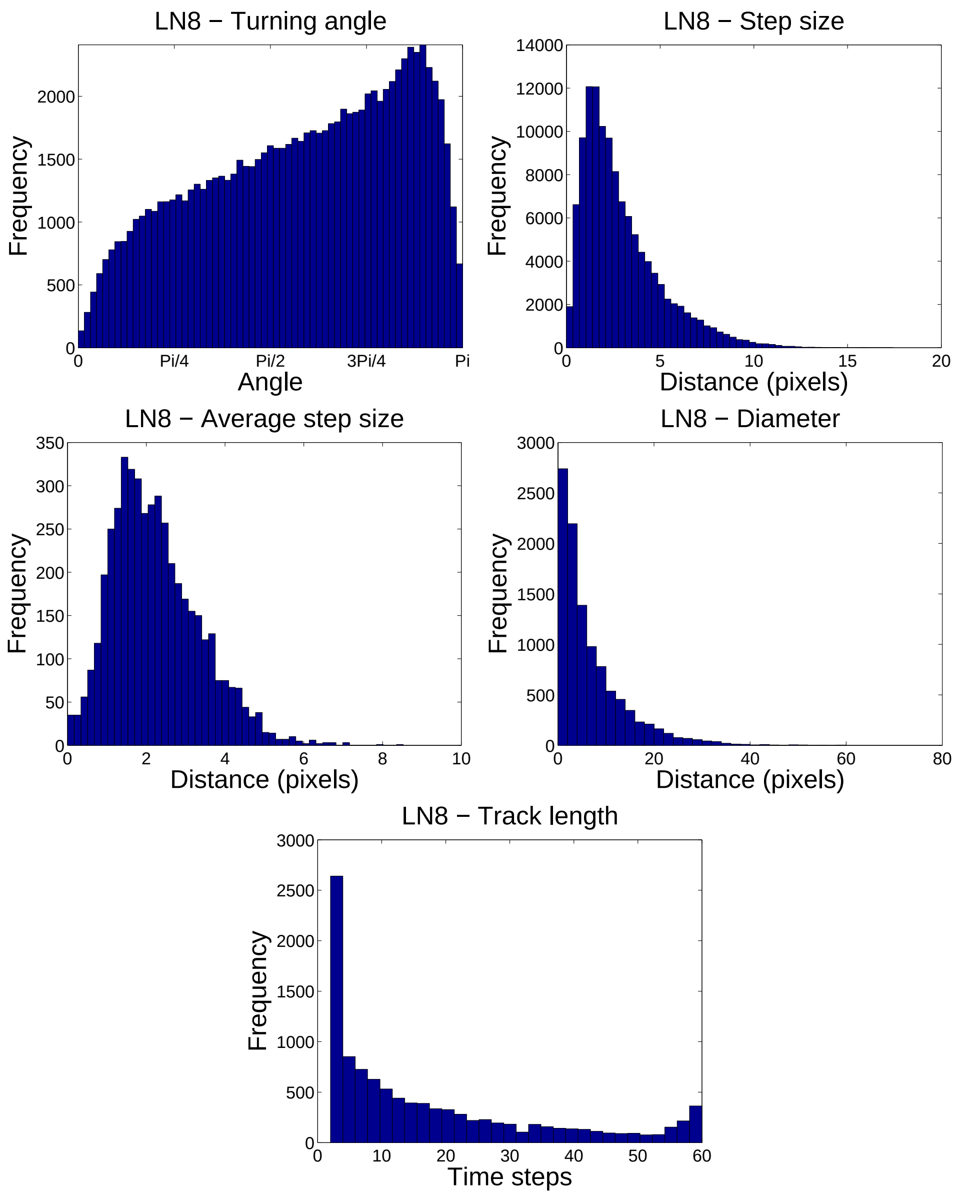


Figure C.2: Histograms for data set LN8 using *mySpotDetector*-bup as detection method with the lower level of denoising.

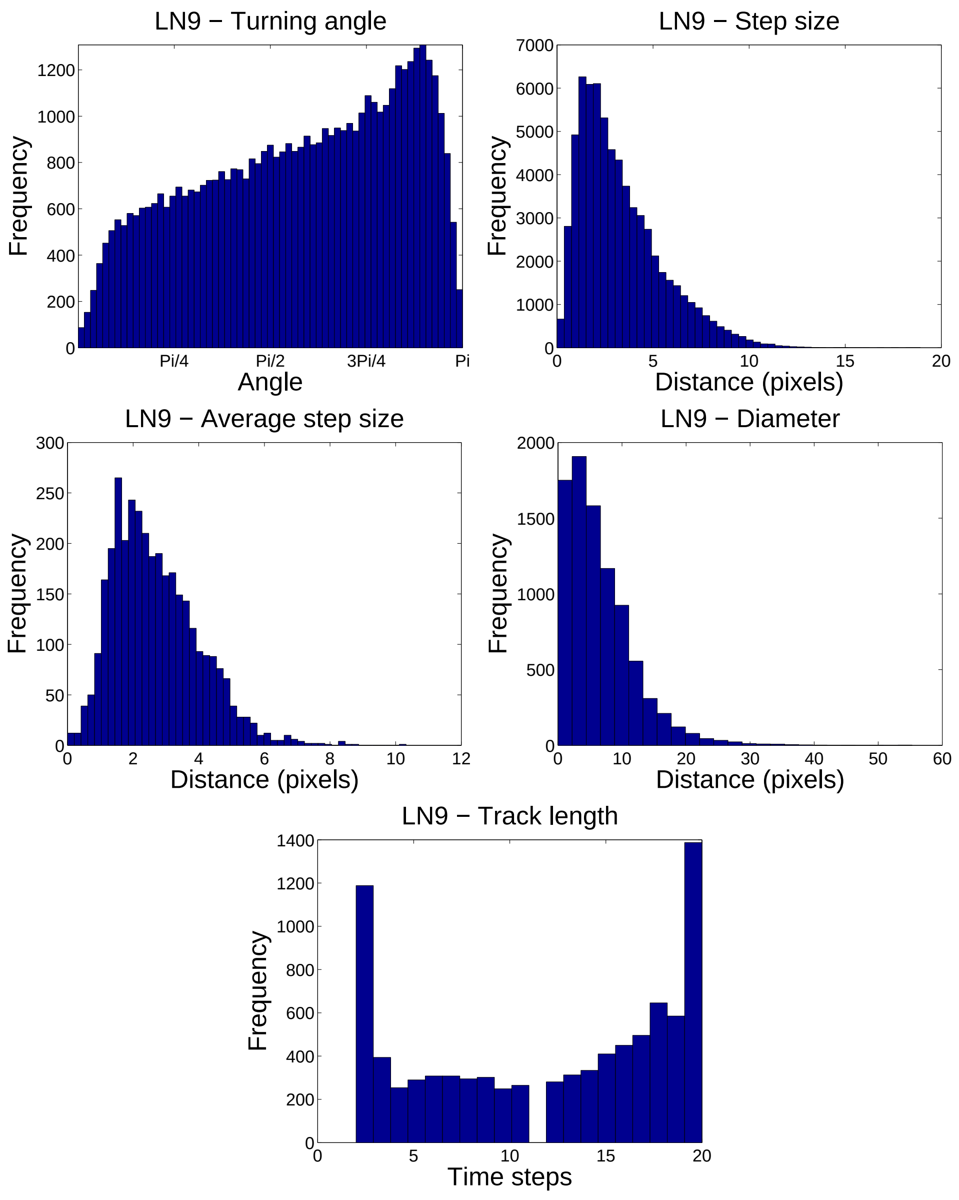


Figure C.3: Histograms for data set LN9 using *mySpotDetector*-bup as detection method with the lower level of denoising.

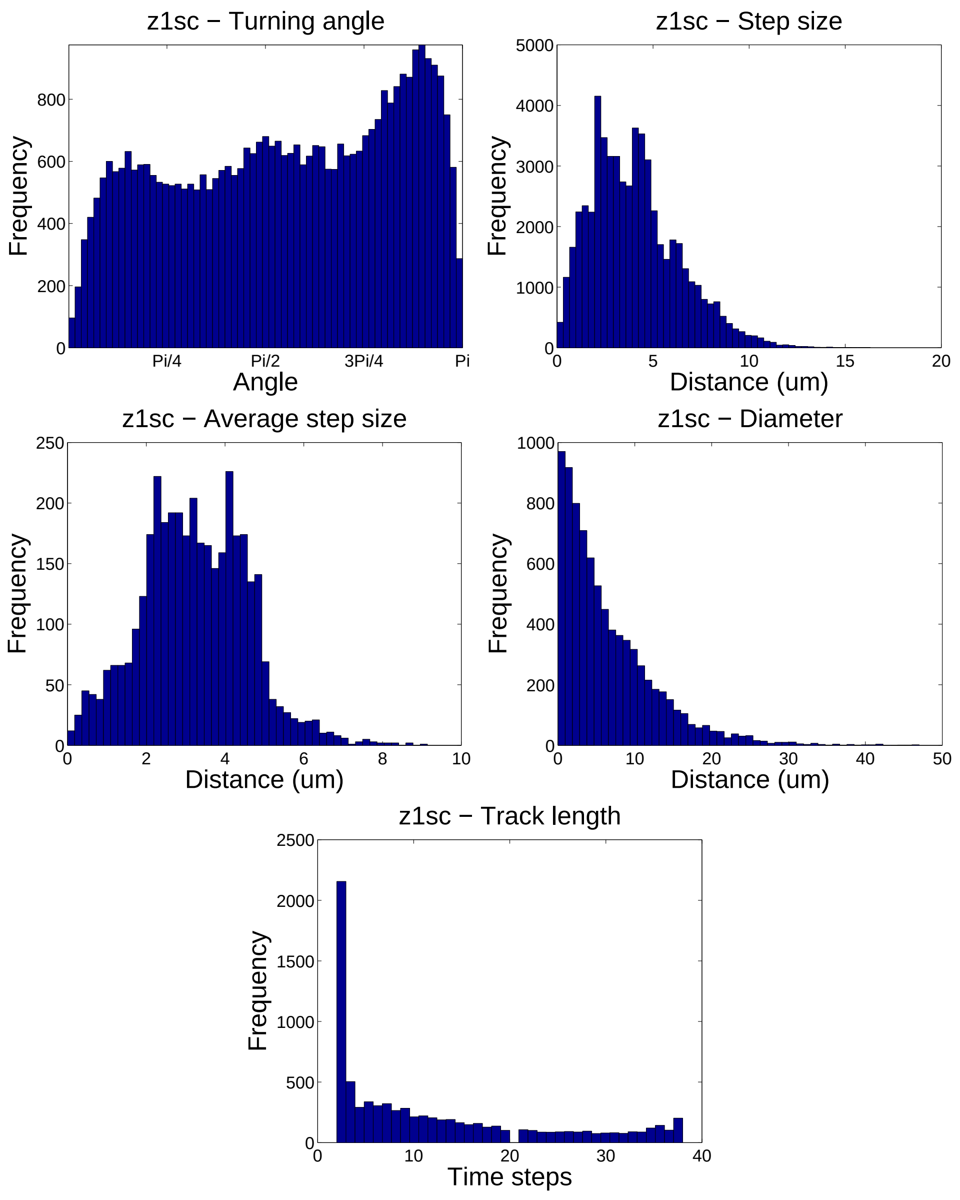


Figure C.4: Histograms for data set z1 using *mySpotDetector*-bup as detection method with the lower level of denoising.

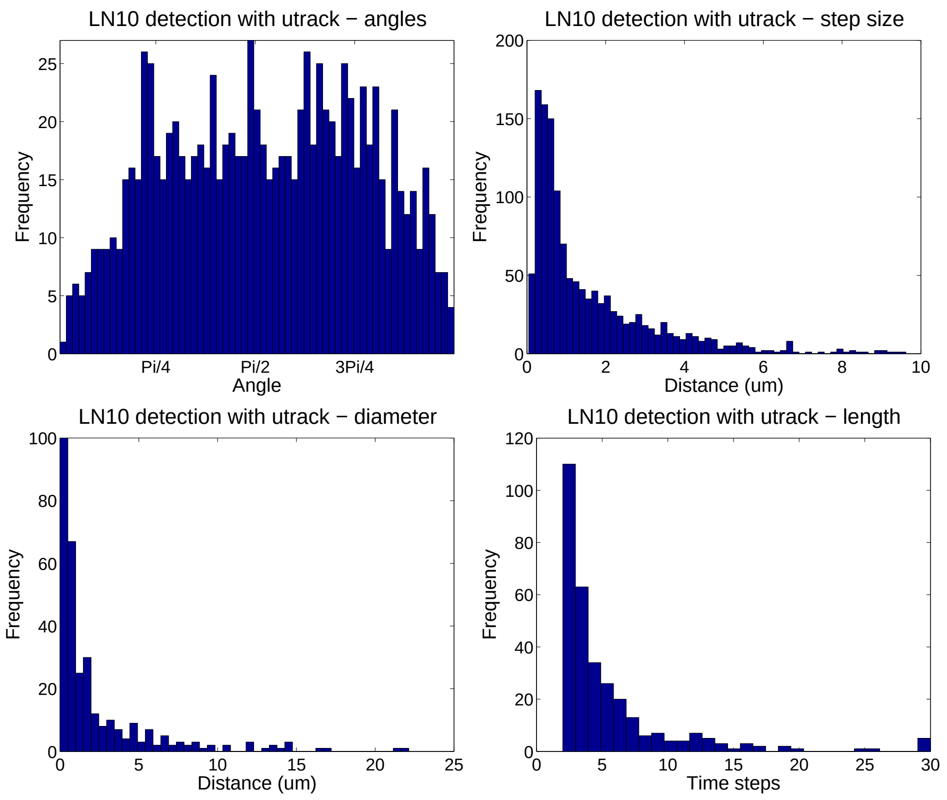


Figure C.5: Histograms corresponding to data set LN10 using *u-track* as method of detection, the z-coordinate determined as the normal average of all planes which contain the cell. For the tracking step, a pure Brownian motion model was assumed.

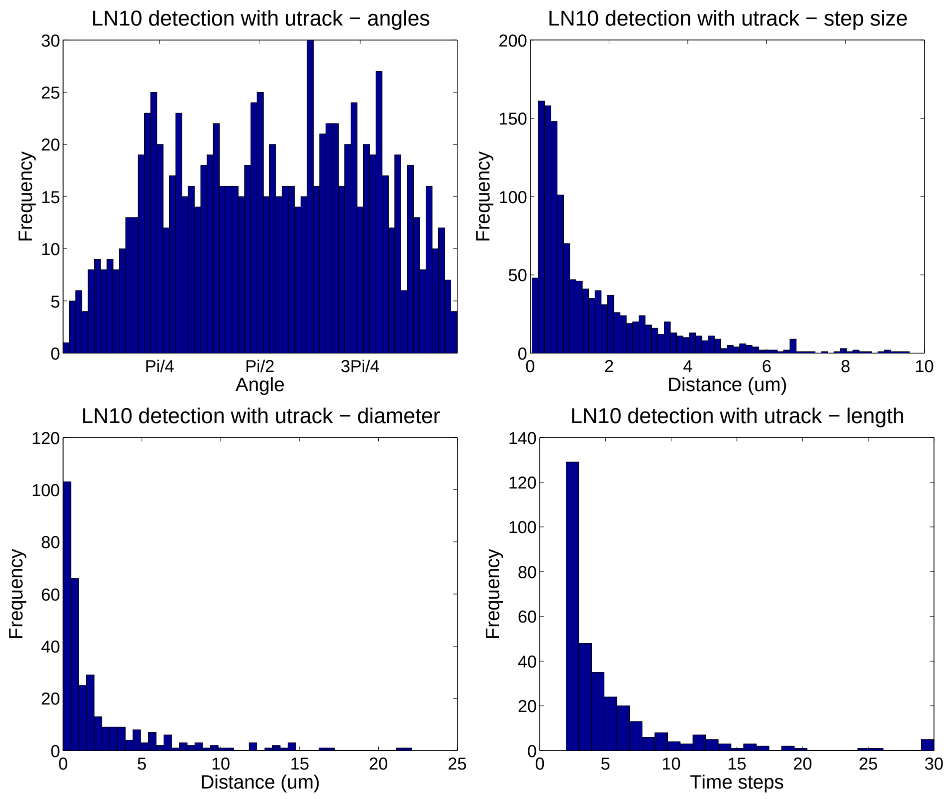


Figure C.6: Histograms corresponding to data set LN10 using *u-track* as method of detection, the z-coordinate determined as the normal average of all planes which contain the cell. For the tracking step, a random motion plus movement with constant velocity model was assumed.

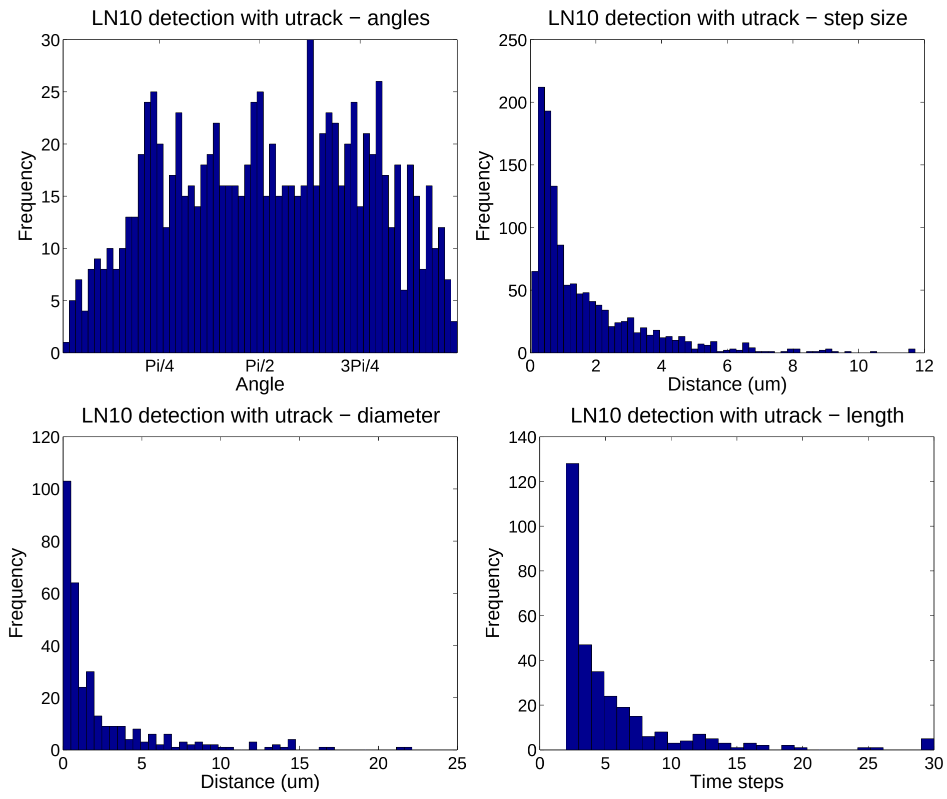


Figure C.7: Histograms corresponding to data set LN10 using *u-track* as method of detection, the z-coordinate determined as the normal average of all planes which contain the cell. For the tracking step the motion model was random motion and movement along a straight line but with the possibility of immediate direction reversal.

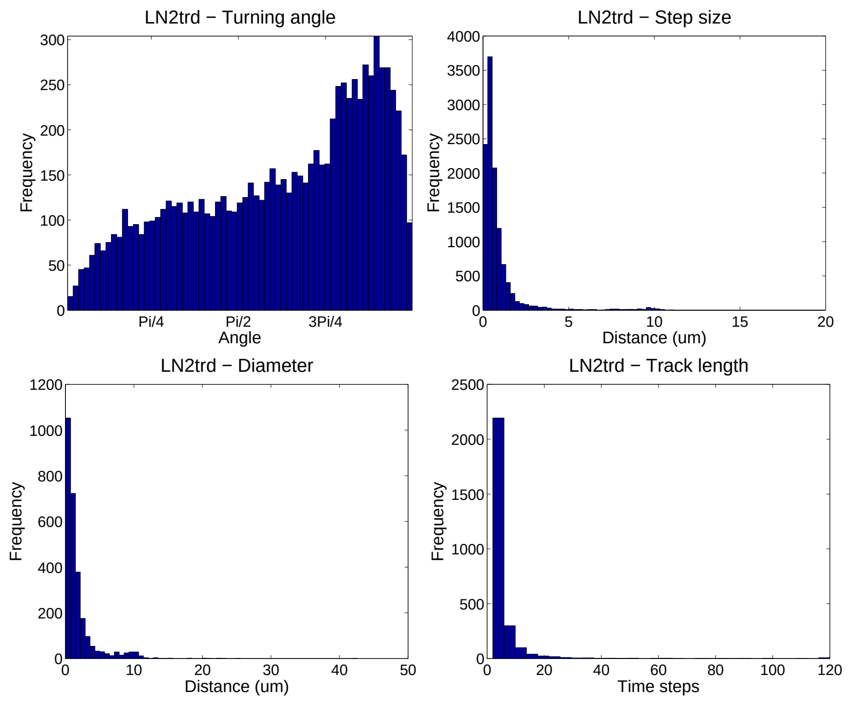


Figure C.8: These histograms correspond to the analysis of the endogenous data set LN2, using *u-track* as method of detection and with the higher level of denoising.

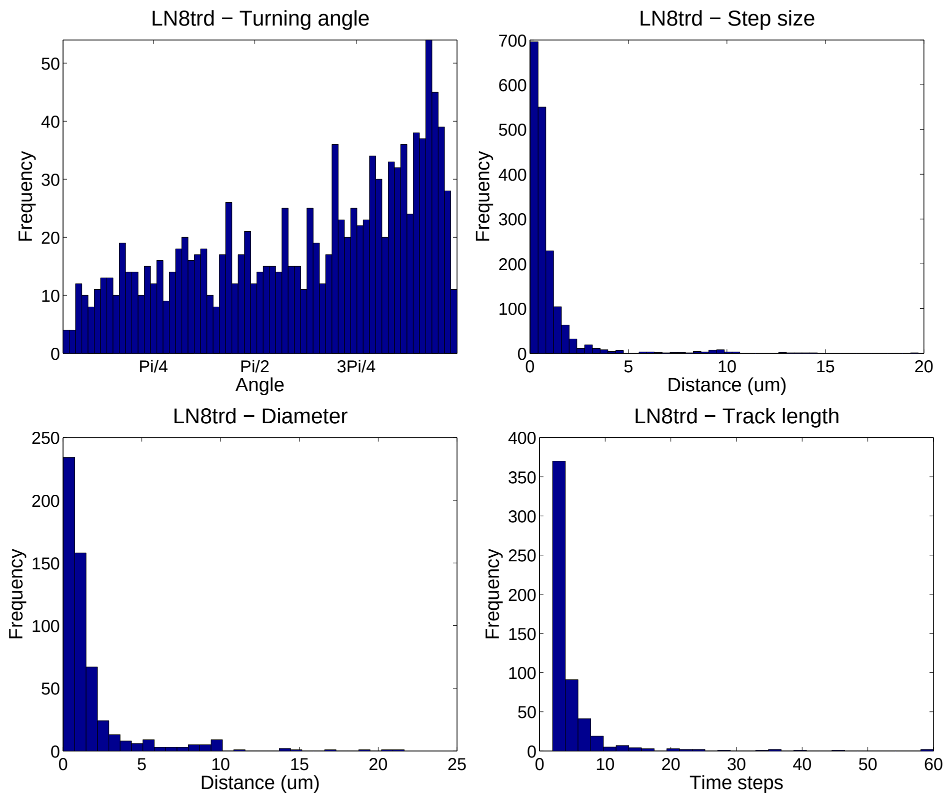


Figure C.9: These histograms correspond to the analysis of the endogenous data set LN8, using *u-track* as method of detection and with the higher level of denoising.

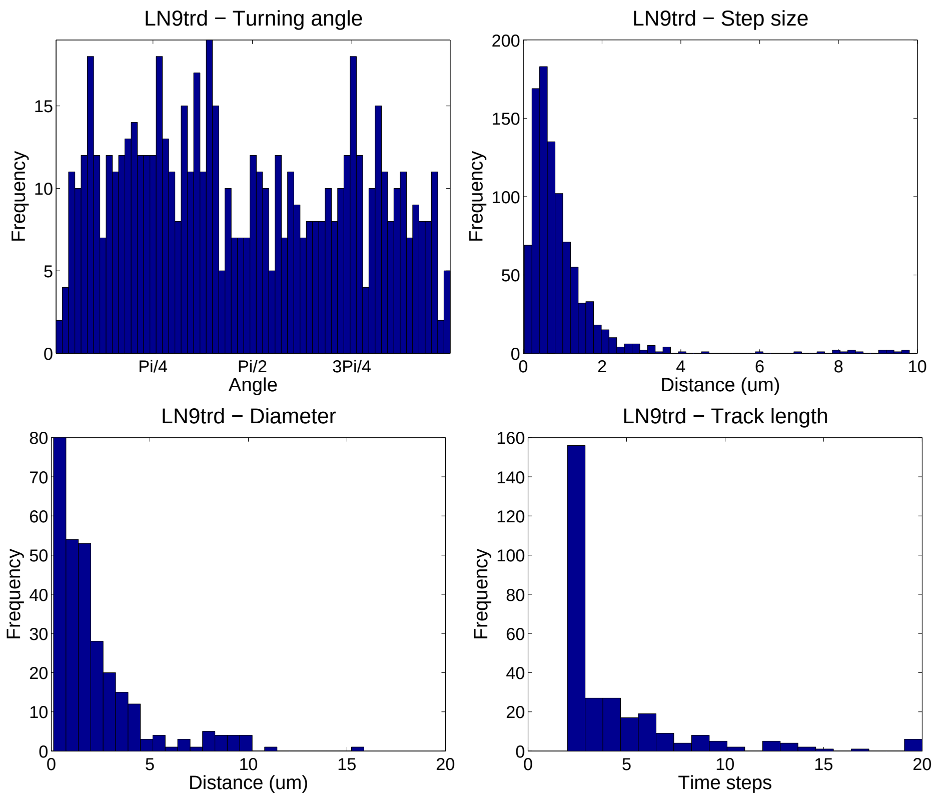


Figure C.10: These histograms correspond to the analysis of the endogenous data set LN9, using *u-track* as method of detection and with the higher level of denoising.

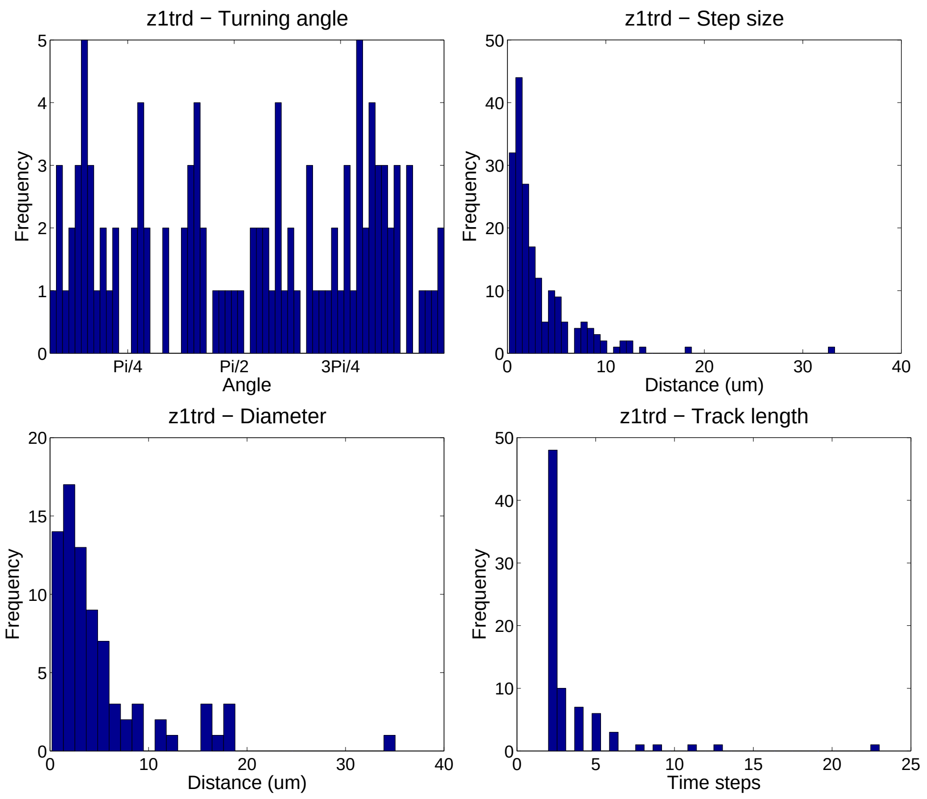


Figure C.11: These histograms correspond to the analysis of the endogenous data set Z1, using *u-track* as method of detection and with the higher level of denoising.