Moving in the Dark

Mathematics of Complex Pedestrian Dynamics

H. C. V. MeghaShyam Veluvali

Faculty of Health, Science and Technology

Master thesis in Mathematics

Second Cycle, 30 hp (ECTS)

Supervisor: Prof. dr. habil. Adrian Muntean, Karlstad University, Sweden

Examiner: Prof. dr. Eddie Wadbro, Karlstad University, Sweden

Karlstad, 2nd June, 2023

Dedication

To Amma and Nanna

Abstract

The field of mathematical modelling for pedestrian dynamics has attracted significant scientific attention, with various models proposed from perspectives such as kinetic theory, statistical mechanics, game theory and partial differential equations. Often such investigations are seen as being a part of a new branch of study in the domain of applied physics, called sociophysics. Our study proposes three models that are tailored to specific scenarios of crowd dynamics.

Our research focuses on two primary issues. The first issue is centred around pedestrians navigating through a partially dark corridor that impedes visibility, requiring the calculation of the time taken for evacuation using a Markov chain model. The second issue is posed to analyse how pedestrians move through a T-shaped junction. Such a scenario is motivated by the 2022 crowd crush disaster took place in Itaewon district of Seoul, Korea. We propose a lattice-gas-type model that simulates pedestrians' movement through the grid by obeying a set of rules as well as a parabolic equation with special boundary conditions.

By the means of numerical simulations, we investigate a couple of evacuation scenarios by evaluating the mean velocity of pedestrians through the dark corridor, varying both the length of the obscure region and the amount of uncertainty induced by the darkness. Additionally, we propose an agent-based-modelling and cellular automata inspired model that simulates the movement of pedestrians through a T-shaped grid, varying the initial number of pedestrians. We measure the final density and time taken to reach a steady pedestrian traffic state. Finally, we propose a parabolic equation with special boundary conditions that mimics the dynamics of the pedestrian populations in a T-junction. We solve the parabolic equation using a random-walk numerical scheme, compare with a finite difference approximation. Furthermore, we prove rigorously the convergence of the random walk scheme to a corresponding finite difference scheme approximation of the solution.

Keywords

Mathematical modelling of pedestrian dynamics, stochastic systems, evacuation time, random walk method, parabolic equation, finite difference method.

MSC2020 Classification

65M06, 60K50, 60J10, 68Q87

Contents

1 Background and related work **6**
1.1 Background 6
1.2 Models of pedestrian dynamics: a literature survey 6
1.3 Objectives 8

2 Moving in the dark **10**
2.1 Introduction 10
2.2 Setting the problem 10
2.2.1 Moving in the dark as a stochastic process 11
2.2.2 Estimating the residence time as a gambler's ruin time 14
2.3 Simulation results 14
2.4 Discussion 17

3 Modelling crowd crush at a T-junction **18**
3.1 Introduction 18
3.2 Modelling approach: Rule-set for the discrete dynamics 19
3.2.1 Algorithm for the discrete dynamics 20
3.3 Simulation output 22
3.4 Discussion 25

4 Pedestrian dynamics using a parabolic PDE formulation **26**
4.1 Introduction 26
4.2 Problem 1 26
4.3 Finite difference approximation scheme 27
4.3.1 The algorithm 28
4.4 Random walk approximation scheme 29
4.4.1 The algorithm 29
4.4.2 Convergence of the random walk numerical scheme 31
4.5 A gradient random walk approximation scheme 33
4.5.1 Problem 2. 34
4.5.2 The algorithm 34
4.5.3 Convergence of the gradient random walk scheme 36
4.6 Numerical output and error comparison 41
4.7 Discussion 44

5 Conclusions and future work **45**
5.1 Bidirectional flows and network of corridors 45

5.2 Mean-field approximation for the T-junction model 45
5.3 Gradient random walk approximation scheme for non-zero boundary conditions 46
5.4 Further comments 46

References **47**

A Implementation of the moving in the dark problem **53**

B Implementation of the stochastic simulation for the T-junction problem **55**

C Implementation of the numerical schemes for the PDE problems **58**

Chapter 1

Background and related work

1.1 Background

In the day and age of increasing migrations to urban spaces, the number of people accessing public spaces is also increasing rapidly. The challenges posed by the pedestrian dynamics are of a great importance at present because large sets of populations are using public transport. We also observe that the people accessing spaces like shopping malls and public parks are also increasing. This fact usually affects the comfort of the people and is related with the critical safety matters. A mathematical understanding of pedestrian dynamics is necessary to be cautious of any possible disasters. To develop such an understanding, there are various science-based approaches to the problems posed by pedestrian dynamics. Finally, we may use such information to avoid disasters and design public spaces that also ensure safety.

1.2 Models of pedestrian dynamics: a literature survey

A lot of work has been done in this field. In this section, we only give a brief overview on some of the most important results connected to our problem.

We refer the reader to [2, Chapter 16] and [25] where the authors present stochastic transport dynamics of pedestrian flows. In [2, Chapter 16] , the authors introduce social force models, which are inspired by Newtonian mechanics. They find these models useful because, at higher densities of the populations, other models of discrete dynamics may fail. To validate the model, the authors recommend using a fundamental diagram [1], as well as collecting empirical data from videos.

We find [14, 20] relevant to our context. In [14], the authors propose a model to describe the evacuation of pedestrians out of a dark room. This model is a combination of a social force model and asymmetric random walk. They have also conducted experiments to validate the model. In [20], the authors approach the same problem to provide certain statistical characteristics of the evacuation dynamics.

[1] A "fundamental diagram" is a relation between the flux ($\vec{J}$) and the density (ρ) of the pedestrians

A floor-field-based approach to pedestrian dynamics is introduced in [25]. Such a model determines the dynamics of the pedestrian based on the current position, the surrounding environment, and the desired direction of each pedestrian. The spatial and dynamic memory, and occupation profile of the pedestrians decide the value of the transition probability.

In [1], a model based on the concept of floor fields is presented in the context of evacuate a large room. The authors observe the counterflow and lane formation. In [15], the discretization effects and influence of maximum speed in Cellular Automata models for pedestrian dynamics is discussed. The authors modelled their system for various cell sizes and maximum speeds of pedestrians. Furthermore, they worked on the evacuation times for a large group in a single room.

In [21], the pedestrian dynamics is described as a visuospatial search of walkers in urban environments through random walks. The authors have constructed networks out of digital maps of selected cities. They were trying to see how long randomly walking pedestrians take to travel through a network of streets when their initial positions are randomly assigned. They are blinded passively by their lack of awareness besides walking in broad daylight

The paper [28] illustrates a way to model pedestrian dynamics as a continuous random walk. By using a continuous random walk approach, we can arrive at a partial differential equation (PDE) for the time-dependent probability distribution function. This becomes an interesting approach in modelling high density pedestrian flows in [2, Chapter 16] for issues regarding high density of pedestrians in a random walk type discrete systems.

In [29], an experiment-based study of pedestrian dynamics in straight corridors and T-junctions is discussed. For various values of the width of the corridor, we are presented with the fundamental diagram of the dynamics. They observe a phase transition type of behaviour with respect to their boundary value parameters and the width of the corridor. Likewise, in [9], a floor-field based cellular automata model is used to understand the dynamics of merging pedestrian streams at a T-junction. In [9], results from the simulations are compared with the results with the experimental data in [29].

The models of pedestrian dynamics have largely been studied but seldom are they posed for dark and obscure scenarios, where visibility for the pedestrians is limited. Let us look at such studies that have dealt with people moving in the dark. In [6], evacuation dynamics is studied in a 2D setting where a part of the domain is dark. It is possible to present the problem posed in a 1-D setting too as a Markov chain process. The second chapter of this thesis deals with similar problem in a 1D setting. We note that the visibility in a room may not always be due to the absence of light. In [22], the authors propose a two-scale model for evacuation scenarios involving fire and smoke through a complex geometric corridor arrangement. In [4], the authors discuss how evacuation through a smoke-filled corridor takes place in the presence or absence of communication. The article [18] deals with another interesting problem where the pedestrians need to find an exit in the dark. Likewise, in [7], the authors discuss path formation and effects of group formation on evacuation time. Finally, the article [3] discusses different modelling strategies for problems of social mechanics.

We also point to the article [8] that provides us with a comprehensive account of multiscale modelling of pedestrians dynamics. It is extremely useful because it categorizes the various approaches we discussed like the social force models and cellular automata models into sub-microscopic, microscopic, mesoscopic and macroscopic scales. Using this article we can make

a choice of the modelling approach depending on the scenario we are dealing with.

The modelling problem of pedestrian dynamics falls under the larger category the dynamics of interacting agents. In such problems, it is often hard to visualize the dynamics of the interacting agents. But certain software programs been freely available for researchers to visualize and see the dynamics of interacting agents. We direct the reader to NetLogo [27] [2], a freely available software for discrete dynamic models. This software is mainly designed for agent-based modelling. One can get easily numerical results and able to illustrate the complexities arisen from local interactions.

1.3 Objectives

In this thesis, we study three different models of pedestrian dynamics:

- A Markov-chain model for pedestrians walking through a partially dark corridor.
- A stochastic model for three human crowds going across a T-shaped junction.
- An initial-boundary value parabolic PDE model for a crowd navigating a street with inflow of pedestrians at a point.

We are interested in numerically simulating the local fluxes and densities of pedestrians. With these simulations at hand, we aim to calculate evacuation times and eventual congestions. The thesis is structured as follows:

In chapter 2, we give our Markov chain model. The dynamics of the pedestrians is determined by a given set of rules. We discuss these rules of dynamics and calculate the mean time taken by the pedestrians to cross the corridor. We define such a mean time as the residence time.[3] This residence time and therefore the mean velocity of the pedestrians are presented and discussed.

In chapter 3, we present our stochastic model for the dynamics of three interacting crowds driven by different purposes in the neighbourhood of a T-junction. We are motivated by the crowd crush incident that took place in Seoul on 29th October 2022. By examining the street junction where this crowd crush occurred, this model is constructed to understand the movement of crowds navigating through a T-shaped junction. In this model, a pedestrian is selected randomly at each time step and is allowed to move as per the pedestrian's motive. We note that the random selection may not exactly correspond to the crowd behaviour and thus, it is a modelling choice. Due to our choice of the rules, we observe a crowd crush of pedestrians as an emergent phenomenon. The results lead to a discussion on steady crowd behaviour and on the time taken for such steady behaviour to take place.

In chapter 4, we formulate a parabolic PDE problem to describe the dynamics of pedestrians in a simplified T-junction. We impose the Dirichlet and Robin type boundary conditions, at the left and right boundaries respectively. We propose a random walk based numerical scheme to solve the PDE. As the analytical solution is unavailable, we test the performance of the

[2] https://ccl.northwestern.edu/netlogo/index.shtml

[3] In the crowd dynamics literature, the residence time is also referred to as evacuation time.

random walk scheme by comparing the random walk approximation with a finite difference approximation. Furthermore, we study a numerical scheme based on a gradient-random walk method. We present then numerical simulation results obtained by both suggested random walk schemes.

Chapter 5 concludes this work. We present there a few comments on our main findings along with a look forward on what else can be possibly done in this framework.

Chapter 2

Moving in the dark

2.1 Introduction

In this chapter, we model the motion of a single pedestrian walking along a partially dark corridor. In Figure 2.2.1, we give a sketch of the walking area of the pedestrian. Via this model, we wish to mimic an evacuation scenario where pedestrians find it hard to perceive their surroundings visually. We assume that the pedestrian's visibility is non-existent from the start of its motion until a certain point in the corridor. We identify this section of it as the dark region. Having walked through this part of the corridor, the pedestrian enters the "visibility region". In the visibility region, the pedestrian gains the ability to perceive the surroundings and then moves towards the exit. We are interested in finding out the typical time it takes to evacuate pedestrians through such a corridor of partial visibility.

Our approach is heavily inspired by [5]. We use the idea of *residence time* proposed in this paper. It is worth noting that the residence time is defined as the typical time taken to cover an entire lattice. In the following sections, we show how to calculate the residence time in the context of this problem.

2.2 Setting the problem

Let the pedestrian be taking steps on a one-dimensional lattice of size $L \in \mathbb{N}$. L is also the total number of cells in the lattice. Each cell is of length 1 metre. The length of the visibility region is of size $L_v \in \mathbb{N}$ and $L_v \leq L$. The corridor is dark for the initial $L - L_v$ metres and visible for the rest of the length as illustrated in Figures 2.2.1 and 2.2.2.

Figure 2.2.1 is a sketch of the geometry where the walking space of the pedestrians. Observe that the dark region is of length $L - L_v$ cells and the visibility region is of length L_v cells. In the dark region, the pedestrian's behaviour is modelled as a symmetric random walk. When situated in the visibility region, the walking behaviour is modelled as an asymmetric random walk with a bias for moving towards the corridor's exit.

When situated in the dark region, the pedestrian takes a step to the right or left with equal probability of 0.5. When situated in the visibility region, the pedestrian takes a step to the right with $0.5 + \varepsilon$ probability and to the left with a $0.5 - \varepsilon$ probability where $\varepsilon \in (0, 0.5)$. By moving to

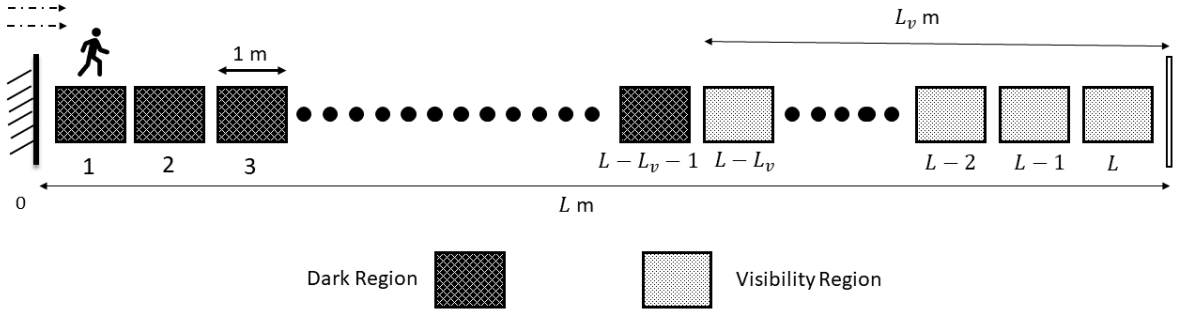


Figure 2.2.1: A sketch of the geometry of the partially dark corridor

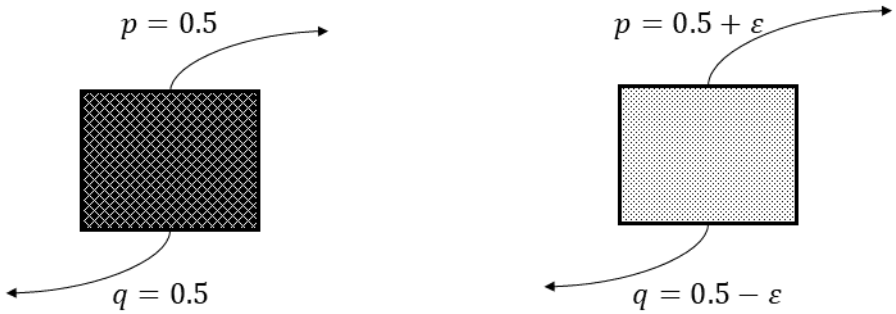


Figure 2.2.2: A diagram representing the transition probabilities of the pedestrian.

the left, the position gets added by -1 cell and by moving to the right, the position gets added by $+1$ cell.

Since the dynamics are characterized by a random walk process, the pedestrian can possibly reach any of the right or the left boundary cells. By reaching the left boundary cell 0, the pedestrian is moved to cell 1. By reaching the right boundary, the pedestrian is understood to have exited the walking space. Thus, we observe the left and right boundaries to be reflective and absorptive respectively.

In this chapter, the aim is to find the mean velocity as a function of the model parameters L_v and ε. The dynamics is simulated with the help of a numerical scheme indicated in the following section.

2.2.1 Moving in the dark as a stochastic process

Let us look at the problem as a stochastic process in this section. We find [24, Chapter 4] extremely useful in understanding these concepts.

Consider that the pedestrian to be taking a N number of steps in an experiment. Each step taken is denoted by Δx and is of magnitude 1 m. The position of the particle at a time step t is denoted by X_t. Let us define a set S as, $S := \{0, 1, 2, \ldots L\}$. X_t is a dimensionless quantity that can only take the values in S. The position of the particle at the succeeding time step, i.e. X_{t+1} can either be $X_t + 1$ or $X_t - 1$. X_{t+1}. Its transition to one of the values occurs probabilistically. The probabilistic transition is characterized by a pre-defined transition probability. We note that all transitions are identical and random in nature.

If $T \subset \mathbb{N}$ denote the index set indicating the time steps taken. The set of steps taken by the pedestrian $W = \{X_t : t \in T\}$ is a stochastic process on a probability space $(S, \mathcal{F}, \mathbf{P})$. Where $\mathcal{F}$ if the set of all the possible events and $\mathbf{P}$ be the probability function $\mathbf{P} : \mathcal{F} \to [0, 1]$.

For an $i \in S$, let $\mathbf{P}(X_t = i)$ denote the probability that $X_t = i$. Then it holds,

$$\mathbf{P}\Big(X_{t+1} = i + 1 | X_t = i\Big) + \mathbf{P}\Big(X_{t+1} = i - 1 | X_t = i\Big) = 1.$$

If $\mathbf{P}(X_{t+1} = j | X_t = i)$ is denoted by $p_{i,j}$ for, $i, j \in S$ we have,

$$p_{i,j} = \begin{cases} 1, & \text{if } i = 0, j = 1 \\ \frac{1}{2}, & \text{if } 1 \leq i < L_v \text{ and } j = i \pm 1 \\ \frac{1}{2} + \varepsilon, & \text{if } L_v \leq i < L \text{ and } j = i + 1 \\ \frac{1}{2} - \varepsilon, & \text{if } L_v \leq i < L \text{ and } j = i - 1 \\ 1, & \text{if } i = L, j = L \\ 0 & \text{for all other transitions} \end{cases} \tag{2.1}$$

Observe that,

$$\mathbf{P}(X_{t+1} = i) = p_{i-1,i}\, \mathbf{P}(X_t = i - 1) + p_{i+1,i}\, \mathbf{P}(X_t = i + 1) \tag{2.2}$$

The state of X_{t+1} only depends on its preceding state X_t. This property makes the sequence $\{X_n\}$ a Markov chain. If $\mathbf{P}(X_{t+1} = j | X_t = i)$ is denoted by $p_{i,j}$ for $i, j \in S$, we have

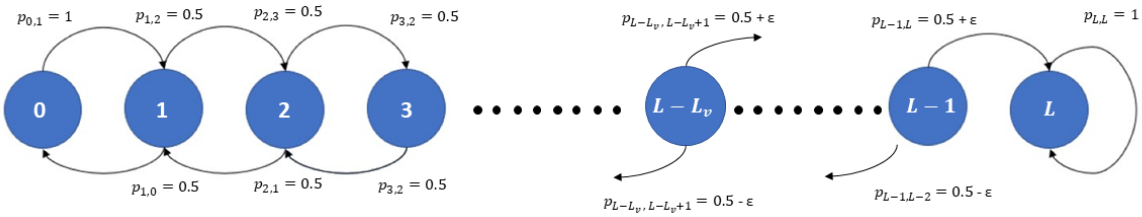


Figure 2.2.3: A diagram representing the pedestrian dynamics as a Markov chain

In (2.1) the parameters, recall that L_v, L and ε are fixed arbitrarily. Additionally, we set $L_v \in (0, L)$, $\varepsilon \in (0, 0.5)$.

A state $j \in S$ is defined to be *accessible* from state $i \in S$, denoted as $i \to j$, if it is possible to reach the state j from i in finite number of transitions. That is to say, when there exists an $m \in \mathbb{N}$ for which

$$i \to j \text{ when } P(X_t + m = j | X_m = i) > 0.$$

Two states $i, j \in S$ are identified *communicating* and written as $i \leftrightarrow j$ if $i \to j$ and $j \to i$. A set $C \subset S$ is said to be a **communicating class** if $i \leftrightarrow j$, for all $i, j \in C$. Consider $C_1 = \{L\}, C_2 = \{0, 1, 2, \ldots, L-1\}$. If $X_0 = L$ then $X_t = L$ for all $t \in \mathbb{N}$. If $x_0 \in C_2$, X_t can attain any value in C_2 for a large enough $t \in \mathbb{N}$. This implies that all the states in C_2 communicate with each other. So C_1 and C_2 are two disjoint communicating classes in S. In this particular case, observe that $C_1 \cup C_2 = S$.

A communicating class C is called closed if $X_t \in C$, and it holds that $X_{t+n} \in C$ for all $n \in \mathbb{N}$. C_1 is a closed communicating class or closed class for $X_t = L$ it holds that $X_{t+n} = L$ for all $n > 0 (n \in \mathbb{N})$. Observe though that, C_2 is not a closed class since for $X_t = L - 1$ it is possible for X_{t+n} to be L for some $n > 0 (n \in \mathbb{N})$. When a communicating class is not closed, we identify all states of such a class as *transient states*.

When we have a set of transient states, it is possible to calculate the mean number of visits from one transient state to another transient state.

The left boundary of the walking stretch is reflective because $X_t = 0$, $X_{t+1} = 1$ because $p_{0,1} = 1$. A pedestrian starting from $X_0 = 1$ would keep taking random steps until $X_n = L$ for some $n > 1$. If $X_n = L$ then $X_{n+1} = L$ only. Hence, the right boundary is absorptive.

It is interesting to find out the mean value of n, the mean of the number of steps taken by the walker to cover the entire L celled lattice. We refer to this time as residence time and denote by T_{mean}. To get T_{mean}, we calculate the mean number of visits made by the walker between any two transient states in C_2

For two transient states $i, j \in C_2$, let $m_{i,j}$ denote the mean number of visits made to j given that the visitor has started at i. We can think of $m_{i,j}$ as the local residence time from state i to j. By using probabilistic conditioning on the initial state we get,

$$\begin{aligned} m_{i,j} &= \delta_{i,j} + p_{i,i-1}\, m_{i-1,j} + p_{i,i+1}\, m_{i+1,j}, \\ &= \delta_{i,j} + \sum_{k=i}^{L-1} p_{i,k}\, m_{k,j}. \end{aligned} \tag{2.3}$$

where $\delta_{i,j}$ is a Kronecker's delta function. $\delta_{i,j} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$.

Consider M, Q and I as matrices of size $L \times L$. The matrix entries of M and Q satisfy $M_{i,j} = m_{i,j}$ and $Q_{i,j} = p_{i,j}$ for, $i, j \in C_2$. I is an identity matrix. Then we can write (2.3) in a matrix form as

$$M = I + QM. \tag{2.4}$$

It is equivalent to say that

$$\begin{aligned} (I - Q)M &= I \\ (I - Q)^{-1}(I - Q)M &= (I - Q)^{-1} \\ M &= (I - Q)^{-1}. \end{aligned}$$

Note that the determinant of $(I - Q)$ is non-zero because it is composed of block matrices of non-zero determinant. $(I - Q)$ is an invertible matrix. We use M to obtain T_m as,

$$T_m = \sum_{k=0}^{L-1} m_{0,k}. \tag{2.5}$$

For given values of L, L_v and ε we can compute the matrices Q and M with a simple program of sparse linear algebra. When we have M, T_m is simply computed as the row sum of the first row of M.

2.2.2 Estimating the residence time as a gambler's ruin time

We refer the reader to [11] for a basic introduction to the concept of gambler's ruin problem. In a gambler's ruin problem, we consider a gambler who at each play of the game wins 1 unit with a probability p and loses a unit with a probability $q = 1 - p$. In the game, assume that the successive plays of the game are independent. For a player starting with z units, we ask what is the expected number of plays such that the player will either reach a fortune of N units or 0 units. Let this expected number of plays be denoted by D then,

$$D = \begin{cases} z(N-z) & \text{when } p = q \\ \frac{z}{q-p} - \frac{N}{q-p} \cdot \frac{1-(q/p)^z}{1-(q/p)^N} & \text{when } p \neq q. \end{cases} \tag{2.6}$$

Consider the game played by the gambler here as a random walk. The event of winning would correspond to a movement by a step by $+1$ cell units with a probability p and the event of losing would correspond to a movement by -1 cell units with a probability q.

The results from the gambler's ruin problem cannot be applied directly to the pedestrian problem in focus because the success probability changes depending upon the position of the pedestrian. The one dimensional problem might as well be characterized as a combination of two gamblers' ruin problems. The first one is with $N = 2(L - L_v)$ units, $z = (L - L_v)$ and $p = 0.5$. The second problem is given with the values $N = 2L_v$ units, $z = L_v$ units, and $p = 0.5 + \varepsilon$. Let the expected duration of the game for these first and the second problems be denoted by T_1 and T_2. We have

$$T_1 = (L - L_v)^2. \tag{2.7}$$

$$T_2 = \frac{L_v}{\varepsilon}\left(\frac{(0.5+\varepsilon)^{L_v} - (0.5-\varepsilon)^{L_v}}{(0.5\varepsilon)^{L_v} + (0.5-\varepsilon)^{L_v}}\right). \tag{2.8}$$

We estimate the total time taken the pedestrian to cover the entire stretch as

$$T_m = T_1 + T_2. \tag{2.9}$$

2.3 Simulation results

The simulation is realized with an inspiration from cellular automata algorithms (see, for instance, [25] for more information). Check Appendix A for the numerical scheme used to obtain the results.

We have set the corridor length to $L = 100$ m. Each step taken by the pedestrian, denoted by Δx, is of magnitude 1 m. Each time step is of magnitude $\Delta t = 1$ sec. The mean velocity (V_m) of the pedestrian is estimated by

$$V_m = \frac{L\ \Delta x}{T_m \Delta t}[m/s]. \tag{2.10}$$

We know that $\varepsilon \in (0, 0.5)$. We choose ε from the set $\{0.01, 0.1, 0.2, 0.3, 0.4\}$. For a fixed value of $\varepsilon \in \{0.01, 0.1, 0.2, 0.3, 0.4\}$, we vary $L_v \in [0, 100]$ and find the value of V_m from the simulation.

We compare it with the formulae obtained for T_m in (2.5) and (2.9).

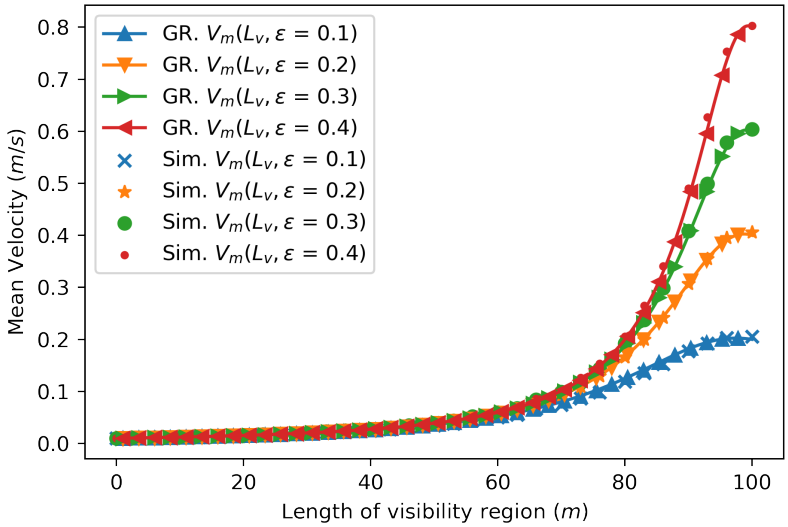


Figure 2.3.1: Comparison of the estimates for V_m from the gambler's ruin formulation and the simulation.

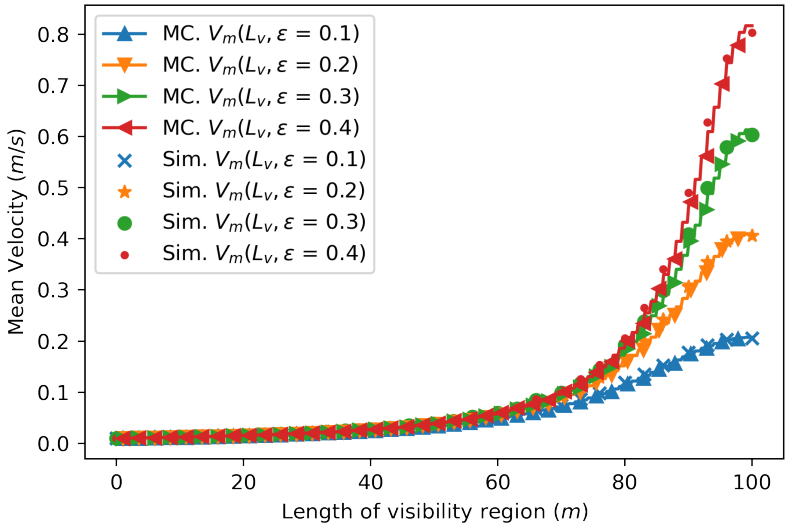


Figure 2.3.2: Comparison of the estimates for V_m from Markov chain formulation and simulation.

In Figures 2.3.1, 2.3.2 and 2.3.3 the labels "Sim. $V_m(L_v, \varepsilon)$, GR. $V_m(L_v, \varepsilon)$, and MC. $V_m(L_v, \varepsilon)$" refer to the V_m calculated from the simulation, the Markov chain formulation and the gambler's ruin formulation respectively.

In Figures 2.3.1 and 2.3.2, we observe that both the formulations provide decent approximations for V_m when compared to the results from the simulation. We compare the effectiveness of both the formulations when $\varepsilon = 0.01$. In Figure 2.3.3, for the same values of L and L_v, we plot the measurement of V_m from the Markov chain and gambler's ruin formulations

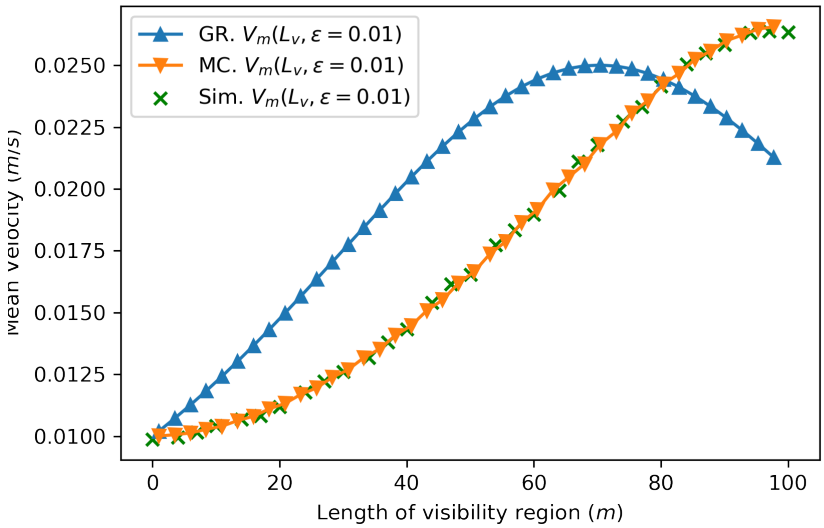


Figure 2.3.3: Comparison of simulation results for V_m and the estimates from the Markov chain and gambler's ruin formulations when $\varepsilon = 0.01$

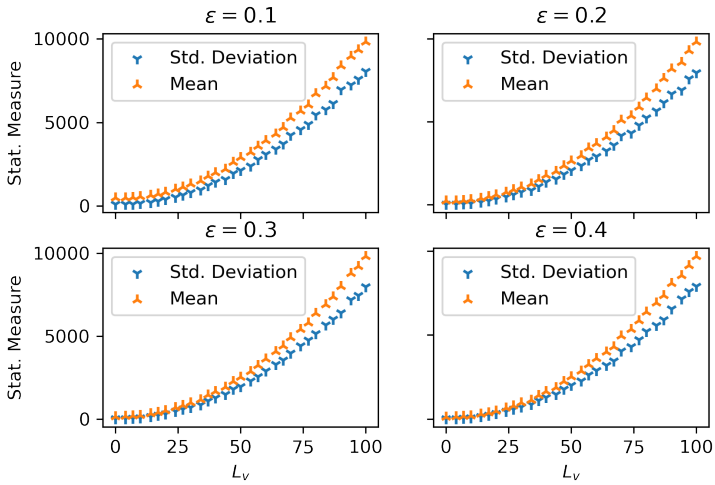


Figure 2.3.4: The standard deviation and the mean of T_{mean} from simulation results. versus L_v.

when $\varepsilon = 0.01$, . We find that the Markov chain formulation approximates the simulation results better than the gambler's ruin formulation.

In Figure 2.3.4 we present you with the profile of the mean and standard deviation of the residence time measurements from the simulation. We used 10^4 realizations to calculate these results. The mean and standard deviation give provide us with the information about the distribution of all the measured values of T_{mean}.

2.4 Discussion

In Figures 2.3.2 and 2.3.2, we can notice that the mean velocity of the pedestrian increases as the length of the visibility region increases. Furthermore, for a fixed length of the visibility region, the higher is the bias ε, the higher is the mean velocity. These observations agree with our intuition because we expect the mean velocity to increase with an increase in the length of the visibility region or the bias. We can compare the merits of both the formulations using these results.

The gambler's ruin formulation could have been sufficient to obtain a formula for results observed from simulation. But, the formula in (2.9) fails for very low values of ε. We found the profiles of V_m from (2.9) and the simulation grossly differed when $\varepsilon = 0.01$ (see Figure 2.3.3). We understand that it is not necessary for the moving-in-the-dark scenario here to be equivalent to the combination of two gamblers' ruin problems. The behaviour of the pedestrian is ambiguous at the interface between the visibility and the dark regions. We may not see it for higher values of ε that is when $\varepsilon \in \{0.1, 0.2, 0.3, 0.4\}$. We found the ambiguity to be very evident when $\varepsilon = 0.01$.

To get a better modelling perspective we reformulated the problem as a Markov chain process. The Markov chain formulation allows for the calculation of mean time spent in transient states. This formulation has approximated the V_m in a better way because it does not fail to provide the correct mean velocity estimate when $\varepsilon = 0.01$. The disadvantage is that it does not have a nice analytical form as in the case of the ruin formulation. We do not know the orders of growth with respect to the variables involved. But it is heartening that the gambler's ruin works at higher values of ε.

Chapter 3

Modelling crowd crush at a T-junction

3.1 Introduction

A very unfortunate crowd crush incident took place on the night of 29 October 2022 in the Itaewon neighbourhood of Seoul, Korea[1]. In about 30 minutes, 159 people died and 196 were injured. It took place at a region where people were navigating through narrow shopping streets. On one side of the junction there were crowds coming from the nearest metro station going to the social events. On the other side there were people going to the metro station. When people started walking towards a T-junction from all three directions, there occurred a jam and eventually a crowd crush, resulting in many injuries and deaths. Here, we propose a stochastic model that helps in investigating the existence of a critical density which differentiates between a free flow of pedestrians and a pedestrian jam.

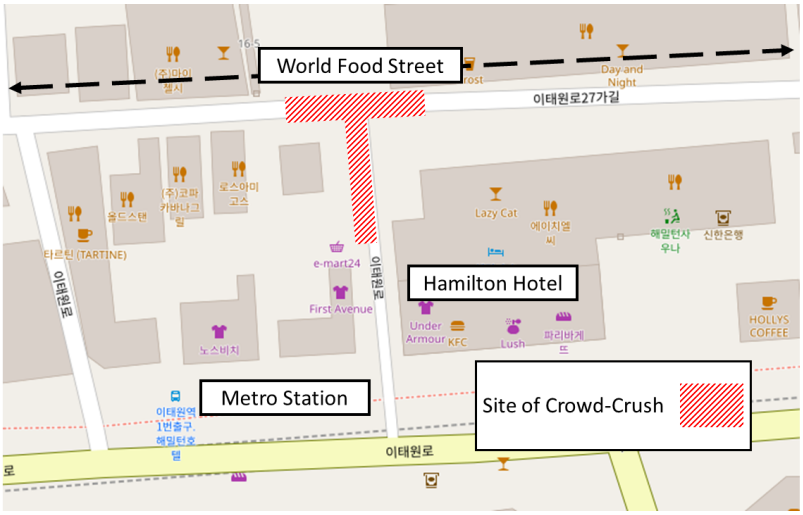


Figure 3.1.1: A snapshot of the streets of Itaewon neighbourhood in Seoul where the crowd crush took place.[2]

[02]The map is obtained from the open source repository OsMap. URL: `https://www.osmap.asia/#19/37.53474/126.99339`

A map of Itaewon and an illustration of the site of crowd crush is shown in Figure 3.1.1. Itaewon is a popular for late-night shopping and social events in Seoul. This was one of the first large scale public event after the relaxation of COVID-19 norms that gathered tens of thousands of young people. In Figure 3.1.1, we see that the crowds can come from the direction of the metro station and the world food street. The crowds from the metro station were eager to get to the world food street. Likewise, the crowds from world food street were keen to go on the direction of the metro station. Meanwhile, some places like clubs situated in the streets were hosting certain social events. When the crowds built up in the streets, they triggered a crowd crush. The events that followed had aggravated the crush, resulting in injuries and deaths.

3.2 Modelling approach: Rule-set for the discrete dynamics

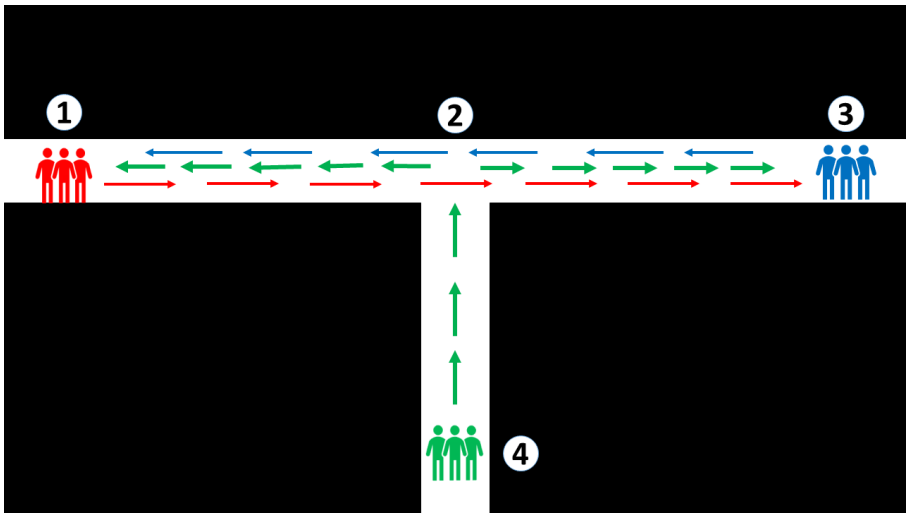


Figure 3.2.1: Sketch of the initial setting of the problem.

We propose a simple stochastic model that is similar to a lattice-gas model. We consider the walking space as a collection of cells. Before beginning the dynamics, we set a threshold on the number of pedestrians occupying a cell. This threshold, denoted by ρ_l, is defined as the maximum local density. At each time step one of the pedestrians is picked at random. The picked pedestrian is moved by a cell in the pedestrian's preferred direction, only if the number of pedestrians at the destination is less than ρ_l. If the destination cell has a ρ_l pedestrians, we do not allow the pedestrian to move.

We consider three distinguishable population sets identified by red, blue and green colours. Each population set is characterized by the set's preferred direction of movement. Let ①,②,③ and ④ denote the locations as labelled in Figure 3.2.1. The objectives of the population sets are as follows.

- The reds begin from ① and want to exit the walking space from the location ③.
- The blues begin from ③ and want to exit the walking space from ①.

[01] We used a report by the Wall Street Journal. URL `https://www.washingtonpost.com/investigations/2022/11/16/seoul-crowd-crush-itaewon-victims/`

- The greens, begin from ④, and move in the direction of ②. Upon reaching ② they decide to either go towards ① or ③ with equal probability.

With this model, we aim to answer the following questions.

(Q1). Does a critical density of pedestrians exists that demarcates the formation of a jam?

(Q2). What is the typical time taken for the jam to occur?

To address the questions *(Q1)* and *(Q2)*, we implement the model as a stochastic simulation. The stochastic simulation is understood better with the help of an algorithm.

3.2.1 Algorithm for the discrete dynamics

Let Λ denote the walking space of the pedestrians. We discretize the domain Λ as a finite collection of points. We set Λ to be

$$\Lambda := \{(0,0),(0,1),\dots(0,L),(\tfrac{L}{2},-1),(\tfrac{L}{2},-2),\dots,(\tfrac{L}{2},-\tfrac{L}{3})\}. \tag{3.1}$$

We set the value of L as a multiple of 6 such that $\Lambda \subset \mathbb{Z}\times\mathbb{Z}$. We denote the population size of each species of pedestrians by k, where $k \in \mathbb{N}$. We denote the positions of all the pedestrians $P_R, P_G, P_B \in (\mathbb{Z}\times\mathbb{Z})^k$ and define by

$$\begin{aligned} P_R &:= (r_1, r_2, \dots, r_k),\\ P_G &:= (g_1, g_2, \dots, g_k),\\ P_B &:= (b_1, b_2, \dots, b_k). \end{aligned}$$

For $m,n \in \mathbb{N}$, consider $\vec{x} \in (\mathbb{Z}\times\mathbb{Z})^m, \vec{y} \in (\mathbb{Z}\times\mathbb{Z})^n$. Let $\vec{x} := (x_1, x_2, \dots, x_m), \vec{y} := (y_1, y_2, \dots, y_n)$., where $x_1, x_2, \dots x_n, y_1, y_2, \dots, y_n \in (\mathbb{Z}\times\mathbb{Z})$ We define the concatenation operator, denoted by '$\frown$', as a mapping from $(\mathbb{Z}\times\mathbb{Z})^m \times (\mathbb{Z}\times\mathbb{Z})^n$ to $(\mathbb{Z}\times\mathbb{Z})^{m+n}$. We define $\vec{x} \frown \vec{y}$ as

$$\vec{x} \frown \vec{y} := (x_1, x_2, \dots, x_m, y_1, y_2, \dots, y_n),$$

where we call $\vec{x} \frown \vec{y}$ as the concatenation of $\vec{x}$ and $\vec{y}$.

Now consider a large position vector $\Pi \in (\mathbb{Z}\times\mathbb{Z})^{3k}$ that is built by concatenating P_R, P_G and P_B. Let Π be defined by

$$\Pi := P_R \frown P_G \frown P_B = (r_1, r_2, \dots, r_k, g_1, g_2, \dots, g_k, b_1, b_2, \dots, b_k).$$

Π is a list of the positions of all the pedestrians irrespective of the classification. Let Π_i denote the i^{th} coordinate of the vector Π. We define a function $C(x)$ to count the number of pedestrians at a given point in the domain. The number of pedestrians at a point $p \in \Lambda$ is given by the function $C : \Lambda \to \mathbb{N}$ as,

$$C(p) = \sum_{i=1}^{|\Lambda|} \delta(p, \Pi_i). \tag{3.2}$$

Here $|\Lambda|$ denotes the cardinality of the set Λ. $\delta(x,y)$ is a Kronecker delta function. For a fixed

i, $\delta(p, \Pi_i) = \begin{cases} 1 \text{ if } p = \Pi_i \\ 0 \text{ if } p \neq \Pi_i \end{cases}$.

Let $n \in \mathbb{N}$ denote the n^{th} time step of the simulation.

The Algorithm

Step 0. At $n = 0$, set

$$P_R = \big((0,0),(0,0),\dots(0,0)\big)$$
$$P_B = \big((L,0),(L,0),\dots(L,0)\big)$$
$$P_G = \big((\tfrac{L}{2},-\tfrac{L}{3}),(\tfrac{L}{2},-\tfrac{L}{3}),\dots,(\tfrac{L}{2},-\tfrac{L}{3})\big).$$

Generate

$$D_G = (X_i)_{i=1}^{i=k} = (X_1, X_2, \dots, X_k). \tag{3.3}$$

Here each X_i is a random variable taking the values either 0 or 1 with an equal probability. We generate D_G to determine the direction of a green pedestrian when the pedestrian has reached the position $(\frac{L}{2}, 0)$.

Step 1. Update n to $n + 1$. Now concatenate the vectors P_R, P_G and P_B as

$$\Pi = P_R \frown P_G \frown P_B. \tag{3.4}$$

Recall that Π indicates a current state of the positions occupied by pedestrians

Step 2. Randomly select one of the pedestrians. Let the selected position of the pedestrian be denoted by $p = (p_x, p_y)$. Clearly, we observe $p \in \Pi$.

Step 2A. If $p \in P_R$, let $r' = (p_x + 1, p_y)$.

If $r' \neq (L, 0)$, then count the number of pedestrians at r' using $C(r')$

- If $C(r') < \rho_l$, then update the position as $p = r'$.
- If $C(r') = \rho_l$, then do not update the position.

Else If $r' = (L, 0)$, the selected red pedestrian is considered to have reached the desired destination. Now delete p from P_R.

Step 2B. If $p \in P_B$, let $b' = (p_x - 1, p_y)$.

If $b' \neq (0, 0)$, then count the number of pedestrians at b' using $C(b')$.

- If $C(b') < \rho_l$, then update the position as $p = b'$.
- If $C(b') = \rho_l$, then do not update the position.

Else if $g' = (0, 0)$, then the selected blue pedestrian is considered to have reached the desired destination. Now delete p from P_G.

Step 2C. If $p \in P_B$, consider one of the following accordingly

1. If $p_y < 0$, then $g' = (p_x, p_y + 1)$. Now count the number of pedestrians at g' using $C(g')$
   - If $C(g') < \rho_l$, then update the position as $p = g'$.
   - If $C(g') = \rho_l$, then do not update the position. Ready for the next step.

2. If $p_y = 0$ and $p_x = \frac{L}{2}$, then check the direction $X_i \in D_G$ of this selected green p given in **Step 0**. Let the selected green be denoted by g_i.
   - If $X_i = 0$, then the direction of g_i is towards $(L, 0)$. We treat the walker as a red walker and then follow the same rule as in **Step 2A.**
   - If $X_i = 1$, then the direction of g_i is towards $(0, 0)$. We treat the walker as a green walker and then follow the same rule as in **Step 2B.**
3. If $p_y = 0$ and $p_x \in \{1, L-1\}$, then delete p from P_G. Ready for the next step.

Step 3. This completes an action for one time step. If $n < T_M$ then go to **Step 1.** and repeat the same process. If $n = T_M$, we end the dynamics.

This finishes the stochastic simulation of the crowd dynamics. After the completion, we find the dimension of the Π vector as the number of remnant pedestrians. Let the final density be denoted by ρ_F.

$$\rho_F = \frac{N_f}{3k}, \tag{3.5}$$

where N_f denotes the final number of remnant pedestrians after the completion of the dynamics.We can think of N_f as the dimension of the vector Π.

We investigate if there exists a $T_c \in \{1, 2, 3, \ldots, T_M\}$ such that the vector Π observes no change at all for every $n \in \{T_c, T_c + 1, T_c + 2, \ldots T_M\}$. Recall that Π denotes a vector of the positions of all the pedestrians at a given time step n. By saying Π not observing a change for all $n \in \{T_c, T_c + 1, T_c + 2, \ldots T_M\}$, we neither observe an update in any pedestrian's position nor a deletion of a pedestrian from P_R, P_G and P_B for all these values of n. This implies a steady pedestrian traffic state for every time step beginning from T_c till the end T_M. Such a steady traffic state may indicate either a total clearance of the pedestrians from the walking domain or a traffic-jam.

Note that the algorithm of dynamics is random in nature. So we are interested in the mean value of ρ_F and T_c. If one implementation of the algorithm from $n = 0$ to $n = T_M$ of the accomplishes one realization, we reiterate the algorithm for numerous realizations and calculate the mean values of ρ_F and T_c.

3.3 Simulation output

To fix the parameters, we set $L = 12$, $T_M = 10^4$, $\rho_l \in A := \{3, 6, 9\}$. We vary the initial number of pedestrians k from 1 to 50 in increments of 1. Thus $k \in B := \{1, 2, 3, \ldots, 50\}$. The simulation is carried out for 500 realizations and each experiment has taken place for 10^4 time steps. The program has taken around 128 min to run on a personal computer with an octa core processor[1].

The computation of ρ_F follows directly from the algorithm. But it is not the case with T_c. We do not know if there exists a value of T_c for all values of possible values $L, T_M, k \in \mathbb{N}$. But for the set values of L, T_M and k, we find that a value of T_c exists without a problem. We find this

[1]The program is implemented in Python, and we used a code optimization tool called Numba. URL: https://numba.pydata.org/

through exhaustion, that is by implementing the algorithm multiple time for all chosen values of the parameters.

Let the mean values of ρ_F and T_c be denoted by $E(\rho_F)$ and $E(T_c)$. We gather all the results and report the following plots:

- For a given $\rho_l \in A$, we plot $E(\rho_F)$ for every $k \in B$.
- For a given $\rho_l \in A$, we plot $E(T_c)$ for every $k \in B$.
- For a given $\rho_l \in A$, we plot $\frac{E(T_c)}{3k}$ for every $k \in B$.

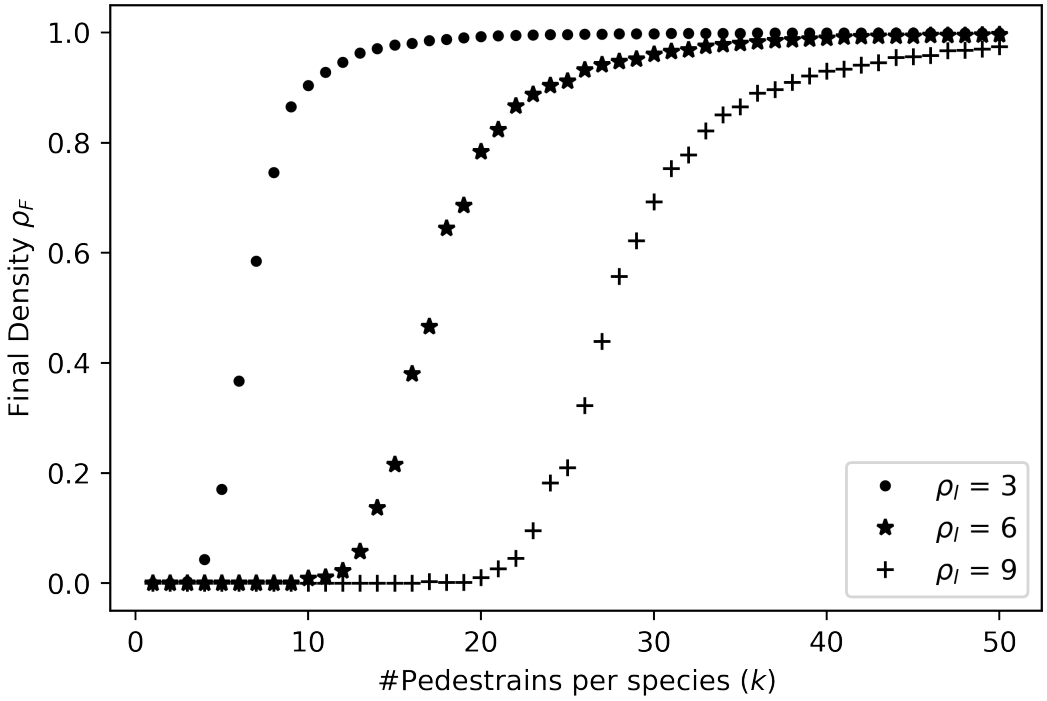


Figure 3.3.1: Behaviour of $E(\rho_F)$ vs. k, for a given ρ_l

In Figure 3.3.1, we see the that $E(\rho_F)$ is plotted as a function of ρ_l and k. For a fixed value of $\rho_l \in A$, we find $E(\rho_F)$ for all values of $k \in B$. Observe that ρ_F is close to zero until a value of for certain lower values of $k \in B$ and close to one for certain later values of $k \in B$. At the intermediary values of $k \in B$ the value of ρ_F can be observed to be increasing.

In Figure 3.3.2, we plot the $E(T_c)$ as a function of k and ρ_l. Here too, for a fixed value of $\rho_l \in A$, we find $E(T_c)$, for all values of $k \in B$. Observe that the profile of T_c is close to a linearly increasing profile with k except for certain intermediary values of k. The increasing behaviour of $E(T_c)$ with respect to k is fairly intuitive because it is the time taken by the entire pedestrian population to arrive at a steady traffic state. But there is an interesting change in the shape of the profile at the intermediary values of k. So we shall scale $E(T_c)$ down by the total number of pedestrians to check $E(T_c)$ per pedestrian. Therefore, in Figure 3.3.3, we plot $\frac{E(T_c)}{3k}$ as a function of k and ρ_l.

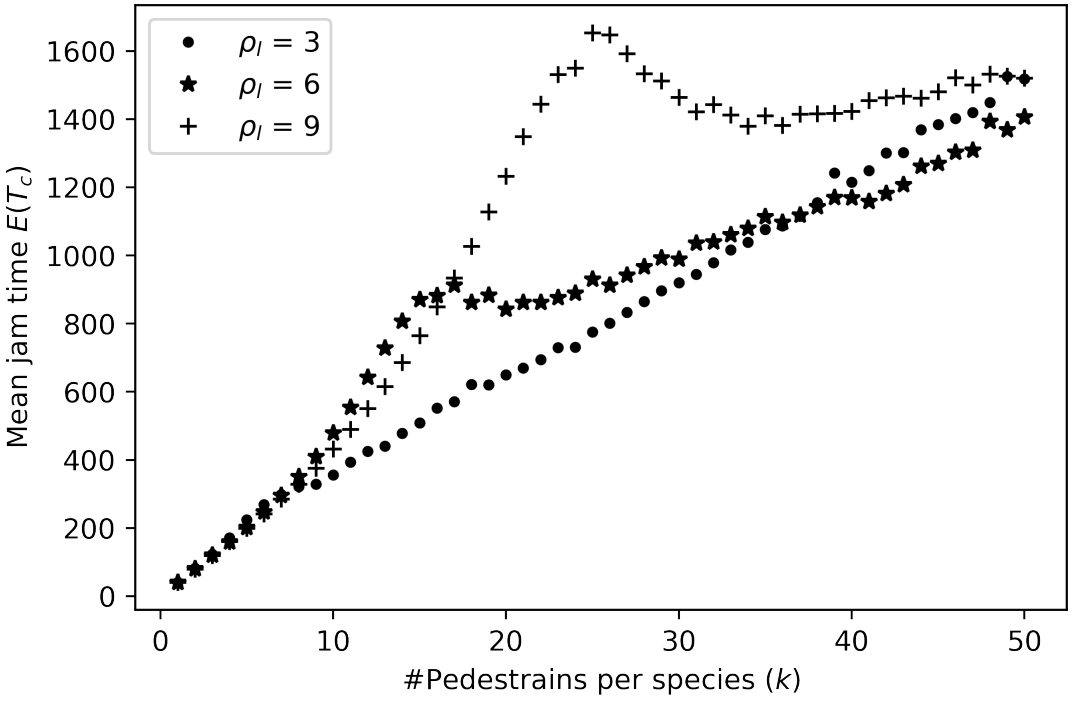


Figure 3.3.2: Behaviour of $E(T_c)$ vs. k, for a given ρ_l

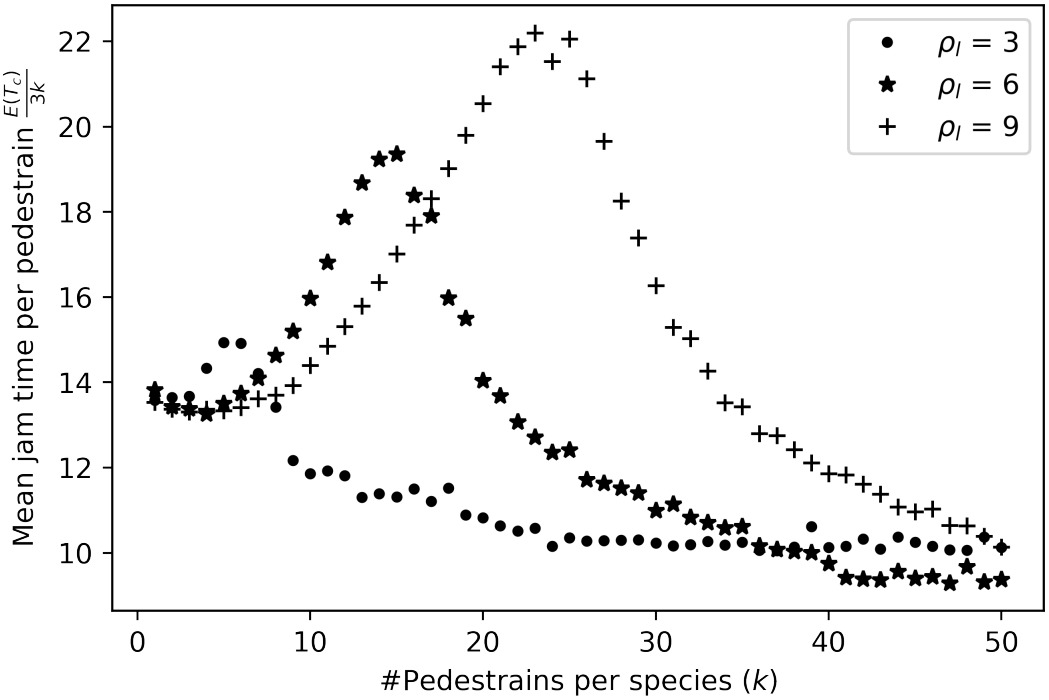


Figure 3.3.3: Behaviour of $\frac{E(T_c)}{3k}$ vs. k for a given ρ_l

In Figure 3.3.3, we observe that $\frac{E(T_c)}{3k}$ attains a maximum at a particular value of $k \in B$. For a given $\rho_l \in A$, we define a $k_c(\rho_l) \in B$ such that

$$E(T_c(k_c, \rho_l)) = \max_{k \in B} E(T_c(k, \rho_l)). \tag{3.6}$$

Observe that, $E(\rho_F(k_c, \rho_l))$ is a non-zero value in Figure 3.3.1. Thus, $k_c(\rho_F)$ happens to be a critical value where the $\frac{E(T_c)}{3k}$ peaks. We suspect this shift in the behaviour of the traffic is a phase transition and $k = k_c$ acts as a tipping point.

3.4 Discussion

To answer the questions we posed at the beginning of this chapter.

(A1) According to this model, there exists an initial number of pedestrians k_c after which the pedestrian jam happens to occur.

(A2) We can estimate a mean time taken by the pedestrians to form a jam. We interestingly observe that the mean time per pedestrian to arrive at a jam, $\frac{E(T_c)}{3k}$ is maximum at the value k_c

Besides being simplistic, this model detects certain tipping points and trends that allow us to make some comments on handling the pedestrian traffic at junctions. We observe that the model detects a direct correspondence between the formation of a jam formation and ρ_l. The higher the ρ_l is, the larger the initial number of pedestrians needed for the formation of a jam (see, for instance, Figure (3.3.1)).

For a given initial number of pedestrians, the time taken for the occurrence of steady traffic increases when ρ_l increases. Likewise, for a fixed ρ_l, the time taken for a clearance or a jam increases when the initial number of pedestrians is increased (see, for instance, Figure (3.3.2)). The correspondence between $E(T_c)$ and k is straightforward because we expect the time taken for a jam formation to increase when the initial number of pedestrians is increasing. To disable a natural linear increase of $E(T_c)$ with respect to k, we scale it down by $3k$ – the total initial number of pedestrians. We find out the behaviour of $\frac{E(T_c)}{3k}$ vs. k in Figure 3.3.3. We observe that $\frac{E(T_c)}{3k}$ observes a maximum for a value of $k_c(\rho_l) \in B$

It is still not clear if ρ_F varies continuously with respect to k or if there exists a discrete discontinuity. It is possible for ρ_F to shift discretely at a particular value of k, (possibly k_c). If such a critical value exists, we may say that the formation of jam occurs as a phase transition. We would like to further investigate such properties of the model.

Chapter 4

Pedestrian dynamics using a parabolic PDE formulation

4.1 Introduction

In Chapter 3, we have seen a discrete particle dynamics description of pedestrians at a T-junction. However, a continuum description of the dynamics in the same scenarios is desired to understand high density pedestrian flows. At high densities, models of discrete dynamics may fail because the dynamics of a crowd are generally affected by certain macroscopic effects with much more significance than individual interactions. One way forward is to model dynamic crowds of high densities as moving fluids. We may not be able to give a microscopic description of the dynamics, but we may comment on the dynamics at a macroscopic level. Thus, in such desired continuum models, we give a PDE depicting a kinetic description of the pedestrian density distribution (see, for instance, [19, Chapter 1]).

As the T-junction structure is a complicated geometry to handle, we begin with a simple one dimensional domain of length L. In this chapter, we describe a convection-diffusion equation to model the pedestrian dynamics on a 1-D domain. With this equation, we aim to capture an essence rather than an exact imitation of the dynamics. We construct a random walk scheme to solve the model equation. We then compare the random walk approximation with a finite difference approximation of the same problem. Additionally, we derive a gradient random walk scheme to solve the model equation and present the numerical simulation results. Our main aim is to prove the convergence results of the random walk approximation schemes.

4.2 Problem 1

In this section, we propose a simple diffusion-convection equation to be governing the dynamics of pedestrians. We impose a Dirichlet boundary condition at the left boundary, to introduce a constant number of pedestrians at every time step. At the right boundary, we impose a Robin boundary condition to capture the flux of both incoming and outgoing pedestrians.

Figure 4.2.1: A visual representation of the domain

The model equation is described as follows: Find $u(t,x)$ satisfying

$$u_t = \partial_x(Du_x - (L-x)u) \qquad (x,t) \in (0,L)\times(0,T) \tag{4.1}$$

$$u(0,x) = u_0(x) = (u_R - 5)e^{-\frac{\gamma}{D}x} + 5 \qquad x \in (0,L) \tag{4.2}$$

$$u(t,0) = u_R \qquad t \in (0,T) \tag{4.3}$$

$$-Du_x(t,L) = \gamma(u-5) \qquad t \in (0,T). \tag{4.4}$$

Here, $u(t,x)$ denotes the density of pedestrians at $(t,x) \in [0,T]\times[0,L]$, where L and $T \in (0,\infty)$ are lengths of the space and time domains respectively. $D \in (0,1)$ denotes the diffusion coefficient. Let $\Omega = (0,L)\times(0,T)$

In (4.3), $u_R \in [0,\infty)$ signifies the constant density of pedestrians at the left boundary of the domain, present at all times. We expect that the pedestrians can only enter the domain from the left boundary but cannot exit. At the right boundary, we have a Robin boundary condition with $\gamma > 0$. Ideally, we want the left boundary to be both an exit and an entry point for the pedestrians. Since we are only dealing with a single population species, $\gamma 5$ in (4.3) represents the maximum density of external pedestrians entering the domain from the right boundary.

In (4.2), $u_0(x)$ describes the initial density profile of the pedestrians. When $t = 0$ we expect a maximum number of pedestrians at the left boundary and a minimum at the right boundary. We want the initial condition to be a decreasing function from $x = 0$ to $x = L$. Therefore, we chose an exponential function with a negative power that also satisfies the boundary conditions. In Figure 4.2.1 we can notice a graphic representation of the walking domain of the problem.

We refer to the set of equations from (4.1) to (4.4) as Problem 1. Note that the initial condition $u_0(x)$ belongs to $L^2(0,L)$. Problem 1 is a well-posed PDE problem with a unique solution for $u(x,t) \in H^1(\Omega)$. We refer the reader to the book [10, Chapter 7], for the proof of the well-posedness of the problem.

4.3 Finite difference approximation scheme

In this section, we examine a finite difference method numerical scheme for solving Problem 1.

To discretize the time and space domains, let K represent the length of the time step and h denote the length of the space mesh, where $h, K \in (0,1)$. We use the notation $u_j^n := u(t_n, x_j)$,

with $x_j := jh$ and $t_n := nK$. Additionally, let $N_h := \frac{L}{h}$ and $N_K := \frac{T}{K}$ be the number of divisions along the space and time dimensions, respectively. We select values for h and K that ensure N_h and N_K are integers, thereby determining the number of divisions in the space and time domains.

We tackle the problem by using the simplest finite difference scheme, the forward Euler and central difference method. The algorithm begins with setting the initial condition, followed by the step-by-step implementation of the diffusion-convection equation by finite differences. The scheme is stable and converges when $\frac{K}{h^2} \leq \frac{1}{2}$.(cf. [16, Chapter 9])

4.3.1 The algorithm

Step 0. Setting the initial condition. For $n = 0$, we take

$$u_j^0 = u_0(x_j).$$

Step 1. We update n to $n+1$. To implement the Robin's boundary condition we define a 'Ghost site' as below

$$u_{N_h+1}^n := u_{N_h}^n - \frac{h*\gamma}{D}(u_{N_h}^n - 5).$$

Step 2. In this step, we discretize each of the differential operators in (4.1) and combine then term wise. For $j \in \{1, 2, \ldots, N_h - 1\}$, compute the terms A_1, A_2 and A_3 as defined by

$$A_1 := \left(\frac{DK}{h^2} - K\frac{L-x_j}{2h}\right)u_{j-1}^n,$$
$$A_2 := \left(1 - \frac{2DK}{h^2} - K\right)u_j^n,$$
$$A_3 := \left(\frac{DK}{h^2} + K\frac{L-x_j}{2h}\right)u_{j+1}^n.$$

$$u_j^{n+1} = A_1 + A_2 + A_3.$$

Step 3. To handle the Dirichlet boundary condition, take

$$u_0^{n+1} = u_R.$$

To handle the Robin boundary condition, take

$$A_1' := \left(\frac{DK}{h^2} - K\frac{L-x_j}{2h}\right)u_{N_h-1}^n,$$
$$A_2' := \left(1 - \frac{2DK}{h^2} - K\right)u_{N_h}^n,$$
$$A_3' := \left(\frac{DK}{h^2} + K\frac{L-x_j}{2h}\right)u_{N_h+1}^n.$$

$$u_{N_h}^{n+1} = A_1' + A_2' + A_3'.$$

Step 4. If $nK \leq T$, then and go to **Step 1.**

When $n = N_K$, we finish the simulation, and we obtain the finite difference approximation for $u(x, t)$ in Problem 1.

4.4 Random walk approximation scheme

In this section, we propose a numerical scheme that is based on random walk methods. Taking inspiration from a similar method outlined in [17], we construct the scheme relevant to Problem 1. The scheme is realized through a fractional step method (see, for instance, [23, Chapter 8]). An implementation of the fractional step scheme involves splitting the differential operators. In this case, we split the diffusion and convection operators. Before initiating the dynamics, the value of the pedestrian density is discretized into a collection of particles. In the first fraction of the time step, these particles are diffused as per the diffusion operator. Later, we apply drift to the newly diffused particles as per the convection operator.

For more information about approximating continuum fields with random walks, we refer the reader to the monograph [26] where details are worked out for the case of diffusion in random fields with direct applications to chemical reactions and transport in groundwater.

We maintain the same notation and discretization as discussed in the finite difference scheme in Section 4.3.1 except for u_j^n. For the random walk approximation, we implement the scheme and obtain multiple approximations for u_j^n. We take the mean of all such u_j^n this obtained as the solution approximation. Thus, $E(u_j^n)$ denotes the random walk approximation, $E(u_j^n) := u(t_n, x_j)$. The algorithm is described as the following,

4.4.1 The algorithm

Step 0. Setting the initial conditions. For $n = 0$, set

$$u_j^0 = u_0(x_j).$$

Step 1. Update n to $n + 1$. Setting the boundary conditions. For $t = nK$ and $j = 0$, set

$$u_0^n = u_R.$$

We define a *'Ghost site'* u_{j+1}^n to implement the Robin's boundary condition

$$u_x(nK, L) = \frac{u_{N_h+1}^n - u_{N_h}^n}{h} = -\tfrac{\gamma}{D}(u_j^n - 5),$$
$$u_{N_h+1}^n = u_{N_h}^n - \tfrac{h*\gamma}{D}(u_{N_h}^n - 5).$$

Step 2. The diffusion is implemented by diffusing 'pedestrian particles' into the domain. We discretize the value of pedestrian density at an x_j as a collection of particles. To do it, we choose an $\alpha \in (0, 1)$ to discretize u_j^n into smaller particles. For all $j \in \{1, 2, \dots N_h\}$ there exists a N_j such that

$$N_j - 1 < \frac{|u_j^n|}{\alpha} \leq N_j,$$
$$k_j^n := \frac{u_j^n}{N_j}, \tag{4.5}$$

where N_j denotes the number of pedestrian 'particles' at x_j and k_j^n denotes the mass of each such particle.

Step 3. The diffusion process is initiated here. At each x_j we place N_j particles with mass k_j^n. For each l^{th} particle placed at x_j, we associate a random variable X_l^j taking values $-1, 0,$ and 1 with probabilities $\frac{DK}{h^2}, 1-\frac{2DK}{h^2},$ and $\frac{DK}{h^2}$ respectively.

- If $X_l^j = 1$ the particle jumps from x_j to x_{j+1}.
- If $X_l^j = -1$ the particle jumps from x_j to x_{j-1}.
- if $X_l^j = 0$ the particle stays at x_j.

Step 4. For $j \in \{1, \ldots, N_h - 1\}$, we define

$$U_j^{n+1} := k_{j-1}^n \sum_{l=1}^{N_{j-1}} I_1(X_l^{j-1}) + k_j^n \sum_{l=1}^{N_j} I_0(X_l^j) + k_{j+1}^n \sum_{l=1}^{N_{j+1}} I_{-1}(X_l^j). \tag{4.6}$$

For $j \in \{0, N_h\}$, we have

$$U_0^{n+1} := k_0^n \sum_{l=1}^{N_0} I_0(X_l^0) + k_1^n \sum_{l=1}^{N_1} I_{-1}(X_l^1), \tag{4.7}$$

$$U_{N_h}^{n+1} := k_{N_h-1}^n \sum_{l=1}^{N_{N_h-1}} I_1(X_l^{N_h-1}) + k_{N_h}^n \sum_{l=1}^{N_{N_h}} I_0(X_l^{N_h}). \tag{4.8}$$

Where $I_i(x)$ denotes an indicator function. $I_i(x) = 1$ if $x = i$ and 0 otherwise. Note that U_j^{n+1} is the number of particles at x_j after the random walk. x_j has the particles coming from x_{j+1} and x_{j-1} and the particles that remained at x_j. This step executes the diffusion part of (4.1).

Step 5. For $j \in \{1, 2 \ldots N_h - 1\}$, take

$$\begin{aligned} u_j^{n+1} &= U_j^{n+1} + K\frac{(L-x_{j+1})U_{j+1}^{n+1} - (L-x_j)U_j^{n+1}}{h} \\ &= k_1(h)U_j^{n+1} + k_2(h)U_{j+1}^{n+1} \end{aligned} \tag{4.9}$$

$$\text{where } k_1(h) := \left(1 - \frac{K(L-x_j)}{h}\right), k_2(h) := \frac{K(L-x_{j+1})}{h}. \tag{4.10}$$

With this step we execute the convection part of (4.1). To handle the boundary conditions, consider for $j \in \{0, N_h\}$ the following:

$$\begin{aligned} u_0^{n+1} &= u_R. \\ u_{N_h}^{n+1} &= \tfrac{DK}{h^2} u_{N_h-1} + \left(1 - \tfrac{2DK}{h^2} - K\right) u_{N_h} + \tfrac{DK}{h^2} u_{N_h+1}. \end{aligned}$$

Step 6. If $nK \leq T$, then update n to $n+1$ and go to **Step 2.**. If $nK = T - 1$, then update n to T, obtain u_j^T for all $j \in \{1, 2, \ldots, N_h\}$.

Using the above algorithm we essentially approximate the smooth solution $u(x,t)$ to the PDE Problem 1.

4.4.2 Convergence of the random walk numerical scheme

In this section, we study at the convergence of the random walk approximation scheme (defined previously) to a corresponding fractional-step finite difference approximation scheme. Our main result is stated in the following theorem.

Theorem 4.4.1. *Observe that $u_0 \in C^1(0, L)$. Let $0 < h, K < 1$ with $\frac{K}{h^2} \leq \frac{1}{2}$ then there exists a constant M independent of h and K such that,*

$$|E(u_j^n) - u(t_n, x_j)| \leq Mh^2. \tag{4.11}$$

Lemma 4.4.2. *Let u_j^n, U_j^n be defined as in the algorithm. Then it holds*

$$E(U_j^{n+1}) = \frac{DK}{h^2}E(u_{j-1}^n) + \Big(1 - \frac{2DK}{h^2}\Big)E(u_j^n) + \frac{DK}{h^2}E(u_{j+1}^n), \tag{4.12}$$

$$E(u_j^{n+1}) = k_1(h)E(U_j^{n+1}) + k_2(h)E(U_{j+1}^{n+1}). \tag{4.13}$$

Proof. The proof of (4.12) and (4.13) directly follow from the proof given in [17]. A sketch of the proof is as follows:

- Firstly, we prove Lemma 4.4.2.
- Secondly, we use Lemma 4.4.2 to find a bound on the difference between $E(u_j^n)$ and u_j^n.
- Finally, we use this result to prove Theorem 4.4.1.

We begin with proving (4.12). Recall the definition k_j^n from (4.5). From the definition of U_j^{n+1} in (4.6) we have

$$\begin{aligned}
E(U_j^{n+1}) &= E\Big(k_{j-1}^n \sum_{l=1}^{N_{j-1}} I_1(X_l^{j-1})\Big) + E\Big(k_j^n \sum_{l=1}^{N_j} I_0(X_l^j)\Big) + E\Big(k_{j+1}^n \sum_{l=1}^{N_{j+1}} I_{-1}(X_l^j)\Big) \\
&= E\big(\tfrac{u_{j-1}^n}{N_{j-1}}\big)E\big(\sum_{l=1}^{N_{j-1}} I_1(X_l^{j-1})\big) + E\big(\tfrac{u_j^n}{N_j}\big)E\big(\sum_{l=1}^{N_j} I_0(X_l^j)\big) \qquad (4.14) \\
&\qquad + E\big(\tfrac{u_{j+1}^n}{N_{j+1}}\big)E\big(\sum_{l=1}^{N_{j+1}} I_{-1}(X_l^{j+1})\big) \\
&= \tfrac{E(u_{j-1}^n)}{N_{j-1}} \sum_{l=1}^{N_{j-1}} E\big(I_1(X_l^{j-1})\big) + \tfrac{E(u_j^n)}{N_j} \sum_{l=1}^{N_j} E\big(I_0(X_l^j)\big) \qquad (4.15) \\
&\qquad + \tfrac{E(u_{j+1}^n)}{N_{j+1}} \sum_{l=1}^{N_{j+1}} E\big(I_{-1}(X_l^{j+1})\big) \\
&= \frac{DK}{h^2}E(u_{j-1}^n) + \Big(1 - \frac{2DK}{h^2}\Big)E(u_j^n) + \frac{DK}{h^2}E(u_{j+1}^n). \qquad (4.16)
\end{aligned}$$

Note that

$$E(I_1(X_l^{(j)})) = E(I_{-1}(X_l^{(j)})) = \frac{DK}{h^2}, \tag{4.17}$$

$$E(I_0(X_l^{(j)})) = 1 - 2\frac{DK}{h^2}. \tag{4.18}$$

This proves (4.12).

It is easy to see that (4.13) holds true as well. By the application of the mean operator on (4.9), it reduces to

$$\begin{aligned} E\big(u_j^{n+1}\big) &= E\Big(k_1(h)U_j + k_2(h)U_{j+1}\Big) \\ &= k_1(h)E(U_j) + k_2(h)E(U_{j+1}). \end{aligned} \tag{4.19}$$

Proof of (4.4.1). By substituting (4.13) into (4.12), we have

$$\begin{aligned} E(u_j^{n+1}) = k_1(h)\big(&\tfrac{DK}{h^2}E(u_{j+1}^n) + (1 - \tfrac{2DK}{h^2})E(u_j^n) + \tfrac{DK}{h^2}E(u_{j-1}^n)\big) \\ + k_2(h)\big(&\tfrac{DK}{h^2}E(u_{j+2}^n) + (1 - \tfrac{2DK}{h^2})E(u_{j+1}^n) + \tfrac{DK}{h^2}E(u_j^n)\big). \end{aligned} \tag{4.20}$$

From the fractional-step finite difference scheme proposed in the articles [17], [13] and the book [23], we have the following scheme

$$\begin{aligned} (\Phi_j^{n+1}) = \big(&\tfrac{D}{2}\Phi_{j+1}^n + (1-D)\Phi_j^n + \tfrac{D}{2}\Phi_{j-1}^n\big) \\ + K\big(&\tfrac{D}{2}g(\Phi_{j+1}^n, (\tfrac{\partial\Phi}{\partial x})_{j+1}^n) + (1-D)g(\Phi_j^n, (\tfrac{\partial\Phi}{\partial x})_j^n) + \tfrac{D}{2}g(\Phi_{j-1}^n, (\tfrac{\partial\Phi}{\partial x})_{j-1}^n)\big), \end{aligned} \tag{4.21}$$

where Φ_j^n is the numerical approximate solution to $u(x_j, t_n)$ from the finite difference method. We find

$$\begin{aligned} (\Phi_j^{n+1}) = \big(&\tfrac{D}{2}\Phi_{j+1}^n + (1-D)\Phi_j^n + \tfrac{D}{2}\Phi_{j-1}^n\big) \\ &+ K\tfrac{L-jh}{h}\big(\tfrac{D}{2}(\Phi_{j+2}^n - \Phi_{j+1}^n) + (1-D)(\Phi_{j+1}^n - \Phi_j^n) + \tfrac{D}{2}(\Phi_j^n - \Phi_{j-1}^n))\big) \\ &- K\big(\tfrac{D}{2}\Phi_{j+1}^n + (1-D)\Phi_j^n + \tfrac{D}{2}\Phi_{j-1}^n\big) \\ = k_1\big(&\tfrac{D}{2}\Phi_{j+1}^n + (1-D)\Phi_j^n + \tfrac{D}{2}\Phi_{j-1}^n\big) + k_2\big(\tfrac{D}{2}\Phi_{j+2}^n + (1-D)\Phi_{j+1}^n + \tfrac{D}{2}\Phi_j^n\big). \end{aligned} \tag{4.22}$$

Let $L_n = \{1, 2, \dots N_h\}$ and set

$$\sigma^n := \max_{j \in L_n} |E(u_j^n) - \Phi_j^n|. \tag{4.23}$$

Taking the maximum over the difference between (4.20) and (4.22) yields

$$\begin{aligned} \sigma^{n+1} &\le k_1\sigma^n + k_2\sigma^n \le (k_1 + k_2)\sigma^n \\ &\le (1-K)\sigma^n \le (1-K)^2\sigma^{n-1} \\ &\le (1-K)^{n+1}\sigma^0 = 0. \end{aligned} \tag{4.24}$$

We note that

$$\sigma^0 := \max_j |E(u_j^0) - \Phi_j^0| = 0. \tag{4.25}$$

We observe that

$$|E(u_j^n) - \Phi_j^n| \leq \max_{j \in L_n} |E(u_j^n) - \Phi_j^n| \leq 0 \tag{4.26}$$

From the theory of finite difference methods, see for instance [23, Chapter 8], we find that,

$$|\Phi_j^n - u(t_m, x_j)| \leq C_3 K + C_4 D h^2, \tag{4.27}$$

for some constants $C_3, C_4 \in (0, \infty)$ independent of h and K.
Combining (4.26) with (4.27) yields,

$$\begin{aligned} |E(u_j^n) - u(t_n, x_j)| &\leq |E(u_j^n) - \Phi_j^n| + |\Phi_j^n - u(t_n, x_j)|, \\ &\leq C_3 K + C_4 D h^2, \leq \Big(\tfrac{C_3}{2} + C_4 D\Big) h^2. \end{aligned}$$

□

This proves Theorem 4.4.1 The inequality (4.11) gives a bound on the error difference solely dependent on h. The result makes it clear that for a finer discretization of the domain, we will have a better approximation of $u(x, t)$.

4.5 A gradient random walk approximation scheme

While the proposed random walk approximation in Section 4.4 converges to the corresponding finite difference approximation, it remains uncertain whether it is possible to establish a h dependent bound on the statistical variance of the approximation thus obtained. A general technique that helps us in achieving a bound on the variance of the approximation is to work with a gradient random walk scheme. In a gradient random walk, we approximate the gradient of the solution by a random walk method. By using this scheme, we can find a h dependent bound on the error and the variance of the approximation. We refer the reader to [12] for more information simulating diffusion using random walk methods.

We are unable to construct a gradient random walk scheme for Problem 1 because of the non-zero boundary conditions. However, we would like to demonstrate the convergence results of it. Therefore, we propose Problem 2, which shares the same parabolic equation as given in Problem 1 but possesses zero boundary conditions and an appropriate initial condition. We present a gradient random walk approximation scheme for Problem 2, showcasing its convergence of both the numerical approximation and variance.

4.5.1 Problem 2.

The modified problem with homogeneous boundary conditions reads as follows.

$$u_t = \partial_x(Du_x + (L-x)u) \qquad x \in (0,L), \quad t \in (0,T) \tag{4.28}$$

$$u(0,x) = u_0(x) = 3\sin\left(\tfrac{5\pi x}{2L}\right)\exp\left(\tfrac{-Lx}{2D}\right) \qquad x \in (0,L) \tag{4.29}$$

$$u(t,0) = 0 \qquad t \in (0,T) \tag{4.30}$$

$$u_x(t,L) = 0 \qquad t \in (0,T). \tag{4.31}$$

In the above equations, $u(t,x)$ denotes the density of the pedestrians at $t,x \in [0,T]\times[0,L]$, where L,T denote the lengths of the time and space domains, and $L,T \in [,\infty)$. D denotes the diffusion coefficient, and $D \in (0,1)$. At the left boundary of the domain, we have a zero Dirichlet boundary condition and at the right boundary we have a zero Neumann boundary condition.

Note that at $x = 0$, the pedestrians cannot exit the domain from the boundary. It observes a 0 pedestrian density for all $t \in [0,T]$. The right boundary is an open boundary with a zero Neumann boundary condition. At $x = L$, the number of pedestrians going out of the domain is equal to the number of pedestrians walking into the domain. Adhering to the two boundary conditions, we propose the initial condition $u_0(x)$ in (4.29). Note that $u_0(0) = 0$ and $\frac{du_0}{dx}(L) = 0$. We recall that the main goal is to solve the problem numerically with a gradient random walk scheme and prove the convergence results.

We refer to (4.28)-(4.29) as the Problem-2.

We may rewrite (4.28) as

$$\begin{aligned} u_t &= Du_{xx} + (L-x)u_x - u \\ &= Du_{xx} + g(u,u_x), \\ \text{with } g\big(u,u_x\big) &= (L-x)u_x - u. \end{aligned} \tag{4.32}$$

This simplification of (4.28) is useful in the analysis and the algorithm. As the gradient random walk algorithm is devised in [17], we will use a fractional step method in this case as well. We follow the same discretization notation as discussed in the finite difference scheme section.

4.5.2 The algorithm

Step 0. Set the boundary conditions. For $n \in \{0,1,\ldots N_K\}$, we set

$$u(t_n,0) = 0 \qquad\qquad -Du_x(t_n,L) = 0.$$

Step 1. Set the initial condition. For $j = 0,1,\ldots N_h$ and $n = 0$, it holds

$$u_j^0 = u(0,x_j) = u_0(x_j). \tag{4.33}$$

Step 2. For $j \in \{1, \dots N_h\}$, it holds

$$\begin{aligned} u_j^n &= u_j^n + Kg(u_j^n, (\tfrac{\partial u}{\partial x})_j^n) \\ &= u_j^n + K((L - jh)(\tfrac{\partial u}{\partial x})_j^n - u_j^n) \\ &= (1 - K)u_j^n + K(L - jh)\frac{u_{j+1}^n - u_j^n}{h} \\ &= (1 - K - \tfrac{K(L-jh)}{h})u_j^n + K(L - jh)u_{j+1}^n \\ &= k_1 u_j^n + k_2 u_{j+1}^n, \end{aligned} \tag{4.34}$$

where $$k_1 := k_1(K, h) = (1 - K - \tfrac{K(L-jh)}{h}) \quad k_2 := k_2(h) = \tfrac{K(L-jh)}{h}. \tag{4.35}$$

Step 3. We compute the following numerical differentiation. Define

$$\xi_j^n := \xi(t_n, x_j) = \frac{\nu_j^n - \nu_{j+1}^n}{h} \text{ for } j = 1, \dots N_h - 1. \tag{4.36}$$

$$\text{Set } \xi_0^n = \xi_{N_h}^n = 0.$$

Step 4. An arbitrary choice of $\alpha \in (0, 1)$ has to be made. So let $\alpha := K^2$. There exists a N_j such that

$$N_j - 1 < \frac{|h\xi_j^n|}{\alpha} \leq N_j, \tag{4.37}$$

$$k_j^n := \frac{h\xi_j^n}{N_j}. \tag{4.38}$$

Step 5. We initiate the diffusion process here. At each x_j we place N_j particles with mass $k_j^n = \frac{h\xi_j^n}{N_j}$. For each particle placed at x_j, we associate a random variable X taking values $-1, 0,$ and 1 with probabilities $-\frac{D}{2}, 1 - D,$ and $\frac{D}{2}$, respectively.

- If $X_l^j = 1$, the particle jumps from x_j to x_{j+1}.
- If $X_l^j = -1$, the particle jumps from x_j to x_{j-1}
- if $X_l^j = 0$, the particle stays at x_j

Step 6. Let

$$N_j k_j^{n+1} = hL_j^{n+1} = \sum_{l=1}^{N_{j-1}} I_1(X_l^{j-1})k_{j-1}^n + \sum_{l=1}^{N_j} I_0(X_l^j)k_j^n + \sum_{l=1}^{N_{j+1}} I_{-1}(X_l^j)k_{j+1}^n. \tag{4.39}$$

$$u_j^{n+1} = \sum_{l \geq j} hL_j^{n+1}. \tag{4.40}$$

Where $I_i(x)$ denotes an indicator function. $I_i(x) = 1$ if $x = i$ and 0 otherwise. L_j^{n+1} approximates the gradient of $u(x_j, t_n)$. (4.40) is the numerical integration of hL_j^{n+1} to obtain u_j^{n+1}.

Step 7. If $nK \leq T$, then update n to $n + 1$ and go to **Step 2**.

This essentially approximates $u(x, t)$ in Problem 2. In the above procedure, **Step 2-3** represent the convection part of (4.28), and **Step 4 - 6** represent the diffusion part. To diffuse the gradient of the pedestrian density, we split the value into discrete particles and execute the

random walk. This step is helpful in estimating the variance of u_j^n. We prove the convergence of the scheme and a bound on the variance in the next section.

4.5.3 Convergence of the gradient random walk scheme

In this section we prove the convergence of the algorithm discussed in Section 4.5.2. The main result is stated in Theorem 4.5.1.

Theorem 4.5.1. . *For $u_0 \in C^1(0, L)$, let $h, K \in (0, 1)$ with $\frac{K}{h^2} = \frac{1}{2}$. Then there exists a constant M independent of h and K such that,*

$$|E(u_j^n) - u(t_n, x_j)| \leq Mh, \tag{4.41}$$

$$Var(u_j^n) \leq Mh. \tag{4.42}$$

To prove this Theorem 4.5.1, we need the following auxiliary Lemma.

Lemma 4.5.2. *Let u_j^n, v_j^n be defined as in the algorithm from Section 4.5.2*

$$E(u_j^{n+1}) = \frac{D}{2}E(v_{j+1}^n) + (1 - D)E(v_j^n) + \frac{D}{2}E(v_{j-1}^n) \tag{4.43}$$

$$\begin{aligned} Var(u_j^{n+1}) \leq (1 + 3K)\bigg(\frac{D}{2}Var(v_{j+1}^n) + (1 - D)Var(v_j^n) + \frac{D}{2}Var(v_{j-1}^n)\bigg) \\ + 10KD(E|v_{j-1}^n - v_j^n| + E|v_j^n - v_{j+1}^n|) \end{aligned} \tag{4.44}$$

$$E(v_j^n) = k_1 E(u_j^n) + k_2 E(u_{j+1}^n) \tag{4.45}$$

$$Var(v_j^n) \leq (1 + KL^2)\big(Var(u_j^n) + Var(u_{j+1}^n)\big). \tag{4.46}$$

Proof. The proof of (4.43) and (4.44) follow from the paper [17] directly. We begin with proving

(4.43). From the definition of u_j^{n+1} in (4.40), we have

$$\begin{aligned}
u_j^{n+1} &= \sum_{l\geq j} hL_l^{n+1} \\
&= \sum_{l\geq j}\Big\{\sum_{i=1}^{N_{l-1}} I_1(X_i^{(l-1)})k_{l-1}^n + \sum_{i=1}^{N_l} I_0(X_i^{(l)})k_l^n + \sum_{i=1}^{N_{l+1}} I_1(X_i^{(l+1)})k_{l+1}^n\Big\} \\
&= \sum_{l_1+1\geq j}\sum_{i=1}^{N_{l_1}} I_1(X_i^{l_1})k_{l_1}^n + \sum_{l_1\geq j}\sum_{i=1}^{N_{l_1}} I_0(X_i^{l_1})k_{l_1}^n + \sum_{l_1-1\geq j}\sum_{i=1}^{N_{l_1}} I_1(X_i^{l_1})k_{l_1}^n \\
&= \sum_{i=1}^{N_{j-1}} I_1(X_i^{(j-1)})k_{j-1}^n - \sum_{i=1}^{N_j} I_{-1}(X_i^{(j)})k_j^n \\
&\qquad + \sum_{l_1\geq j}\sum_{i=1}^{N_{l_1}}[I_1(X_i^{(l_1)}) + I_0(X_i^{(l_1)}) + I_{-1}(X_i^{(l_1)})]k_{l_1}^n \\
&= \sum_{i=1}^{N_{j-1}} I_1(X_i^{(j-1)})k_{j-1}^n - \sum_{i=1}^{N_j} I_{-1}(X_i^{(j)})k_j^n + \sum_{j\geq j} h\xi_l^n \\
&= \nu_j^n + \sum_{i=1}^{N_{j-1}} I_1(X_i^{(j-1)})k_{j-1}^n - \sum_{i=1}^{N_j} I_{-1}(X_i^{(j)})k_j^n \\
&= \nu_j^n + \sum_{i=1}^{N_{j-1}} I_1(X_i^{(j-1)})k_{j-1}^n - N_j k_j^n + \sum_{i=1}^{N_j} I_1(X_i^{(j)})k_j^n + \sum_{i=1}^{N_j} I_0(X_i^{(j)})k_j^n \\
&= v_j^n - (v_j^n - v_{j+1}^n) + \mathcal{D}_{j-1}^+ + \mathcal{D}_j^0 + \mathcal{D}_j^+ +,
\end{aligned}$$

$$\begin{aligned}
&\tfrac{D}{2}(v_{j-1}^n - v_j^n) + (1-D)(v_j^n - v_{j+1}^n) + \tfrac{D}{2}(v_j - v_{j+1}^n) \\
&= \frac{D}{2}v_{j+1}^n + (1-D)v_j^n + \frac{D}{2}v_{j-1}^n + \mathcal{D}_{j-1}^+ + \mathcal{D}_j^0 + \mathcal{D}_j^+, \qquad (4.47)
\end{aligned}$$

where D_j^0 and D_j^+ are defined as follows

$$\mathcal{D}_j^0 := \sum_{l=1}^{N_j}[I_0(X_l^{(j)}) - E(I_0(X_l^{(j)}))]\frac{v_j^n - v_{j+1}^n}{N_j}. \qquad (4.48)$$

$$\mathcal{D}_j^+ := \sum_{l=1}^{N_j}[I_1(X_l^{(j)}) - E(I_1(X_l^{(j)}))]\frac{v_j^n - v_{j+1}^n}{N_j}. \qquad (4.49)$$

Note that

$$E(I_1(X_l^{(j)})) = E(I_{-1}(X_l^{(j)})) = \frac{D}{2}. \qquad E(I_0(X_l^{(j)})) = 1 - D. \qquad (4.50)$$

Proceeding as in the proof of Lemma 1 in [17], when we take the mean operator on (4.47) we get (4.43). Similarly, upon acting the variance operator on (u_j^{n+1}) we obtain (4.44)
Now, we prove (4.45).
Consider (4.34). By acting the mean operator on this equation we get (4.45) because k_1 and k_2

are constants that do not depend on a realization of the simulation. Recall (4.34)

$$v_j^n = k_1 u_j^n + k_2 u_{j+1}^n.$$

By applying the mean operator, we get

$$E(v_j)^n = k_1 E(u_j^n) + k_2 E(u_{j+1}^n).$$

Now consider acting the Variance operator on (4.34)

$$\begin{aligned} Var(v_j^n) &= k_1^2 Var(u_j^n) + k_2^2 Var(u_{j+1}^n) \\ &\leq (k_1^2 + k_2^2)\big(Var(u_j^n) + Var(u_{j+1}^n)\big). \end{aligned} \tag{4.51}$$

Recall k_1, k_2 from (4.35), that $h, K \in (0,1)$, and $h^2 = 2K$ from the finite difference method. Now consider

$$\begin{aligned} k_1^2 + k_2^2 &= (1-K)^2 + 2\tfrac{K^2(L-jh)^2}{h^2} - 2(1-K)\tfrac{K(L-jh)}{h} \\ &\leq (1-K)^2 + 2\tfrac{K^2(L-jh)^2}{h^2} \\ &\leq 1 + 2K^2 \frac{(L-jh)^2}{2K} \\ &\leq 1 + K(L-jh)^2 \\ &\leq 1 + KL^2 = k_3. \end{aligned} \tag{4.52}$$

Therefore, using (4.52) in (4.51) we get

$$\begin{aligned} Var(v_j^n) &= (k_1^2 + k_2^2)\big(Var(u_j^n) + Var(u_{j+1}^n)\big) \\ &\leq (1 + \tfrac{L^2}{h^2})\big(Var(u_j^n) + Var(u_{j+1}^n)\big) \\ &= k_3\big(Var(u_j^n) + Var(u_{j+1}^n)\big). \end{aligned} \tag{4.53}$$

This proves (4.46). Now we return to the proof of Theorem 4.5.1. We substitute (4.45) into (4.43) to get

$$\begin{aligned} E(u_j^{n+1}) = k_1\big(\tfrac{D}{2}E(u_{j+1}^n) + (1-D)E(u_j^n) + \tfrac{D}{2}E(u_{j-1}^n)\big) \\ + k_2\big(\tfrac{D}{2}E(u_{j+2}^n) + (1-D)E(u_{j+1}^n) + \tfrac{D}{2}E(u_j^n)\big). \end{aligned} \tag{4.54}$$

As discussed in [23], [13] and [17], we have the finite difference scheme specified in (4.21). We

have

$$\begin{aligned}(\Phi_j^{n+1}) &= \big(\tfrac{D}{2}\Phi_{j+1}^n + (1-D)\Phi_j^n + \tfrac{D}{2}\Phi_{j-1}^n\big) + K\big(\tfrac{D}{2}g(\Phi_{j+1}^n, (\tfrac{\partial\Phi}{\partial x})_{j+1}^n) \\ &\quad + (1-D)g(\Phi_j^n, (\tfrac{\partial\Phi}{\partial x})_j^n) + \tfrac{D}{2}g(\Phi_{j-1}^n, (\tfrac{\partial\Phi}{\partial x})_{j-1}^n)\big) \qquad (4.55)\\ &= \big(\tfrac{D}{2}\Phi_{j+1}^n + (1-D)\Phi_j^n + \tfrac{D}{2}\Phi_{j-1}^n\big) + K\tfrac{L-jh}{h}\big(\tfrac{D}{2}(\Phi_{j+2}^n - \Phi_{j+1}^n) \\ &\quad + (1-D)(\Phi_{j+1}^n - \Phi_j^n) + \tfrac{D}{2}(\Phi_j^n - \Phi_{j-1}^n))\big) \\ &\quad - K\big(\tfrac{D}{2}\Phi_{j+1}^n + (1-D)\Phi_j^n + \tfrac{D}{2}\Phi_{j-1}^n\big) \\ &= k_1\big(\tfrac{D}{2}\Phi_{j+1}^n + (1-D)\Phi_j^n + \tfrac{D}{2}\Phi_{j-1}^n\big) \\ &\quad + k_2\big(\tfrac{D}{2}\Phi_{j+2}^n + (1-D)\Phi_{j+1}^n + \tfrac{D}{2}\Phi_j^n\big). \qquad (4.56)\end{aligned}$$

From the theory of finite difference methods in [23], recall from (4.27) that

$$|\Phi_j^n - u(t_m, x_j)| < C_3K + C_4Dh^2. \qquad (4.57)$$

for some constants $C_3, C_4 \in (0, \infty)$.
Let

$$\sigma^n := \max_{j\in L_n} |E(u_j^n) - \Phi_j^n|. \qquad (4.58)$$

Then by taking the maximum over the difference between (4.54) and (4.56), we get

$$\begin{aligned}\sigma^{n+1} &\le (k_1 + k_2)\sigma^n \\ &\le (1-K)\sigma^n \\ &\le (1-K)^{n+1}\sigma^0 = 0. \qquad (4.59)\end{aligned}$$

Combining the (4.59) and (4.56), we get

$$\begin{aligned}|E(u_j^n) - u(t_n, x_j)| &\le |E(u_j^n) - \Phi_j^n| + |\Phi_j^n - u(t_n, x_j)| \\ &\le C_3K + C_4Dh^2. \qquad (4.60)\end{aligned}$$

Now to prove the condition for the convergence of variance i.e. (4.42), we substitute (4.46) into (4.44) to get

$$\begin{aligned}Var(u_j^{n+1}) \le (1+3K)k_3\Big(&\frac{D}{2}Var(v_{j+1}^n) + (1-D)Var(v_j^n) + \frac{D}{2}Var(v_{j-1}^n) \\ &+ \frac{D}{2}Var(v_{j+2}^n) + (1-D)Var(v_{j+1}^n) + \frac{D}{2}Var(v_j^n)\Big) \\ &+ 10KD(E|v_{j-1}^n - v_j^n| + E|v_j^n - v_{j+1}^n|). \qquad (4.61)\end{aligned}$$

By (4.34) and since $k_1 > 0$ and $k_2 > 0$, we may write the following inequality

$$\begin{aligned}v_j^n &= k_1u_j^n + k_2u_{j+1}^n \\ &\le (k_1 + k_2)(u_j^n + u_{j+1}^n) \\ &= (1-K)(u_j^n + u_{j+1}^n). \qquad (4.62)\end{aligned}$$

It results,

$$\begin{aligned} |v_{j-1}^n - v_j^n| &\leq (1-K)((u_{j-1}^n + u_j^n) - u_j^n - u_{j+1}^n)) \\ &= (1-K)(u_{j-1}^n - u_{j+1}^n). \end{aligned} \tag{4.63}$$

From (26), in [17], it follows directly that

$$\begin{aligned} E|u_{j-1}^{n+1} - u_j^{n+1}| &\leq \tfrac{D}{2}E(|h\xi_{j-2}^n|) + (1-D)E(|h\xi_{j-1}^n|) + \tfrac{D}{2}E(h\xi_j^n) \\ &\leq \tfrac{D}{2}E(|v_{j-2}^n - v_{j-1}^n|) + (1-D)E(|v_{j-1}^n - v_j^n|) \\ &\qquad + \tfrac{D}{2}E(|v_j^n - v_{j+1}^n|) \\ &\leq (1-K)\big(\tfrac{D}{2}E(|u_{j-2}^n - u_{j-1}^n|) + (1-D)E(|u_{j-1}^n - u_j^n|) \\ &\qquad + \tfrac{D}{2}E(|u_j^n - u_{j+1}^n|)\big). \end{aligned} \tag{4.64}$$

Combining (4.61) and (4.64) leads to

$$\begin{cases} Var(u_j^{n+1}) & \leq (1+3K)k_3\Big(\tfrac{D}{2}Var(v_{j+1}^n) + (1-D)Var(v_j^n) + \tfrac{D}{2}Var(v_{j-1}^n) \\ & \quad + \tfrac{D}{2}Var(v_{j+2}^n) + (1-D)Var(v_{j+1}^n) + \tfrac{D}{2}Var(v_j^n)\Big) \\ & \quad + 10KD(1-K)\Big(\tfrac{D}{2}E(|u_{j-2}^n - u_{j-1}^n|) \\ & \quad + (1-D)E(|u_{j-1}^n - u_j^n|) + \tfrac{D}{2}E(|u_j^n - u_{j+1}^n|) \\ & \quad + \tfrac{D}{2}E(|u_{j-1}^n - u_j^n|) + (1-D)E(|u_j^n - u_{j+1}^n|) \\ & \quad + \tfrac{D}{2}E(|u_{j+1}^n - u_{j+2}^n|)\Big). \end{cases} \tag{4.65}$$

Now define

$$\delta^n := \max_j Var(u_j^n). \qquad\qquad \eta^n := \max_j E|u_{j-1}^n - u_j^n|. \tag{4.66}$$

For $C = \max|u_0'|$, we can say that $\eta^0 \leq 2hC$. Using δ^n and η^n in (4.65) we find,

$$\eta^{n+1} \leq (1-K)\eta^n \leq \eta^n, \tag{4.67}$$

$$\delta^{n+1} \leq 2(1-K)k_3\delta^n + 20KD(1-K)\eta^n. \tag{4.68}$$

$$\begin{aligned} \frac{\delta^{n+1}}{2} &\leq k_3\delta^n + 10KD\eta^0 \\ &\leq k_3\delta^n + 10KD\eta^0 \\ &\leq (k_3)^2\delta^{n-1} + (k_3)10KD\eta^0 + 10KD\eta^0 \\ &\leq ((k_3)^n + \cdots + 1)10KD\eta^0 \\ &= \left(10KD\frac{1\cdot(k_3)^{n+1} - 1}{k_3 - 1}\right)(2hC) \\ &= 20KDC\left(\frac{(1+KL^2)^{n+1} - 1}{KL^2}\right) \\ &\leq 20DC\left(\frac{e^{TL^2} - 1}{L^2}\right)h = Mh. \end{aligned} \tag{4.69}$$

For

$$M = 20DC\left(\frac{e^{TL^2}-1}{L^2}\right). \tag{4.70}$$

Note that

$$\left(1+\frac{TL^2}{n}\right)^{n+1} \leq e^{TL^2}. \tag{4.71}$$

In (4.70) we see that M is clearly independent of K and h. Therefore, as $h \to 0$ we have error $|E(u_j^n) - u(x_n, t_n)| \to 0$. □

4.6 Numerical output and error comparison

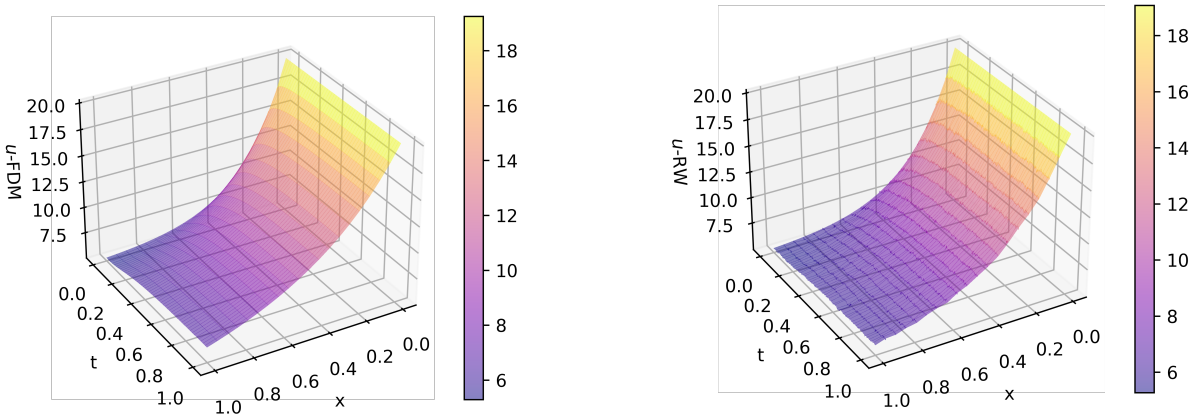
To test the algorithm and visualize the numerical approximate solutions, we have given values to all the parameters as listed in the tables below.

L	T	D	u_R	h	K	α
1	1	0.5	20	0.05	$2h^2$	10^{-3}

Table 4.6.1: Parameter values for the implementation of approximations regarding Problem 1

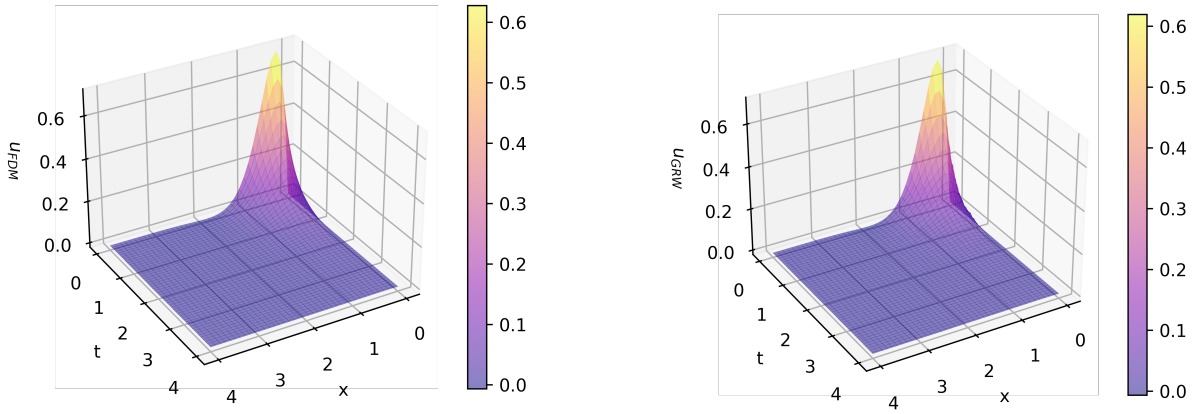
L	T	D	u_R	h	K	α
4	1	0.5	20	0.05	$2h^2$	10^{-3}

Table 4.6.2: Parameter values for the implementation of approximations regarding Problem 2.



(a) Finite-Difference approximation (b) Random walk approximation

Figure 4.6.1: Numerical output for Problem 1.



(a) Finite difference approximation (b) Gradient random walk approximation

Figure 4.6.2: Numerical output for Problem 2.

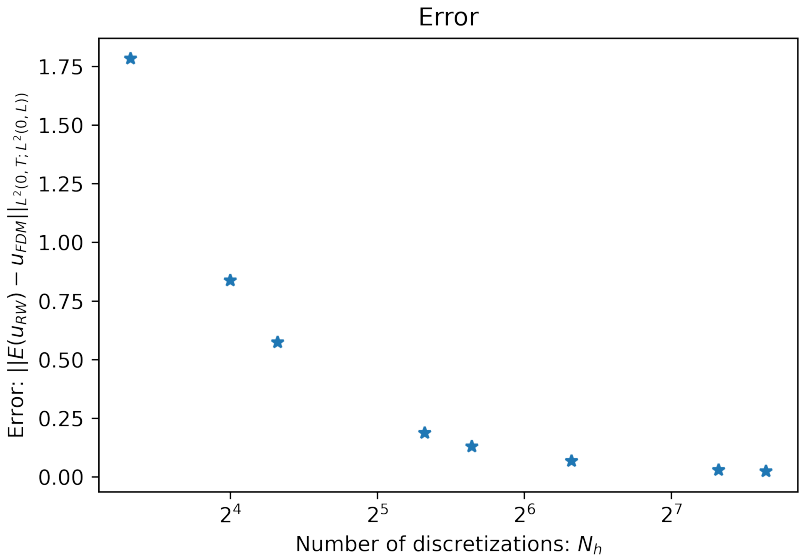


Figure 4.6.3: Difference between the finite-difference and random walk numerical approximations in $L^2(0,T;L^2(0,L))$ norm.

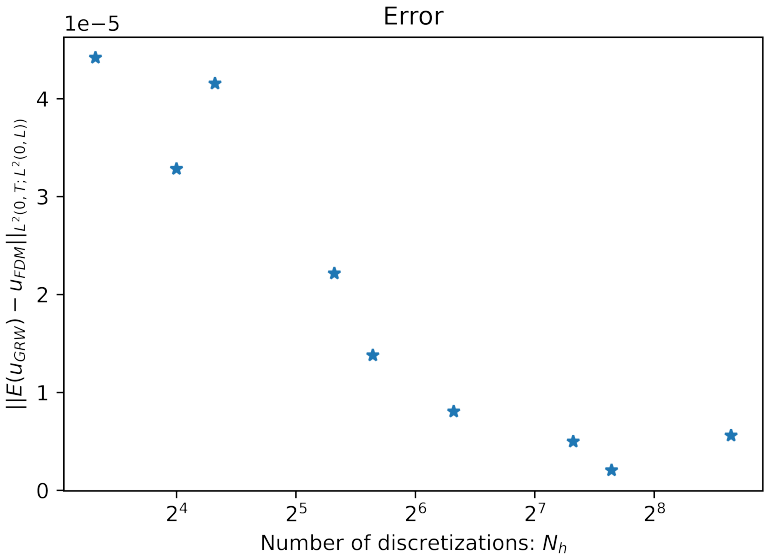


Figure 4.6.4: Difference between the finite-difference and gradient random walk approximations in $L^2(0,T;L^2(0,L))$ norm.

4.7 Discussion

We present the profile of the pedestrian density over time in Figures 4.6.1a, 4.6.2a, 4.6.1b and 4.6.2b. The partial differential equation diffuses in both the left and right directions, and drifts the pedestrians towards the right boundary. The PDEs may not be much insightful in terms of the dynamics per se. Thus, we would like to work more on determining the parameters L, D, γ and ψ that reflect realistic situations.

If u_{RW} and u_{FDM} denote the random walk and finite difference approximations, then the error difference between them in $L^2(0,T;L^2(0,L))$ norm is calculated using the formula

$$||E(u_{RW}) - u_{FDM}||_{L^2(0,T;L^2(0,L))} = \frac{\sum_{n=0}^{n=N_K}\sum_{j=0}^{j=N_h}|(E(u_{RW})_j^n - (u_{FDM})_j^n|^2}{N_h N_K}, \tag{4.72}$$

where N_h and N_K are the number of space and time discretization respectively. In Figures 4.6.3 and 4.6.4, we took a mean of 10 realizations of random walk and gradient random walk approximations to get $E(u_{RW})$ and $E(u_{GRW})$.

We want to bring your attention to the similarity between the solutions in the finite difference and the random walk algorithms. In Figures 4.6.3 and 4.6.4, we compare the difference between the random walk and finite difference approximates in L^2 norm. We observe that the error is decreasing as the step-size $h \to 0$. The decreasing profile of the error is monotonic in Figure 4.6.3 and non-monotonic in 4.6.4. But observe that the error is of the order of 10^{-5} for the gradient random walk scheme. The error is negligible when compared to the max (u_{GRW}) and max (u_{FDM}).

Observe that the error difference with respect to the gradient random walk is at the order of 10^{-3} units but the general random walk scheme is approximating at 10^{-1} units. The gradient random walk scheme is seemingly doing a better job at approximating the solution as opposed to the random walk scheme. Since we are dealing with two different problems we may not claim that it is a global behaviour. We find our motivation here to come up with a gradient random walk scheme even for Problem 1.

Chapter 5

Conclusions and future work

5.1 Bidirectional flows and network of corridors

This project has allowed us to appreciate Markov-chains and gambler's ruin problems in the context of pedestrian dynamics. In Chapter 2, the chosen problem is a simpler 1-D version of the problem investigated in [6]. However, as studied in [7], we have not considered the flow of bidirectional pedestrians. We would like to extend this study to bidirectional flows as further work on the problem.

Since our problem is in 1D, it is possible to come up with a mean-field approximation. We hope to formulate a parabolic equation to obtain the mean-field approximation. The computation of residence time using the Markov-chain formulation does not allow us to make as many analytical observations as the gambler's ruin formulation does. We may have to compare both these approaches more closely to get some insight.

We wish to expand our model in Chapter 2 to a network of connected corridors. It would be interesting to find out how long would the pedestrian(s) take to begin at a point, move through the network of corridors by stochastic decision. We hope to extend this model further along the lines suggested in [5].

5.2 Mean-field approximation for the T-junction model

We were able to present a simple model that detects interesting results as emergent crowd behaviour. We see that there is a critical behaviour of traffic around a particular initial number of pedestrians. At this value of critical pedestrian number, we then notice the formation of a jam (see, for instance, Figure 3.3.1). We also see the mean time, taken by a pedestrian to arrive at a jam, peaking at the same value of critical pedestrian number (see, for instance, Figure 3.3.3). We wish to expand the model further to investigate these observations. We wish to see if these critical values act as tipping points for crowd behaviour in this specific modelling setting. We are still not sure if the model shows a phase transition type of behaviour. It would be helpful to find out a mean-filed approximation to this stochastic model.

We still have not investigated the limitations of the model. The results presented in Chapter 3 are for a specific set of numerical values. We hope that the nature and limitations will become much clearer with a mean-filed approximation.

5.3 Gradient random walk approximation scheme for non-zero boundary conditions

We have showed how we can use random walk type methods to come up with novel numerical schemes for which we understand rigorously the convergence. A random walk scheme is an interesting approach for this problem because we can model of pedestrians as individual agents diffusing through the walking domain.

Note that the scheme first diffuses the pedestrians and then implements the convection conditions. Each time step is implemented in two fractions, in the first half we diffuse the pedestrians and in the second we apply the convection. This approach is fairly intuitive if we think of the way a crowd moves. It is subject to scrutiny despite the positive results. For the future work we want to investigate swapping of the diffusion and convection steps in the fractional step process.

The gradient random walk (GRW) approximation scheme possesses an analytical advantage. For this case, we can prove the convergence results for both the variance and the mean of the approximation. It is challenging to come up with a GRW scheme for a PDE problem when the boundary conditions are non-zero. The next step is to think of a GRW approximation scheme to solve a parabolic equation with non-zero boundary conditions.

A larger goal is also to expand the PDE problem to a T-junction and work with three populations of pedestrians flow entering the domain in three directions. It would be very interesting to come up with boundary conditions at the nodes of interaction to mimic realistic scenarios.

5.4 Further comments

All the programs are implemented in the programming language Python. The programs are openly made available and attached as appendices. We must mention and appreciate that the Python code optimizer Numba [1] has been extremely helpful in reducing the runtime of the programs.

[1] https://numba.pydata.org/

Bibliography

[1] Burstedde, C., Klauck, K., Schadschneider, A., and Zittartz, J. “Simulation of pedestrian dynamics using a 2-dimensional cellular automaton, 2001”. In: *Physica A* 295 (2002), p. 507.

[2] Chakrabarti, B.K., Chakraborti, A., and Chatterjee, A. In: *Econophysics and Sociophysics: trends and perspectives*. John Wiley & Sons, 2006.

[3] Ciallella, A., Cirillo, E.N.M., Curşeu, P.L., and Muntean, A. “Free to move or trapped in your group: Mathematical modeling of information overload and coordination in crowded populations”. In: *Mathematical Models and Methods in Applied Sciences* 28.09 (2018), pp. 1831–1856.

[4] Cirillo, E.N.M, Colangeli, M., and Muntean, A. “Effects of Communication Efficiency and Exit Capacity on Fundamental Diagrams for Pedestrian Motion in an Obscure Tunnel—A Particle System Approach”. In: *Multiscale Modeling & Simulation* 14.2 (2016), pp. 906–922.

[5] Cirillo, E.N.M., Colangeli, M., and Di Francesco, A. “Residence time in one-dimensional random walks in presence of moving defects”. In: *Probabilistic Engineering Mechanics* 69 (2022), p. 103260.

[6] Cirillo, E.N.M., Colangeli, M., Muntean, A., and Thieu, T.K.T. “When diffusion faces drift: Consequences of exclusion processes for bi-directional pedestrian flows”. In: *Physica D: Nonlinear Phenomena* (2020), p. 132651.

[7] Cirillo, E.N.M. and Muntean, A. “Dynamics of pedestrians in regions with no visibility—A lattice model without exclusion”. In: *Physica A: Statistical Mechanics and its Applications* 392 (2013), pp. 3578–3588.

[8] Corbetta, A. and Toschi, F. “Physics of human crowds”. In: *Annual Review of Condensed Matter Physics* 14.1 (2023), pp. 311–333.

[9] Craesmeyer, M. and Schadschneider, A. “Simulation of merging pedestrian streams at T-junctions”. In: *Transportation Research Procedia* 2 (2014), pp. 406–411.

[10] Evans, L. C. In: *Partial Differential Equations*. Vol. 19. American Mathematical Society, 2022.

[11] Feller, W. In: *An Introduction to Probability Theory and its Applications*. 2nd Edition. J. Wiley, 1971.

[12] Ghoniem, A. F. and Sherman, F. S. “Grid-free simulation of diffusion using random walk methods”. In: *Journal of Computational Physics* 61.1 (1985), pp. 1–37.

[13] Hald, O.H. “Convergence of random methods for a reaction-diffusion equation”. In: *SIAM Journal on Scientific and Statistical Computing* 2.1 (1981), pp. 85–94.

[14] Isobe, M., Helbing, D., and Nagatani, T. “Experiment, theory, and simulation of the evacuation of a room without visibility”. In: *Physical Review E* 69 (6 2004), p. 066132.

[15] Kirchner, A., Klüpfel, H., Nishinari, K., Schadschneider, A., and Schreckenberg, M. “Discretization effects and the influence of walking speed in cellular automata models for pedestrian dynamics”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2004.10 (2004), P10011.

[16] Larsson, S. and Thomée, V. In: *Partial Differential Equations with Numerical Methods*. Springer, 2003.

[17] Lu, W. “Convergence of a random walk method for a partial differential equation”. In: *Mathematics of computation* 67.222 (1998), pp. 593–602.

[18] Muntean, A., Cirillo, E.N.M., Krehel, O., and Böhm, M. “Pedestrians moving in the dark: Balancing measures and playing games on lattices”. In: *Collective dynamics from bacteria to crowds: an excursion through modelling, analysis and simulation* 553 (2014), pp. 75–104.

[19] Muntean, A. and Toschi, F. In: *Collective Dynamics from Bacteria to Crowds: an Excursion through Modeling, Analysis and Simulation*. Vol. 553. Springer Science & Business Media, 2014.

[20] Nagatani, T. and Nagai, R. “Statistical characteristics of evacuation without visibility in random walk model”. In: *Physica A: Statistical Mechanics and its Applications* 341 (2004), pp. 638–648.

[21] Natapov, A., Czamanski, D., and Fisher-Gewirtzman, D. “Visuospatial search in urban environment simulated by random walks”. In: *International Journal of Design Creativity and Innovation* 4.2 (2016), pp. 85–104.

[22] Richardson, O., Jalba, A., and Muntean, A. “Effects of environment knowledge in evacuation scenarios involving fire and smoke: a multiscale modelling and simulation approach”. In: *Fire technology* 55 (2019), pp. 415–436.

[23] Richtmyer, R. D. and Morton, K.W. In: *Difference Methods for Initial-value Problems*. Interscience Publishers, 1975.

[24] Ross, S. M. In: *Stochastic Processes*. John Wiley and sons, 1983.

[25] Schadschneider, A., Chowdhury, A., and Nishinari, K. In: *Stochastic Transport in Complex Systems*. Elsevier Science, 2010.

[26] Suciu, N. In: *Diffusion in Random Fields: Applications to Transport in Groundwater*. Springer, 2019.

[27] Wilensky, U. *NetLogo*. URL: https://ccl.northwestern.edu/netlogo/index.shtml.

[28] Xiong, H., Yao, L., Tan, H. W., and Wuhong. "Pedestrian walking behaviour revealed through a random walk model". In: *Discrete Dynamics in Nature and Society* 2012 (2012), pp. 1–13.

[29] Zhang, J., Klingsch, W., Schadschneider, A., and Seyfried, A. "Transitions in pedestrian fundamental diagrams of straight corridors and T-junctions". In: *Journal of Statistical Mechanics: Theory and Experiment* 2011.06 (2011), P06004.

Acknowledgements

I would like to express my sincere gratitude to my thesis advisor, Prof. dr. habil. Adrian Muntean, from the Faculty of Health, Natural Sciences, and Technology at Karlstad University. Throughout my studies at Karlstad University, Prof. Muntean has provided unparalleled academic support. His guidance has continuously helped me in achieving my goals. Prof. Muntean has allowed me to take part in many academic dialogues that took place at the department which provided me with so much inspiration to pursue a research-oriented career.

I would also like to extend my thanks to my examiner, Prof. Dr. Eddie Wadbro, for his insightful comments and constructive feedback. Dr. Wadbro has not only been an inspiring examiner for this project but also an exceptional teacher with whom I have had numerous enlightening discussions during my time at Karlstad.

Additionally, I would like to express my appreciation to Dr. Matteo Colangeli from the Department of Engineering, Information Sciences, and Mathematics at the University of L'Aquila. Dr. Colangeli encouraged me to undertake this project when I approached him to discuss my research interests. His guidance and expertise were invaluable in navigating the simulation aspects of the project.

I am grateful to my friends, Vishnu Raveendran and Surendra Nepal, who are currently pursuing their Ph.D. under the supervision of Prof. Muntean at Karlstad. They were always available and willing to assist me with any analytical and numerical inquiries related to my project.

I would also like to acknowledge Dr. Rainey Lyons for his conceptual assistance at a crucial stage, which accelerated the process of obtaining results. I would also like to thank Dr. Michael Eden for providing valuable feedback about my manuscript.

I thank Dr. Johanna Gustavsson for facilitating the visit of the RiskLab of CSR (Center for Societal Risk Research) of Karlstad University.

Last but not least, I want to express my deep appreciation to my parents and friends for their unwavering support and continuous encouragement throughout my years of study. Without their presence, this achievement would not have been possible.

Thank you once again.

MeghaShyam Veluvali

List of Figures

2.2.1 A sketch of the geometry of the partially dark corridor 11
2.2.2 A diagram representing the transition probabilities of the pedestrian. 11
2.2.3 A diagram representing the pedestrian dynamics as a Markov chain 12
2.3.1 Comparison of the estimates for V_m from the gambler's ruin formulation and the simulation. 15
2.3.2 Comparison of the estimates for V_m from Markov chain formulation and simulation. 15
2.3.3 Comparison of simulation results for V_m and the estimates from the Markov chain and gambler's ruin formulations when $\varepsilon = 0.01$ 16
2.3.4 The standard deviation and the mean of T_{mean} from simulation results. versus L_v. 16

3.1.1 A snapshot of the streets of Itaewon neighbourhood in Seoul where the crowd crush took place.[2] 18
3.2.1 Sketch of the initial setting of the problem. 19
3.3.1 Behaviour of $E(\rho_F)$ vs. k, for a given ρ_l 23
3.3.2 Behaviour of $E(T_c)$ vs. k, for a given ρ_l 24
3.3.3 Behaviour of $\frac{E(T_c)}{3k}$ vs. k for a given ρ_l 24

4.2.1 A visual representation of the domain 27
4.6.1 Numerical output for Problem 1. 42
4.6.2 Numerical output for Problem 2. 42
4.6.3 Difference between the finite-difference and random walk numerical approximations in $L^2(0,T;L^2(0,L))$ norm. 43
4.6.4 Difference between the finite-difference and gradient random walk approximations in $L^2(0,T;L^2(0,L))$ norm. 43

List of Tables

4.6.1 Parameter values for the implementation of approximations regarding Problem 1 41
4.6.2 Parameter values for the implementation of approximations regarding Problem 2. 41

Appendix A

Implementation of the moving in the dark problem

We present the code written to obtain the results in chapter 2.

```
import numpy as np
import numpy.random as r
# The command 'r.random()' generates a random number between 0 and 1
eps = 0.1  # Setting the bias value
L, L_v = 100, 30 # Setting values for L and L_v parameters
x = 1 # Initial Position
Trails = 500 # Large enough for the data
def Experiment(L, L_v, x, eps):
    t = 1
    while t > 0:
        if x == 0:
            x = 1
        if x > 0 and x < L-L_v:
        #Moves by a cell towards left or right with a 0.5 probability
            if r.random() >= 0.5:
                x = x + 1
            if r.random() < 0.5:
                x = x - 1
        if x >= L-L_v:
        #Moves by a cell towards left with 0.5 + eps probability
            if r.random() <= 0.5 + eps:
                x = x + 1
        #Moves by a cell towards right with 0.5 - eps probability
            if r.random() > 0.5 + eps:
                x = x - 1
        t = t + 1
        if x == L:
            # Time Taken = t
            break
    return t

Times = []
for i in range(Trails):
```

```
34      Times.append(Experiment(100, 30, 1, 0.1))
35
36  T_mean = np.mean(Times)
```

Listing A.1: The program is used to implement the simlation and obtain the a results of resident time and mean velocity

Appendix B

Implementation of the stochastic simulation for the T-junction problem

We present the code written to obtain the results in chapter 3. For a given number of pedestrians initially and allowed maximum of pedestrians to be occupying a site, we have the dynamics as,

```
k= 7 #Initial number of pedestrains per population
nR, nB, nG = k, k, k
L = 12 #Length of the Horizontal from location (1) to location (3)
maxT = 10**4 #Number of Time steps.
max_loc_rho = 3 #Maximum local density

def experiment(nR, nB, nG, L,max_loc_rho):
    TimeR, TimeB, TimeG = [], [], []
    #Generate a vector to keep a track of the pedestains
    posR = np.zeros(nR)
    posB = np.zeros(nB) + L
    posG = np.zeros((nG,2))
    posG[:,1] = -L/3
    posG[:,0] = 0
    #The directions for each of the greens when it reached loation (2).
    direc = np.random.randint(2, size=nG, dtype=int)
    for t in range(maxT):
#Choose a population
        grp_choice = r.randrange(3)
        pos = np.concatenate((posR, posB,posG[:,0]), axis=None)
#Update the red position
        if grp_choice == 0:
            if len(posR) > 0:
                wc0 = r.randrange(len(posR))
#Choose the walker
                p_newr = posR[wc0] + 1
                if p_newr < L:
                    #Check occupancy
```

```
                    occ = np.count_nonzero(pos == p_newr)
                    if occ < max_loc_rho:
                        posR[wc0] = p_newr
                    else:
                        None
                elif p_newr == L:
#Delete the pedestain from the array upon reaching the boundary
                    posR = np.delete(posR,wc0)

#Update a blue's position
        if grp_choice ==1:
            if len(posB) > 0:
                wc1 = r.randrange(len(posB))
#Choose the walker
                p_newb = posB[wc1] - 1
                if p_newb > 0:
                    occ = np.count_nonzero(pos == p_newb)
                    if occ < max_loc_rho:
                        posB[wc1] = p_newb
                    else:
                        None
                elif p_newb == 0:
                    posB = np.delete(posB,wc1)

#Update a Green's position
        if grp_choice == 2:
            posGx, posGy = posG[:,0], posG[:,1]
            if len(posGx) >0:
                wc2 = r.randrange(len(posGx))
#We check if the green hs been below location (2) or not
                if posGy[wc2] < -1:
                    py_newg = posGy[wc2] + 1
                    occ_y = np.count_nonzero(posGy == py_newg)
                    #print(occ_y)
                    if occ_y < max_loc_rho:
                        posGy[wc2] = py_newg
                    else:
                        None
#The following rule specifies if the positon if right below location (2).
                elif posGy[wc2] == -1:
                    py_newg, px_newg = 0, L/2
                    occ_y = np.count_nonzero(posGy == py_newg)
                    occ_x = np.count_nonzero(pos == px_newg)
                    occ = occ_x +occ_y
                    #print(occ)
                    if occ < max_loc_rho:
                        posGy[wc2] = py_newg
                        posGx[wc2] = px_newg
                    else:
                        None
#If the Green had reached the Horizontal then we assign eithwe a Red or a
    Blue behaviour.
                elif posGy[wc2] == 0:
```

```
                    if direc[wc2] == 0:
                        px_newg = posGx[wc2] + 1
                        if px_newg < L:
                            occx = np.count_nonzero(pos == px_newg)
                            #print(occ_x)
                            if occx < max_loc_rho:
                                posGx[wc2] = px_newg
                            else:
                                None
                        if px_newg == L:
                            posGx, posGy = np.delete(posGx,wc2), np.delete(
    posGy,wc2)
                            posG = np.delete(posG,(wc2),axis = 0)

                    elif direc[wc2] == 1:
                        px_newg = posGx[wc2] - 1
                        if px_newg > 0:
                            occx = np.count_nonzero(pos == px_newg)
                            #print(occx)
                            if occx < max_loc_rho:
                                posGx[wc2] = px_newg
                            else:
                                None
                        if px_newg == 0:
                            posGx, posGy = np.delete(posGx,wc2), np.delete(
    posGy,wc2)
                            posG = np.delete(posG,(wc2),axis = 0)
                posG[:,0],posG[:,1] = posGx, posGy
    lenR = len(posR)
    lenB = len(posB)
    lenG = len(posG[:,0])
    lenFinal = lenR + lenG + lenB

    return lenFinal
#We get the total number os pedestrains Remaining after running the
    experiment for large number of time steps
```

Listing B.1: This is the entire simulation program as discussed in Chapter 3.

Appendix C

Implementation of the numerical schemes for the PDE problems

We present the codes written to get the solution approximations to the PDE problems posed in chapter-4.

```
import numpy as np
import matplotlib.pyplot as plt
L, T = 1, 1
uR = 20
v = 5
gamma, D = 2, 0.5
h = 0.1
K = (h**2)/2
lambdda = (D*K)/(h**2)
prob = (2*D*K)/(h**2)
Nh, NK = int(L/h), int(T/K)
scaler = 0.001
prob_bdd = (gamma*h)/D

def lambda_x(j):
    Numerator = (L - (j*h))*K
    alpha = Numerator/h
    return alpha

def alpha(j):
    Numerator = (L - ((j)*h))
    alpha = Numerator/(2*h)
    return alpha

def u0(x):
    uu0 = (uR - v)*np.exp(-(gamma/D)*x) + v
    return uu0

def FDM_sol(h):
    K = (h**2)/2
    lambdda = (D*K)/(h**2)
    prob = (2*D*K)/(h**2)
```

```
    Nh, NK = int(L/h), int(T/K)
    prob_bdd = (gamma*h)/D
    x = np.linspace(0,L,Nh)
    u_FDM = np.zeros([NK,Nh])
    #u0 = (uR - v)*np.exp(-(gamma/D)*x) + v
    u_FDM[0,:] = u0(x)
    for tn in range(1,NK):
        u_prev = u_FDM[tn-1,:]
        u_pres = np.zeros(Nh)
        u_pres[0] = uR
        for i in range(1,Nh-1):
            A1 = (lambdda - K*alpha(i))*u_prev[i-1]
            A2 = (1-(2*lambdda)-K)*u_prev[i]
            A3 = (lambdda + K*alpha(i))*u_prev[i+1]
            u_pres[i] = A1 + A2 + A3
        robin_term = u_prev[-2] - ((2*h*gamma)/D)*(u_prev[-1] - v)
        A_Bd_1 = lambdda*u_prev[-2]
        A_Bd_2 = (1-(2*lambdda)-K)*u_prev[-1]
        A_Bd_3 = lambdda*robin_term
        u_pres[-1] = A_Bd_1 + A_Bd_2 +A_Bd_3
        u_FDM[tn,:] = u_pres
    return u_FDM
a = FDM_sol(0.1)
```

Listing C.1: The program gives a finite difference approximation to the Problem 1. in Chapter 3.

```
from numba import jit
from numba.extending import overload
import numpy as np
import matplotlib.pyplot as plt

L, T = 1, 1
uR = 20
v_bd = 5
gamma, D = 2, 0.5
scaler = 10**-3
h = 0.1
K = h**2/2
lambdda = (D*K)/(h**2)
prob = (2*D*K)/(h**2)
Nh, NK = int(L/h), int(T/K)
prob_bdd = (gamma*h)/D
probbie = np.array([prob/2,1-prob,prob/2])
probbie1 = np.array([prob_bdd,1-prob_bdd])
x = np.linspace(0,L,Nh)
u_RW = np.zeros([NK,Nh])
sol_shape = np.zeros([NK,Nh])
def k1(j):
    ppp = K*(L-j*h)/h
    pp2 = 1 - K - ppp
    return pp2

```

```
def k2(j):
    ppp = K*(L-j*h)/h
    return ppp

@overload(np.random.rand)
def rand(*size):
    if len(size) == 0:
        # Scalar output
        def rand_impl(*size):
            return np.random.random()

    else:
        # Array output
        def rand_impl(*size):
            return np.random.random(size)

    return rand_impl


@jit(noPython=True, nogil=True)
def tossing(times, prob):
    data = []
    if sum(prob) != 1:
        return print('Error')
    else:
        for i in range(times):
            r = np.random.rand()
            if r < prob[0]:
                data.append(0)
            elif r > prob[0] and r < prob[0] + prob[1]:
                data.append(1)
            elif r > prob[0] + prob[1]:
                data.append(2)
    return data


def GRW_sol(h,scaler):
    K = h**2/2
    lambdda = (D*K)/(h**2)
    prob = (2*D*K)/(h**2)
    Nh, NK = int(L/h), int(T/K)
    prob_bdd = (gamma*h)/D

    x = np.linspace(0,L,Nh)
    u_RW = np.zeros((NK,Nh))

    u_RW[0,:] = u0(x)#(uR - v_bd)*np.exp(-(gamma/D)*x) + v_bd
    for tn in range(1,NK):
        u_prev = u_RW[tn-1,:]

        robin_term = u_prev[-1] - ((2*h*gamma)/D)*(u_prev[-1] - v_bd)

        xi_tn = u_prev
```

```
80
81          Nj = xi_tn/scaler
82          Nj = np.floor(Nj)+1
83          kj = (xi_tn)/Nj
84          fwd, stay, bcd = [], [], []
85          for j in range(Nh):
86              num_par = int(Nj[j])
87              toss_up = tossing(num_par,probbie)
88              toss_up = list(toss_up)
89              heads, zero ,tales = toss_up.count(0), toss_up.count(1),
    toss_up.count(2)
90              fwd.append(heads)
91              stay.append(zero)
92              bcd.append(tales)
93
94          kkkk = int(stay[-1])
95          toss_up = tossing(kkkk,probbie1)
96          heads, tales, kkkk = toss_up.count(0), toss_up.count(1), toss_up.
    count(2)
97          hL_jn = np.zeros(Nh)
98          for j in range(1,Nh-1):
99              hL_jn[j] = fwd[j-1]*kj[j-1] + stay[j]*kj[j] + bcd[j+1]*kj[j+1]
100         hL_jn[0] =  stay[0]*kj[0] + bcd[1]*kj[1]
101         hL_jn[-1] =  fwd[-2]*kj[-2] + heads*kj[-1]
102
103         u_pres = np.zeros(Nh)
104         u_pres[0] = uR
105         for i in range(1,Nh-1):
106             u_pres[i] = hL_jn[i] + K*((L-x[i+1])*hL_jn[i+1] - (L-x[i])*
    hL_jn[i])/h
107
108         robin_term = u_prev[-2] - ((2*h*gamma)/D)*(u_prev[-1] - v)
109         A_Bd_1 = lambdda*u_prev[-2]
110         A_Bd_2 = (1-(2*lambdda)-K)*u_prev[-1]
111         A_Bd_3 = lambdda*robin_term
112         u_pres[-1] = A_Bd_1 + A_Bd_2 +A_Bd_3
113
114         u_RW[tn,:] = u_pres
115     return u_RW
```

Listing C.2: This progam gives a random-walk approximation of Problem P1. in Chapter 3..

```
1  import numpy as np
2
3  h = 0.1
4  K = (h**2)/(2)
5  L, T = 5,1
6  D = 0.7
7  Nh, NK = int(L/h), int(T/K)
8  lambdda = (D*K)/(h**2)
9
10 def alpha1(j):
```

```
    Numerator = (L-j*h)*K
    alpha1 = Numerator/(2*h)
    return alpha1

alpha = -L/(2*D)
x = np.linspace(0,L,Nh)
u_FDM = np.zeros([NK,Nh])
u0 = 3*np.sin((5*np.pi*x)/(2*L))*np.exp(alpha*x)
u_FDM[0,:] = u0
for tn in range(1,NK):
    u_prev = u_FDM[tn-1,:]
    u_pres = np.zeros(Nh)
    u_pres[0] = 0
    for i in range(1,Nh-1):
        A1 = (lambdda - alpha1(i))*u_prev[i-1]
        A2 = (1-(2*lambdda)-K)*u_prev[i]
        A3 = (lambdda + alpha1(i))*u_prev[i+1]
        u_pres[i] = A1 + A2 + A3
    u_pres[-1] = 2*lambdda*u_prev[-2] +  (1-(2*lambdda)-K)*(u_prev[-1])
    u_FDM[tn,:] = u_pres
```

Listing C.3: This program gives a finite difference approximation to the Problem 2. in Chapter 3..

```
import numpy as np

def prob(h):
    K = (h**2)/(2)
    L, T = 5,1
    D = 0.7
    Nh, NK = int(L/h), int(T/K)
    lambdda = (D*K)/(h**2)

    Solution = np.zeros([NK,Nh])
    x = np.linspace(0,L,Nh)
    prob = D
    alpha = -L/(2*D)
    Solution[0,:] = 3*np.sin((5*np.pi*x)/(2*L))*np.exp(alpha*x)
    scaler = 0.001

    for tn in range(NK-1):
        u_tn = Solution[tn,:]
        #Construct the v_tn vector
        v_tn = np.zeros(Nh)
        for j in range(Nh-1):
            v_tn[j] = (1-K)*u_tn[j] + ((L-j*h)*K/h)*(u_tn[j+1]-u_tn[j])
        v_tn[-1] = (1-K)*u_tn[-2]

        xi_tn = np.zeros(Nh)
        for j in range(Nh-1):
            xi_tn[j] = (v_tn[j] - v_tn[j+1])/h
        xi_tn[-1] = 0
        #xi_tn[-1] = (v_tn[-1] - v_tn[-2])/h
```

```
30
31          Nj = abs(h*xi_tn)/scaler
32          Nj = np.floor(Nj) + 1
33          kj = (h*xi_tn)/Nj
34          fwd, stay, bcd = [], [], []
35          for j in range(len(Nj)):
36              num_par = int(Nj[j])
37              toss_up = np.random.choice(3, num_par, p=[prob/2,1-prob,prob
    /2])
38              toss_up = list(toss_up)
39              heads, zero ,tales = toss_up.count(0), toss_up.count(1),
    toss_up.count(2)
40              fwd.append(heads)
41              stay.append(zero)
42              bcd.append(tales)
43
44          hL_jn = np.zeros(len(Nj))
45          for j in range(1,len(Nj)-1):
46              hL_jn[j] = fwd[j-1]*kj[j-1] + stay[j]*kj[j] + bcd[j+1]*kj[j+1]
47          hL_jn[0] =   stay[0]*kj[0] + bcd[1]*kj[1]
48          hL_jn[-1] =  fwd[Nh-2]*kj[Nh-2] + stay[Nh-1]*kj[Nh-1]
49
50          u_new = np.zeros(len(u_tn))
51          for jj in range(1,Nh-1):
52              u_new[jj] = sum(hL_jn[jj:])
53          #print(hL_jn,'hL')
54
55          u_new[0] = 0
56          u_new[-1] = u_new[-2]
57          Solution[tn+1,:] = u_new
58      return Solution
59
60  a2 = prob(0.1)
```

Listing C.4: The program gives the gradient-random-walk aproximation of the PDE problem 2. in chapter 3.