Markus Krammer

Coulomb Interactions in Kinetic Monte Carlo Simulations for Charge Transport in Organic Semiconductors

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's Degree Program: Technical Physics

submitted to

Graz University of Technology

Supervisor:
Dipl.-Phys. Dr.rer.nat. Karin Zojer

Institute of Solid State Physics

Co-Supervisor:
Univ.-Prof. Dr. Peter Hadley

Graz, June 2016

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present masters thesis.

Datum/Date	Unterschrift/Signature

Danksagung/Acknowledgement

First of all I want to thank Chris for the great support at the beginning of my master thesis during my stay at Durham University. The kind introduction to the research field of kinetic Monte Carlo simulations and the productive meetings were responsible for my efficient and detailed immersion into the topic.

Thank you Peter, for always asking the right questions at the right time. Those questions ensured that I was not loosing sight of the big picture.

I want to thank Egbert and his group for showing me a different view on more or less the same topic and worrying about details that I probably would not have recognised.

Liebe Karin, dir möchte ich ganz besonders dafür danken, dass du immer für mich Zeit hattest und meinen Enthusiasmus für das Thema, auch nach dieser langen Zeit, die ich zum abschließen benötigt habe, immer noch teilst.

Vielen Dank auch an Philipp, dafür dass wir uns in Durham eine lustige Zeit gemacht haben und für die anregenden Diskussionen die wir führten.

Ganz besonders großer Dank gebührt natürlich meinen Eltern, ohne euren bedingungslosen Rückhalt wäre mein Leben niemals so erfüllt mit Freude wie es ist. In meiner langen Studienzeit habt ihr mir genau den passenden Druck gemacht um mich zu motivieren, nicht mehr und nicht weniger. Ich fühlte mich immer verstanden und gut aufgehoben.

Auch dir will ich danken lieber Alex, dafür dass du immer für mich da bist wenn ich mal Abschalten will um etwas Abstand zu allem zu kriegen. Ob im AV-Loch oder im Xeis, mit dir habe ich überall Spaß.

Zuletzt möchte ich noch dir danken liebe Melanie. Egal wie entmutigend und frustrierend manche Abschnitte der Master Arbeit auch waren, eine Umarmung von dir bescherte mir wieder neue Motivation. Und wenn es die Umarmung mal nicht schaffte, so lieferte ein metaphorischer Tritt in den Allerwertesten den nötigen Aufwind.

Bei all jenen die ich vergessen habe namentlich zu erwähnen möchte ich mich entschuldigen und mit folgendem Zitat herzlichst bedanken:

Der schönste Dank ist jener, der aus lauter Freude vergessen wird.

(Walter Ludin)

Kurzfassung

Um ein besseres Verständnis für Landungstransport in organischen Halbleitermaterialien zu erlangen, ist die Anwendung von Kinetic Monte Carlo Simulationen (KMC) mittlerweile weit verbreitet. Die Transporteigenschaften des Systems werden dabei anhand der Trajektorien aller Ladungsträger ermittelt, die durch ein dreidimensionales Gitter von Zellen in Abhängigkeit der individuellen Zelleneigenschaften hüpfen.

Momentan sind zwei unterschiedliche KMC Simulations-Methoden in Verwendung: Dynamic Monte Carlo (DMC) und First Reaction Method (FRM). Bei DMC werden alle Hüpfraten von allen Ladungsträgern nach jedem Simulationsschritt neu berechnet, was korrekte Ergebnisse liefert, jedoch mit hohem Rechenaufwand verbunden ist. Insbesondere bei einer großen Anzahl an Ladungsträgern, verursacht durch hohe Ladungsträgerdichten oder große Systeme, muss man enorme Rechenzeiten in Kauf nehmen. FRM hingegen berechnet nur die Hüpfrate des eben gehüpften Ladungsträgers neu und ist daher zwar schnell; der Verfahrensfehler ist jedoch bei hohen Ladungsträgerdichten nicht vernachlässigbar.

Gerade für dotierte organische Halbleitermaterialien und nahe des Kontakts eines organischen Dünnfilm-Transistors, wo hohe Ladungsträgerdichten auftreten, ist FRM nicht anwendbar. Daher haben wir uns in dieser Masterarbeit damit beschäftigt, eine neue Methode zu entwickeln, die die Schnelligkeit von FRM mit den exakten Ergebnissen von DMC verbindet. Diese Methode wurde zunächst an einfachen Bulksimulationen getestet, wo die Mobilität in Abhängigkeit der Ladungsträgerdichte ermittelt wurde. Nach erfolgreicher Anwendung auf diesem einfachst möglichen Modellsystem, beschäftigten wir uns mit der Weiterentwicklung der Methode für Injektionssimulationen, wo ein Metallkontakt an einer Seite des organischen Halbleitermaterials angebracht wird. Für unterschiedliche Regimes, definiert durch unterschiedliche externe elektrische Feldstärken und Injektionsbarrieren, wurde die gemessene Stromdichte untersucht.

Sowohl für Bulk- als auch für Injektionssimulationen konnten wir zeigen, dass die neue Methode einen geringen Verfahrensfehler mit start reduzierten Rechenzeiten (Faktor 10-100 im Vergleich zu DMC) verbindet. Weiters gelang es uns, den Grund für die erzielten Verbesserungen der neue Methode physikalisch zu interpretieren.

Abstract

Kinetic Monte Carlo simulations (KMC) have become a widely used tool to get a better understanding of the behaviour of charge transport in organic semiconductors and devices. With the help of such simulations, the transport-related properties of a system are derived from the trajectories of all particles hopping through a three-dimensional grid of cells as a function of individually chosen cell properties.

Two types of KMC simulation methods are commonly in use: The Dynamic Monte Carlo method (DMC) and the so-called First Reaction Method (FRM). The DMC method is characterized by updating the hopping rates for all charge carriers in each simulation step and so provides the most exact predictions. However, the method is inherently slow when considering huge amounts of charge carriers and becomes computationally expensive for large system sizes or large charge carrier densities. FRM, by contrast, relies on an approximate, very fast update mechanism in which only the rate of the previously hopped charge carrier is updated. Obviously this method is not considered accurate for large charge carrier densities.

The weakness of FRM is important, since charge carrier densities are large for doped organic semiconductors as well as in the vicinity of the injecting contacts of organic thin-film transistors. In this master thesis we investigate an update mechanism that replicates the accuracy of DMC, particularly in the limit of high charge carrier densities, while being much faster than DMC and only slightly more expensive than FRM. In a first step the performance of this improved update mechanism is examined by determining bulk mobilities as a function of the charge carrier density. After this simplest case of a KMC simulation, the approach is transferred to the more complicated scenario of injection simulations, in which a metal contact is used to inject charge carriers. For different injection regimes, determined by the external electric field strength and the zero-field-energy barrier, the behaviour of our new update mechanism was analysed by measuring the current density.

For both, bulk and injection simulations, the new method combines a low methodological error with a reduced computational effort (factor 10-100 compared to DMC). Furthermore, we are able to give a physical interpretation for the improvements associated with our new method.

Contents

1 Introduction to Kinetic Monte Carlo

Our modern way of life would not be the same without organic semiconductor materials. They are widely used as organic light-emitting diode (OLED) in displays especially for smartphones. Applications for organic photovoltaic (OPV) devices and organic field-effect transistors (OFET) are getting more and more as well. At the moment it is not attempted to replace the commonly used inorganic semiconductor materials. Especially in their core competences like mircocontrollers, storage and other miniaturised electronic systems, organic materials are not competitive. For totally disordered polymer films, charge mobilities of 10^{-6} to 10^{-3} cm^2/Vs are reported and for crystalline organic semiconductors charge mobilities of up to 20 cm^2/Vs are achieved. [1] Comparable values for inorganic semiconductors are much higher (e.g. electron mobilities of 1400 cm^2/Vs for Silicon and 8500 cm^2/Vs for Gallium Arsenide). [2] The low mobilities increase the switching time for transistors a lot and thus make organic electronic devices inherently slower than their inorganic opponents. Additionally the advance of decades of research in the field of miniaturisation for inorganic semiconductors is not so easy to catch up. Nevertheless organic semiconductors are opening a door to a completely new division. Due to properties like flexibility, unbreakability, lightness and low production costs, applications in the area of biomedicine are likely. Amongst others the sector of portable media will continue profiting from organic electronics and also sensors are a promising field of research. [3]

The huge variety of possible materials for organic semiconductors is giving scientists many opportunities but also arises lots of questions. The task to find the perfect material for the desired application is often quite challenging. On one hand the inexhaustible amount of alternatives regarding the molecular design of the materials is exceeding experimentalists patience and on the other hand theoreticians are still struggling with the basic mechanisms that make the organic semiconductor act like it does. Especially in the sector of charge transport many questions are still open. An example would be that for experimentalists it is a well known fact that doping the contact region increases the conductivity of the contact a lot. [4] Exactly this question, why a doped contact is better than a pure one, was the original motivation for this master thesis. As doping a contact in principle means making it more dirty, the answer is not as trivial as it might look like at first sight.

To gain a better understanding of the mechanisms that assign an organic semiconductor its properties, a variety of simulation techniques is used for different length and time-scales. Regarding electronic properties, three of the most popular ones are Density Functional Theory (DFT), Kinetic Monte Carlo (KMC) simulations and Drift Diffusion (DD) simulations. In comparison, with DFT single molecules or at least a handful of molecules can be investigated, KMC is able to simulate a region within a size of up to a few hundred nanometres and DD can analyse a whole device. But with the increasing length scale, the accuracy, of course, suffers. In DFT the fully quantum-mechanical calculations are nearly exact, only correlations beyond mean field approximations are neglected. This means in case of a weakly correlated electronic system, DFT gives good results without the necessity of assuming any non-physical parameters. In contrast KMC simulations cannot be performed without some assumptions. Additionally the time evolution of the system itself is done classically neglecting all quantum-mechanical correlations, only the transition probabilities between the different states of the system are partly calculated quantum-mechanically. The non-physical parameters of the simulations can be either adapted to fit experimental data or taken from a more fundamental theory like DFT. A DD simulation is even more classical, there all quantum mechanical and some classical effects are packed into the mobility. Such a simulation is suited very well for studying geometry effects of devices, but it is not straightforward to consider doping. For this consideration the effect of doping on the mobility has to be known in advance. In future, it is imaginable that those three techniques complement one another in terms of getting information for the assumptions made in one method from the next more precise method. For instance Olivier *et al.* presented a combination of DFT and KMC. [5]

For a KMC simulation, the most challenging part is to evaluate the Coulomb interaction between all charge carriers in a preferably exact, but computationally affordable way. The computational effort scales with the number of charge carriers squared, so the runtime of the simulation explodes when many

charge carriers are in the simulation. Especially for doping scenarios [6] as well as near the contacts of an organic thin-film transistor [7], high charge carrier densities are expected. During this master thesis a method was developed that can digest such high charge carrier densities within a reasonable simulation time in which the Coulomb interactions are considered as accurate as necessary to reproduce the results of an exact implementation.

Kinetic Monte Carlo simulations are a form of Markov Chain Monte Carlo methods. To ensure the correctness of this method, chapter 1.1 is concerning the conditions which have to be fulfilled for a proper kinetic Monte Carlo Simulation. In a next step chapter 1.2 is presenting the used model system and proves the satisfaction of those frame conditions.

1.1 Theoretical Foundations

To develop an understanding for kinetic Monte Carlo simulations, the underlying technique for this method has to be studied. This is the reason why we start by exploring the Markov Chain Monte Carlo method here. As we are interested in kinetic Monte Carlo simulations, only aspects which are relevant for this method are discussed. Additionally we only focus on equilibrium properties of time-homogeneous systems, whereas transient procedures or time dependent external fields are not considered. We start off by investigating discrete-time Markov chains as they are quite intuitive and easy to understand. The concepts introduced during the study of discrete-time Markov chains will help to understand the more complicated behaviour of continuous-time Markov chains, which we will focus on afterwards. As a kinetic Monte Carlo simulation is a realisation of a continuous-time Markov chain, it is obvious that the knowledge of continuous-time Markov chains is a major key to understanding a kinetic Monte Carlo simulation. In a next step we will have to think about taking measurements and evaluating their corresponding errors in our simulation and finally the physical interpretation of our mathematically constructed stochastic process has to be questioned.

Some of the information discussed in the chapters 1.1.1, 1.1.2 and 1.1.3 about 'Markov Chain Monte Carlo', 'Discrete-Time Markov Chains' and 'Continuous-Time Markov Chains' is provided in [8]. A very detailed and more general view on discrete-time Markov chains can be found in [9]. The information provided in chapter 1.1.3 is discussed very suitably and in more detail in [10]. The statements found in chapter 1.1.4 are mainly taken out of [11]. In chapter 1.1.5 the equivalence of the continuous-time Markov Chain Monte Carlo method and the master equation [12] as well as the Poisson process [13, 14] is demonstrated. For some chapters, source [15] is helpful as it provides a rather basic but quite easily accessible approach to Markov Chain Monte Carlo simulations concerning discrete-time Markov chains and error analysis.

1.1.1 Markov Chain Monte Carlo

Markov Chain Monte Carlo is a powerful tool to simulate probability distributions $\pi(x)$ in a high dimensional state space $\mathcal{X}$, in which the population of the states $x \in \mathcal{X}$ is dominated by a small part of the state space. In terms of our goal to perform kinetic Monte Carlo simulations for charge transport in organic semiconductors, only discrete state spaces with a finite number of states are considered here. In physics a very common application is to simulate many body systems with a model Hamiltonian $\hat{H}$. In thermal equilibrium the system obeys the Boltzmann statistics, which means that the expectation value $\langle \hat{A} \rangle_{\text{therm}}$ of an operator $\hat{A}$ in thermal equilibrium is given by

$$\langle \hat{A} \rangle_{\text{therm}} = \frac{1}{Z} \text{tr}\left(\hat{A} e^{-\beta \hat{H}}\right) \quad \text{with the partition function} \quad Z = \text{tr}\left(e^{-\beta \hat{H}}\right) \tag{1.1}$$

where tr is the trace of the operator(s), $\beta = \frac{1}{k_B T}$ is the inverse of the Boltzmann factor k_B times the temperature T and $\hat{H}$ is the Hamilton operator. The system reduces to a classical one if the Hamilton

operator is diagonal in the chosen state space $\mathcal{X}$. In such a case the partition function simplifies to

$$Z = \mathrm{tr}\left(e^{-\beta\hat{H}}\right) = \sum_{x\in\mathcal{X}} \langle x| e^{-\beta\hat{H}} |x\rangle = \sum_{x\in\mathcal{X}} e^{-\beta E_x} \quad \text{with} \quad \hat{H}|x\rangle = E_x|x\rangle \tag{1.2}$$

and an expectation value gets

$$\left\langle \hat{A} \right\rangle_{\mathrm{therm}} = \frac{1}{Z} \sum_{x\in\mathcal{X}} \langle x|\hat{A}e^{-\beta\hat{H}}|x\rangle = \sum_{x\in\mathcal{X}} \left\langle \hat{A} \right\rangle_x \frac{e^{-\beta E_x}}{Z} \quad \text{with} \quad \left\langle \hat{A} \right\rangle_x = \langle x|\hat{A}|x\rangle . \tag{1.3}$$

The classical nature of the system is getting obvious in the last equation in which the Boltzmann distribution can be abstracted

$$\pi_x = \frac{e^{-\beta E_x}}{Z} \tag{1.4}$$

In general any probability distribution π_x with classical states x out of a discrete state space $\mathcal{X}$ can be sampled with Markov Chain Monte Carlo. Especially for high dimensional state spaces the goal of such a simulation is to get expectation values

$$\langle A \rangle = \sum_{x\in\mathcal{X}} A_x \pi_x \tag{1.5}$$

for measurable quantities A of the system, where A_x is the value for this quantity in state x, rather than sampling the whole state space. To achieve this, a sequential stochastic process is introduced in Markov Chain Monte Carlo, namely the Markov chain. Each link of such a chain is a state $x_t \in \mathcal{X}$ and the sequence is ordered by a Markov time t. For a discrete-time Markov chain the time distance between two neighbouring links of the chain Δt_{min} is always the same. As the Markov time itself usually has no physical relevance, the time interval can be chosen to $\Delta t_{min} = 1$. In contrast kinetic Monte Carlo simulations are continuous-time Markov chains (also called Markov jump processes) in which the time evolution is continuous $\Delta t_{min} \in \mathbb{R}^+$. For both alternatives the essential characteristic of the Markov chain is the **Markov property**

$$P(x_{t+\Delta t_{min}} | x_u, u \leq t) = P(x_{t+\Delta t_{min}} | x_t) \tag{1.6}$$

which means that the probability that a state $x_{t+\Delta t_{min}}$ is visited at time $t + \Delta t_{min}$ is only depending on the state x_t that was visited directly prior in Markov time. As a consequence the future evolution of a Markov chain only depends on the currently occupied state x_t and the transition probabilities $P(x'_{t+\Delta t}|x_t)$ to get from state x, occupied at time t, to state x' within a time Δt.

1.1.2 Discrete-Time Markov Chains

In this chapter only discrete-time Markov chains with $\Delta t_{min} = 1$ are studied. As we are only considering discrete state spaces with a finite number of states, a single transition between two neighbouring links in a Markov chain can be described by a transition matrix with matrix element $\mathbf{T}_{x,x'} := P(x'_{t+1}|x_t) = P(x'_1|x_0)$ where in the last part the time-homogeneity was used. Note that only due to the Markov property a transition matrix $\mathbf{T}$ can be used to describe the evolution of the Markov chain. For convenience we define $p_1(x,x') := P(x'_{t+1}|x_t) = P(x'_1|x_0)$. All elements of $\mathbf{T}$ are non-negative and the elements in each row have to sum up to unity. Such a matrix is called a stochastic matrix. Performing n steps leads to a transition probability $p_n(x,x') := P(x'_n|x_0) = (\mathbf{T}^n)_{x,x'}$. For any $l,m,n \in \mathbb{N}$ with $l+m=n$ the relation

$$p_n(x,x') = (\mathbf{T}^n)_{x,x'} = (\mathbf{T}^{l+m})_{x,x'} = \sum_{y\in\mathcal{X}} (\mathbf{T}^l)_{x,y} (\mathbf{T}^m)_{y,x'} = \sum_{y\in\mathcal{X}} p_l(x,y) p_m(y,x') \tag{1.7}$$

holds. This relation is called the Chapman-Kolmogorov equation.

Now we can introduce a row vector $\tilde{\pi} = (\tilde{\pi}_x)$ with entries representing the probability that a certain state

is occupied. Starting with an initial probability distribution $\tilde{\pi}^0$, the Markov chain evolves this probability distribution

$$\tilde{\pi}^n = \tilde{\pi}^0 \mathbf{T}^n \tag{1.8}$$

within a Markov time $n \in \{0,1,2,...\}$. Finally we would like to end up with our desired probability distribution

$$(\pi_x) = \lim_{n\to\infty} \tilde{\pi}^n \tag{1.9}$$

not depending on the starting configuration $\tilde{\pi}^0$. To assure this, certain conditions have to be fulfilled. So we will now have a look at some properties of a Markov chain.

Fundamental Terminology:
A state $x \in \mathcal{X}$ is said to lead to state $x' \in \mathcal{X}$, written as $x \to x'$, if a finite number $n \geq 0$ exists for which a transition from x to x' is possible $p_n(x,x') > 0$. Two states $x,x' \in \mathcal{X}$ are said to communicate with each other $x \leftrightarrow x'$ if $x \to x'$ and $x' \to x$. A set of states $\mathcal{C} \subseteq \mathcal{X}$ is said to be a communicating class if all states within this class $x \in \mathcal{C}$ communicate with each other and no states from outside $x' \in \mathcal{X} \setminus \mathcal{C}$ are communicating with states in $\mathcal{C}$. A Markov chain is said to be **irreducible**, if the only communicating class is $\mathcal{X}$. In other words the Markov chain can access every state x' from all states x within a finite Markov time.

A state is said to be periodic with period $d \geq 2$, if the state can only be revisited after a multiple of d steps. More precise d is the largest number for which the condition

$$\sum_{n=1}^{\infty} p_{n\cdot d}(x,x) = 1 \tag{1.10}$$

holds. The state is called **aperiodic** when $d = 1$. A sufficient but not necessary condition for aperiodicity is $p_1(x,x) > 0$. As an example a state that could be revisited at the earliest after m steps, but also after $m+1, m+2, m+3, \ldots$ steps is aperiodic as well. In case of an irreducible chain, all states have the same period. This leads to the fact that if one aperiodic state exists in an irreducible chain, the whole chain is aperiodic.

If the chain is irreducible and aperiodic, a positive integer n has to exist for which the transition matrix $\mathbf{T}$ only contains strictly positive numbers $\exists n \in \mathbb{N}$ for which $(\mathbf{T}^n)_{x,x'} > 0 \ \forall x,x' \in \mathcal{X}$. A Markov chain satisfying the condition of only strictly positive entries in the transition matrix is called a **regular chain**.

A state $y \in \mathcal{X}$ is said to be **recurrent**, if the probability that the state is revisited within a time $\Delta t < \infty$ is unity; $P(x_{\Delta t} = y | x_0 = y \wedge \Delta t < \infty) = 1$. If this is not the case, it is called transient. A recurrent state where the expectation value of the time needed to revisit the state is finite $\langle \Delta t \rangle < \infty$ is called **positive recurrent**, otherwise it is null-recurrent. Note that aperiodicity does not necessarily imply positive recurrence as aperiodicity describes the finite Markov time t behaviour, whereas positive recurrence considers the limiting behaviour of $t \to \infty$.

A stochastic process $\{x_t, t \geq 0\}$ is called **regenerative** if it 'regenerates' itself at certain times $t_0 \leq t_1 < t_2 < t_3 \ldots$ of the form $t_n = A_1 + A_2 + \cdots + A_n$ with $n \in \{1,2,3,\ldots\}$ and $\{A_n\}$ being identically independent distributed. Regeneration means that the stochastic process $\{x_{t_0+t}, t \geq 0\}$ has the same distribution as the stochastic process $\{x_{t_n+t}, t \geq 0\}$. In other words the probabilistic behaviour of the stochastic process starting at time t_n is exactly the same as at time t_0. After a time t_n the process acts as if it has just started new. The t_n are called regeneration times and the first regeneration time t_0 has a special status; if $t_0 = 0$ the process is named pure, otherwise it is named delayed. The $\{A_n\}$ are called cycle lengths and it is said to have a lattice distribution if A_n takes values in the lattice $\{a + n \cdot b, n \in \mathbb{N}\}$ for some values a and $b \neq 0$ where b is called the period.

Existence of a Limiting Distribution:
Those definitions are preparing the path towards the most important part of this chapter, the limiting or steady-state behaviour of our Markov chain as the number of steps in Markov time $n \to \infty$.
First we will ensure the existence of a limiting distribution with the help of the following theorem:

Theorem 1.1 (Regeneration Theorem - [8] Theorem A.9.1, page 631) *Let $\{x_t\}$ be a discrete-time regenerative process with cycle length distribution of period $b = 1$ and expectation $\langle A_1 \rangle < \infty$. Then, x_t converges in distribution to a random number x, such that for all joint probability density functions f*

$$\langle f(x) \rangle = \frac{1}{\langle A_1 \rangle} \left\langle \sum_{t=t_0}^{t_1-1} f(x_t) \right\rangle \tag{1.11}$$

provided that the expectation exists.

As this theorem is valid for all joint probability density functions f it is also valid for the probability density $\tilde{\pi}_y^t = P(x_t = y)$ and the convergence in distribution ensures that a limiting distribution $\lim_{t\to\infty} \tilde{\pi}_x^t = \tilde{\pi}_x$ exists. Our Markov chain $\{x_t\}$ is a discrete-time regenerative process, in which possible regeneration times are the times when the process revisits a specific state. With the help of theorem 1.1 and assuming aperiodicity and irreducibility, it can be shown that

$$\lim_{t\to\infty} p_t(x,x') = \tilde{\pi}_{x'} \tag{1.12}$$

which means that the transition probability $p_t(x,x')$ for all pairs of states $x,x' \in \mathcal{X}$ leads to a quantity $\tilde{\pi}_{x'} \in [0,1]$ that in the limit $t \to \infty$ does not depend on the initial state x. If the state x' is additionally positive recurrent the quantity $\tilde{\pi}_{x'} > 0$ and, if not, $\tilde{\pi}_{x'} = 0$. As the transition probability $p_t(x,x')$ is equal to the matrix entry $(\mathbf{T}^t)_{x,x'}$ of the stochastic transition matrix $\mathbf{T}$ which has to satisfy $\sum_{x'} \mathbf{T}_{x,x'} = 1$, and, hence, $\sum_{x'} (\mathbf{T}^t)_{x,x'} = 1$, the quantity $\tilde{\pi}_x$ is a probability density. The simplified reason behind this convergence is, that the Markov chain 'forgets' where it started and so, guaranteed that every state can be visited at any time, it reaches an equilibrium probability density $\tilde{\pi}_{x'}$ independent of the starting probability density $\tilde{\pi}_y^0 = \delta_{x,y}$. Equation (1.12) is equivalent to

$$\lim_{t\to\infty} (\mathbf{T}^t)_{x,x'} = \tilde{\pi}_{x'} \quad \forall x,x' \in \mathcal{X} \tag{1.13}$$

which means that every row of the transition matrix $\mathbf{T}^t$ converges to the probability distribution vector $\tilde{\pi}$.
In summary we can say that given the Markov chain is irreducible and aperiodic, a probability density $\tilde{\pi}_{x'} = \lim_{t\to\infty} p_t(x,x')$ for all $x,x' \in \mathcal{X}$ with $\sum_{x\in\mathcal{X}} \tilde{\pi}_x = 1$ exists and for all states that are positive recurrent additionally $\tilde{\pi}'_x > 0$ holds. In a next step we will have a look at the uniqueness of $\tilde{\pi}$.

Uniqueness:

Theorem 1.2 (Limiting Distribution - [8] Theorem A.10.1, page 634) *For an irreducible, aperiodic Markov chain with transition matrix $\mathbf{T}$, if the limiting distribution $\tilde{\pi}$ exists, then $\tilde{\pi}$ is uniquely determined by the solution of the constrained system of equations*

$$\tilde{\pi} = \tilde{\pi}\mathbf{T}, \quad \sum_{x\in\mathcal{X}} \tilde{\pi}_x = 1, \quad \tilde{\pi}_x > 0 \quad \forall x \in \mathcal{X} \tag{1.14}$$

Conversely, if there exists a row vector $\tilde{\pi}$ satisfying (1.14)*, then $\tilde{\pi}$ is the limiting distribution of the Markov chain. In addition $\tilde{\pi}_x > 0 \ \ \forall x \in \mathcal{X}$ and all states are positive recurrent.*

This important theorem not just ensures the uniqueness of the probability distribution $\tilde{\pi}$, it additionally tells us that irreducibility and aperiodicity leads to the fact that all states are positive recurrent. A probability distribution $\tilde{\pi}$ fulfilling (1.14) is called a **stationary distribution** as it does not change during the Markov process.

Correct Sampling:
Now having an existing and unique probability density $\tilde{\pi}$, the last step is to develop a convenient check if we are sampling out of the right probability distribution π with our transition matrix $\mathbf{T}$. Fortunately this check is delivered with theorem 1.2. If we multiply the very left equation in (1.14) in its sum representation $\tilde{\pi}_{x'} = \sum_{x\in\mathcal{X}} \tilde{\pi}_x p_1(x,x') \;\; \forall x' \in \mathcal{X}$ by $\sum_{x\in\mathcal{X}} p_1(x',x) = 1$ we get the **global balance equation**

$$\sum_{x\in\mathcal{X}} \tilde{\pi}_{x'} p_1(x',x) = \sum_{x\in\mathcal{X}} \tilde{\pi}_x p_1(x,x') \qquad \forall x' \in \mathcal{X} \tag{1.15}$$

which indicates the balance between the 'probability flux' out of state x' and into state x'. A much stronger assumption than (1.15) is the **detailed balance equation**

$$\tilde{\pi}_{x'} p_1(x',x) = \tilde{\pi}_x p_1(x,x') \qquad \forall x,x' \in \mathcal{X}. \tag{1.16}$$

If this equation is fulfilled for all $x,x' \in \mathcal{X}$, (1.15) is fulfilled as well. With (1.16) we have found our convenient check to assure that we are sampling from the right probability density. Additionally, if we find a probability distribution $\tilde{\pi}$ which satisfies (1.16), the existence of a non-trivial solution to (1.14) is guaranteed as well.

Finally we can stress that an irreducible and aperiodic Markov chain with transition probabilities $p_1(x,x')$ and a transition matrix $\mathbf{T}_{x,x'} = p_1(x,x')$ has a unique stationary distribution $\tilde{\pi} = \tilde{\pi}\mathbf{T}$ which can be found by $\tilde{\pi}_{x'} = \lim_{t\to\infty}(\mathbf{T}^t)_{x,x'} \;\; \forall x,x' \in \mathcal{X}$ and if a probability distribution π satisfies the detailed balance equation $\pi_{x'} p_1(x',x) = \pi_x p_1(x,x') \;\; \forall x,x' \in \mathcal{X}$ the probability distributions are the same $\pi = \tilde{\pi}$.

1.1.3 Continuous-Time Markov Chains

Like in the last chapter about 'Discrete-Time Markov Chains' we again want to get a convenient recipe to check if we are sampling out of the desired probability distribution π. As our Markov time $t \in \mathbb{R}^+$ now evolves in continuous-time, things get more complicated. Even the question of how the sample-path behaviour of a Markov chain in continuous-time looks like is not trivial any more and will be answered at the end of this chapter.

Transition Function and Kolmogorov Equations:
As for discrete-time Markov chains we again start with the Markov property, which is the basic concept of a Markov chain. It tells us that the evolution of a Markov chain does not depend on the whole history of the chain but only on the last point in time of this history. In a more mathematical formulation, the probability that the Markov chain $\{x(t), t \geq t_0\}$ starting at time t_0 is in a state x' at time t', given the history of the Markov chain $\{x(t), t_0 \leq t \leq t_1\}$ for all times t within an interval $[t_0,t_1]$, is determined only by the state that it is in at time t_1.

$$P(x'(t')|\{x(t), t_0 \leq t \leq t_1\}, t' \geq t_1) = P(x'(t')|x(t_1), t' \geq t_1) \tag{1.17}$$

We are only considering time-homogeneous Markov chains and so we can set $t_1 = 0$ and define the transition probability $p_t(x,x') = P(x'(t)|x(0))$ in the same way as above. Considering only state spaces with a finite number of states, we can again arrange this transition probabilities in a matrix $\mathbf{T}_{x,x'}(t) = p_t(x,x')$. The very crucial difference to the discrete case is that the Markov time is no longer simply the power of the transition matrix $\mathbf{T}^t$ but rather has a functional dependence $\mathbf{T}(t)$. Thus we call the matrix $\mathbf{T}(t)$ a transition function. At this point some definitions have to be made:

A function $\mathbf{T}_{x,x'}(t)$ is called a transition function if

1. $\mathbf{T}_{x,x'}(t) \geq 0$ for all $t \geq 0$ and $x,x' \in \mathcal{X}$

   and $\quad \mathbf{T}_{x,x'}(0) = \delta_{x,x'} \quad$ where $\quad \delta_{x,x'} = \begin{cases} 1 & \text{for } x = x' \\ 0 & \text{for } x \neq x' \end{cases}$ is the Kronecker delta;

2. $\sum_{x' \in \mathcal{X}} \mathbf{T}_{x,x'}(t) \leq 1 \ \ \forall t \geq 0$ and $\forall x \in \mathcal{X}$. If $\sum_{x' \in \mathcal{X}} \mathbf{T}_{x,x'}(t) = 1 \ \ \forall t \geq 0$ and $\forall x \in \mathcal{X}$, $\mathbf{T}_{x,x'}(t)$ is said to be honest, otherwise it is dishonest;

3. $\mathbf{T}_{x,x'}(t+s) = \sum_{y \in \mathcal{X}} \mathbf{T}_{x,y}(t)\mathbf{T}_{y,x'}(s) \ \ \forall s,t \geq 0$ and $\forall x,x' \in \mathcal{X}$ (this is the Chapman-Kolmogorov equation).

$\mathbf{T}_{x,x'}(t)$ is called a standard transition function if it additionally fulfils

4. $\lim_{t \to 0} \mathbf{T}_{x,x}(t) = 1 \ \ \forall x \in \mathcal{X}$ (due to $0 \leq \sum_{x' \in \mathcal{X} \setminus \{x\}} \mathbf{T}_{x,x'}(t) \leq 1 - \mathbf{T}_{x,x}(t)$ we have $\lim_{t \to 0} \mathbf{T}_{x,x'}(t) = \delta_{x,x'} \ \ \forall x,x' \in \mathcal{X}$)

For kinetic Monte Carlo simulations, we are not starting from a transition function, but rather from a **rate equation** which gives us the transition probabilities per unit time between two states. With this rate equation we can build up a so called q-matrix. For our time-homogeneous system this q-matrix is not time-dependent.
A square matrix $Q = (q_{x,x'})$ with $x,x' \in \mathcal{X}$ is called a q-matrix if

$$0 \leq q_{x,x'} < \infty \ \ \forall x,x' \in \mathcal{X} \quad \text{with} \quad x \neq x' \tag{1.18}$$

and

$$\sum_{x' \in \mathcal{X} \setminus \{x\}} q_{x,x'} \leq q_x \leq \infty \ \ \forall x \in \mathcal{X} \quad (\text{where } \ q_{x,x} := -q_x). \tag{1.19}$$

Q is called stable if $q_x < \infty \ \ \forall x \in \mathcal{X}$, conservative if

$$\sum_{x' \in \mathcal{X} \setminus \{x\}} q_{x,x'} = q_x \ \ \forall x \in \mathcal{X}, \tag{1.20}$$

and uniformly bounded if

$$\sup_{x \in \mathcal{X}} q_x < \infty. \tag{1.21}$$

Note, that in our case of a finite dimensional state space, stable and uniformly bounded is in principle the same.
A transition function $\mathbf{T}_{x,x'}(t)$ is called a Q-function if Q is the q-matrix of $\mathbf{T}_{x,x'}(t)$ which means

$$\mathbf{T}'_{x,x'}(0) = \lim_{t \to 0} \frac{\mathbf{T}_{x,x'}(t) - \delta_{x,x'}}{t} = q_{x,x'} \tag{1.22}$$

or in matrix representation simply

$$\mathbf{T}'(0) = \lim_{t \to 0} \frac{\mathbf{T}(t) - \mathbb{1}}{t} = Q \tag{1.23}$$

where $\mathbb{1}$ is the identity matrix. As we have a finite dimensional state space we do not have to care about the convergence of infinite summations and we can write

$$\mathbf{T}'(t) = \lim_{h \to 0} \frac{\mathbf{T}(t+h) - \mathbf{T}(t)}{h} = \lim_{h \to 0} \frac{\mathbf{T}(t)\mathbf{T}(h) - \mathbf{T}(t)}{h} = \mathbf{T}(t) \left(\lim_{h \to 0} \frac{\mathbf{T}(h) - \mathbb{1}}{h} \right) = \mathbf{T}(t)Q \tag{1.24}$$

$$\mathbf{T}'(t) = \lim_{h \to 0} \frac{\mathbf{T}(h+t) - \mathbf{T}(t)}{h} = \lim_{h \to 0} \frac{\mathbf{T}(h)\mathbf{T}(t) - \mathbf{T}(t)}{h} = \left(\lim_{h \to 0} \frac{\mathbf{T}(h) - \mathbb{1}}{h} \right) \mathbf{T}(t) = Q\mathbf{T}(t) \tag{1.25}$$

with the help of the Chapman-Kolmogorov equation. We call $\mathbf{T}'(t) = \mathbf{T}(t)Q$ the Kolmogorov forward equation and $\mathbf{T}'(t) = Q\mathbf{T}(t)$ the Kolmogorov backward equation.

Theorem 1.3 ([10] Theorem 2.2.2, page 70) *Let Q be a stable but not necessarily conservative q-matrix. Then there exists a (possibly dishonest) transition function $f_{x,x'}(t)$ satisfying both the Kolmogorov backward and forward equation, with the property that $f_{x,x'}(t)$ is the minimal solution of each of these equations, in the sense that if $\mathbf{T}_{x,x'}(t)$ is any non-negative solution (not necessarily a transition function) of either the backward or forward equation, then $f_{x,x'}(t) \leq \mathbf{T}_{x,x'}(t) \;\; \forall x,x' \in \mathcal{X}$ and $\forall t \geq 0$. Furthermore, $f_{x,x'}(t)$ is the minimal Q-function; that is, if $\mathbf{T}_{x,x'}(t)$ is any other Q-function (not necessarily a solution of either the backward or forward equation), then $f_{x,x'}(t) \leq \mathbf{T}_{x,x'}(t) \;\; \forall x,x' \in \mathcal{X}$ and $\forall t \geq 0$.*

This theorem assures that for every stable q-matrix a corresponding transition function (a Q-function) exists and this so called minimal Q-function additionally is a solution to the Kolmogorov forward and backward equation.

Theorem 1.4 ([10] Proposition 2.2.9, page 83) *Let Q be a not necessarily conservative uniformly bounded q-matrix. Then the minimal solution $f_{x,x'}(t)$ is the unique Q-function.*

Now we define a function

$$\mathbf{T}(t) := e^{tQ} = \sum_{n=0}^{\infty} \frac{t^n}{n!} Q^n \tag{1.26}$$

for a conservative, finite dimensional, stable (and hence uniformly bounded) q-matrix. Proving that $\mathbf{T}(t)$ is a transition function, $(\mathbf{T}'(0) = Q)$ and that $\mathbf{T}(t)$ satisfies the Kolmogorov backward and forward equation would prove that it is a Q-function. Due to theorem 1.4, only one unique Q-function can be found for a uniformly bounded q-matrix which is the minimal Q-function. I.e., once we find a Q-function, it has to be the minimal Q-function.

With the summation representation

$$\mathbf{T}(t) = \sum_{n=0}^{\infty} \frac{t^n}{n!} Q^n = \mathbb{1} + tQ + \frac{t^2}{2} Q^2 + \frac{t^3}{6} Q^3 + \dots \tag{1.27}$$

it is obvious that $\mathbf{T}'(0) = Q$ and $\mathbf{T}'(t) = \mathbf{T}(t)Q = Q\mathbf{T}(t)$. However, to prove that $\mathbf{T}(t)$ is a transition function is not as obvious. For that we have to demonstrate all properties that defines a transition function. We start to do this by showing that $\mathbf{T}_{x,x'}(t) \geq 0 \;\; \forall x,x' \in \mathcal{X}$. As a preparation we need to show that for two commuting matrices A and B, which means $[A,B] = AB - BA = 0$ or equivalently $AB = BA$ the matrix exponential satisfies $e^{A+B} = e^A e^B$:

$$\begin{aligned}
e^{A+B} &= \sum_{n=0}^{\infty} \frac{(A+B)^n}{n!} && (1.28)\\
&= \sum_{n=0}^{\infty} \frac{\sum_{m=0}^{n} A^m B^{n-m} \frac{n!}{m!(n-m!)}}{n!} && (1.29)\\
&= \sum_{n=0}^{\infty} \sum_{m=0}^{n} \frac{A^m}{m!} \frac{B^{n-m}}{(n-m)!} && (1.30)\\
&= \sum_{k=0}^{\infty} \frac{A^k}{k!} \sum_{l=0}^{\infty} \frac{B^l}{l!} && (1.31)\\
e^{A+B} &= e^A e^B && (1.32)
\end{aligned}$$

where, (i), in the second line the binomial formula $(A+B)^n = \sum_{m=0}^{n} A^m B^{n-m} \frac{n!}{m!(n-m)!}$ was used (note that the sorting of products like $ABA = A^2B$ is only possible due to the fact that A and B commute) and, (ii), in the fourth line the summation over (n,m) was replaced by a summation over $(k,l) = (m, n-m)$ with the corresponding summation limits $n \geq 0$ and $0 \leq m \leq n$ transforming to $k \geq 0$ and $l \geq 0$.

Furthermore, for a matrix M which has only non-negative entries $M_{x,x'} \geq 0$, the relation

$$(e^M)_{x,x'} = \left(\sum_{n=0}^{\infty} \frac{M^n}{n!} \right)_{x,x'} \geq 0 \tag{1.33}$$

is trivially fulfilled, as we are only summing up non-negative terms. As we know that all off-diagonal entries of our q-matrix are non-negative $q_{x,x'} \geq 0 \ \ \forall x,x' \in \mathcal{X}$ with $x \neq x'$ and the diagonal elements have a finite infimum $\exists \lambda > -\infty$ for which $q_{x,x} > \lambda \ \ \forall x \in \mathcal{X}$, the matrix $M = t(Q - \lambda \mathbb{1})$ has only non-negative entries.

$$e^M = e^{t(Q-\lambda\mathbb{1})} = e^{tQ}e^{-t\lambda\mathbb{1}} = e^{tQ}\mathbb{1}e^{-t\lambda} = \mathbf{T}(t)e^{-t\lambda} \tag{1.34}$$

Here we used that the identity matrix commutes with every other matrix and $e^{\alpha\mathbb{1}} = \mathbb{1}e^{\alpha} \ \ \forall \alpha \in \mathbb{R}$. From this we see that, as $e^{-t\lambda} > 0 \ \ \forall t, \lambda \in \mathbb{R}$, the relation $(e^M)_{x,x'} \geq 0 \ \ \forall x,x' \in \mathcal{X}$ directly implies that $\mathbf{T}_{x,x'}(t) \geq 0 \ \ \forall x,x' \in \mathcal{X}$ and $\forall t \geq 0$.

The relation $\mathbf{T}(0) = \mathbb{1}$ follows directly from the summation representation of the matrix exponential (1.27).

$$\mathbf{T}(t)\mathbf{1} = \mathbb{1}\mathbf{1} + \left(\sum_{n=1}^{\infty} \frac{t^n}{n!} Q^{n-1}\right) Q\mathbf{1} = \mathbf{1} \tag{1.35}$$

with $\mathbf{1}$ a vector $\mathbf{1}_x = 1 \ \ \forall x \in \mathcal{X}$. This means that our conservative q-matrix $Q\mathbf{1} = 0$ leads to a honest transition function $\mathbf{T}(t)\mathbf{1} = \mathbf{1}$.

With (1.32) and $[tQ, sQ] = ts[Q,Q] = 0$ the Chapman-Kolmogorov equation is easy to prove:

$$\mathbf{T}(t+s) = e^{(t+s)Q} = e^{tQ}e^{sQ} = \mathbf{T}(t)\mathbf{T}(s) \tag{1.36}$$

Furthermore the relation

$$\lim_{t\to 0} \mathbf{T}(t) = \lim_{t\to 0} \sum_{n=0}^{\infty} \frac{t^n}{n!} Q^n = \mathbb{1} \tag{1.37}$$

tells us that $\mathbf{T}(t)$ is standard.

Summing up, we can say that $\mathbf{T}(t) = e^{tQ}$ is the unique, honest, standard transition function that belongs to our conservative, finite dimensional, stable (and hence uniformly bounded) q-matrix.

Existence and Uniqueness:

As we now exactly know the properties of our transition function for our Markov chain, we can go on with thinking about the limiting behaviour.

Theorem 1.5 ([10] Theorem 5.1.6, page 160) *Suppose that $\mathbf{T}_{x,x'}(t)$ is an irreducible transition function.*

1. *Then the limits $\tilde{\pi}_{x'} = \lim_{t\to\infty} \mathbf{T}_{x,x'}(t)$ exist and are independent of x for all $x' \in \mathcal{X}$. The set $\{\tilde{\pi}_x, x \in \mathcal{X}\}$ is a stationary distribution and either*
   - *(a) $\tilde{\pi}_x = 0 \ \ \forall x \in \mathcal{X}$ or*
   - *(b) $\tilde{\pi}_x > 0 \ \ \forall x \in \mathcal{X}$ and $\sum_{x\in\mathcal{X}} \tilde{\pi}_x = 1$.*
2. *Suppose $\pi = (\pi_x, x \in \mathcal{X})$ is a probability vector such that $\pi\mathbf{T}(t) = \pi$ for some $t > 0$. Then $\pi\mathbf{T}(t) = \pi$ for all $t \geq 0$ (this means π is a stationary distribution) and $\pi = \tilde{\pi}$ where $\tilde{\pi}$ is as in part 1.*

The term irreducible is defined in the same way as for discrete-time Markov chains. For a given q-matrix $x \to x'$ with $x,x' \in \mathcal{X}$ if a finite series of n steps of strictly positive matrix elements $\{q_{y_i,y_{i+1}} > 0, i \in \{1,2,\ldots,n-1\}$ with $y_1 = x$ and $y_n = x'$ can be found.

Like in chapter 1.1.2 about discrete-time Markov chains, we are interested in finding a convenient check to assure that we are sampling out of the correct probability density π. With theorem 1.5, we guarantee that the stationary distribution $\pi\mathbf{T}(t) = \pi$ is exactly the one that we are approaching in the long time behaviour. The fact that $\mathbf{T}(t)$ is honest $\mathbf{T}(t)\mathbf{1} = \mathbf{1}$ in addition to our finite dimensional state space ensures that the stationary distribution of the limiting behaviour $\tilde{\pi}$ is not vanishing.

Checking the relation $\pi\mathbf{T}(t) = \pi$ is not really convenient, so we use (1.26) to get a more handy condition:

$$\pi\mathbf{T}(t) = \pi e^{tQ} = \pi\left(\mathbb{1} + Q\sum_{n=1}^{\infty}\frac{t^n}{n!}Q^{n-1}\right) = \pi + (\pi Q)\sum_{n=1}^{\infty}\frac{t^n}{n!}Q^{n-1} \stackrel{!}{=} \pi \tag{1.38}$$

Obviously, $\pi\mathbf{T}(t) = \pi$ is true if $\pi Q = 0$. This leads us, like for the discrete-time Markov chains, to a global balance equation $\sum_{x\in\mathcal{X}}\pi_x q_{x,x'} = 0 \ \forall x' \in \mathcal{X}$. For our conservative q-matrix $\sum_{x'\in\mathcal{X}\setminus\{x\}} q_{x,x'} = -q_{x,x} \ \forall x \in \mathcal{X}$, we get

$$\sum_{x'\in\mathcal{X}\setminus\{x\}}\pi_{x'}q_{x',x} = \sum_{x'\in\mathcal{X}\setminus\{x\}}\pi_x q_{x,x'} \quad \forall x \in \mathcal{X}. \tag{1.39}$$

This implies the desired, much stricter, but more convenient requirement

$$\pi_{x'}q_{x',x} = \pi_x q_{x,x'} \ \forall x,x' \in \mathcal{X}. \tag{1.40}$$

This equation is again called detailed balance equation.

So finally we can sum up our results: A conservative, finite dimensional, stable (and hence uniformly bounded) and irreducible q-matrix leads to a unique, honest, standard transition function $\mathbf{T}(t) = e^{tQ}$ and this transition function leads to the limiting distribution $\pi_{x'} = \lim_{t\to\infty}\mathbf{T}_{x,x'}(t) \ \forall x \in \mathcal{X}$. This limiting distribution is the same distribution as the stationary distribution determined by the detailed balance equation (1.40).

Construction of a Continuous-Time Markov Chain:
It seems that we are finished with this chapter as we now have shown that our stochastic process $\{x(t), t \geq 0\}$ does exactly what we want. But unfortunately we did not care about how we could construct our Markov chain.

Theorem 1.6 (Sample-Path Behaviour - [8] Theorem A.11.1, page 636) *For each stable and conservative q-matrix there exists a continuous-time Markov chain $\{x(t), t \geq 0\}$ whose paths are right-continuous step functions up to a certain random time t_∞. Moreover, the sample-path behaviour up to t_∞ can be described as follows:*

1. *Given its past, the probability that $\{x(t), t \geq 0\}$ jumps from its current state x to state x' is*

$$P(x'|x) = \frac{q_{x,x'}}{q_x}. \tag{1.41}$$

2. *The amount of time that $\{x(t), t \geq 0\}$ spends in state x has an exponential distribution with rate parameter q_x, independent of the past history.*

t_∞ is the so called time of the first infinity, which is $t_\infty = \infty$ in our case of a conservative, finite dimensional, stable (and hence uniformly bounded) and irreducible q-matrix. This means that our sample-path behaviour is as described in theorem 1.6 for all times $t \geq 0$.

1.1.4 Taking Measurements and Error Analysis

Expectation Values:
In chapter 1.1.3 we have learned, (i), how to construct our Markov chain $\{x(t), t \geq 0\}$ and, (ii), that the probability that the chain is in a certain state x' tends to the desired probability distribution $\lim_{t\to\infty} P(x(t) = x') = \pi_{x'}$. To take measurements, we could start a first Markov chain with a certain state $x_1(0)$ and let the system evolve corresponding to theorem 1.6 for a sufficiently long time t_0 so that the system has equilibrated. Now, as the state $x_1(t_0)$ is representative for the probability distribution π_x, we could take the measurement $A_1 = A_{x_1(t_0)}$ of an observable A_x. If we repeat this procedure N_m times (start at a certain

state $x_2(0)$, equilibrate and measure $A_2 = A_{x_2(t_0)}$, start at a certain state $x_3(0)$, equilibrate and measure $A_3 = A_{x_3(t_0)}$, ...) we would get N_m measures $A_1, A_2, \ldots, A_{N_m}$ and could calculate an estimator for the expectation value $\langle A \rangle = \frac{1}{N_m} \sum_{n=1}^{N_m} A_n$. This procedure is actually not favourable as we are spending a lot of time with getting rid of the starting configuration and reaching the limiting distribution to get just one single measurement. Additionally the procedure could be biased due to the choice of the starting configuration.

A much better approach is to evolve only one Markov chain $\{x(t), t \geq 0\}$. Of course, we have to get rid of the starting probability density also in this case, i.e., we again have to let the chain evolve for a certain period of time t_0. This part of the simulation is called thermalisation. After thermalisation, it is supposed that the Markov chain has reached the limiting distribution $P(x(t_0)) = \pi_x$. Once having reached the limiting (stationary) distribution, the state of the Markov chain at any later time $t_0 + t$ with $t \geq 0$ is representing the limiting (stationary) distribution as well $P(x'(t_0+t)) = \sum_{x \in \mathcal{X}} P(x(t_0)) \mathbf{T}_{x,x'}(t) = (\pi \mathbf{T}(t))_{x'} = \pi_{x'}$. This means that after thermalisation we can continuously take measurements $A_{x(t)}$ for a time interval Δt and use them for calculating an estimator of the expectation value $\langle A \rangle = \frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} A_{x(t)} dt$.

The Markov chain consists of right-continuous step functions at certain jump-times t_i with $i \in \mathbb{N}$ (see theorem 1.6). The measurements start after thermalisation at time t_0 and we can define the retention time $\Delta t_i = t_{i+1} - t_i$ for $i = \{0, 1, 2, \ldots, N_M - 1\}$ as the time in which the Markov chain stays in a certain state $x_i = x(t_i)$. The number of Markov jumps after thermalisation is N_M. With this the estimator of the expectation values gets

$$\langle A \rangle = \frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} A_{x(t)} dt = \frac{1}{\Delta t} \sum_{i=0}^{N_M - 1} A_{x(t_i)} \Delta t_i \tag{1.42}$$

The estimator of an expectation value is worth nothing without an error. Assuming that our Markov chain generates independent random variables out of our probability distribution π, the error could be evaluated from the variance $var(A)$ of the observable A.

$$var(A) = \left\langle (A - \langle A \rangle)^2 \right\rangle = \frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} \left(A_{x(t)} - \langle A \rangle \right)^2 dt = \cdots = \left\langle A^2 \right\rangle - \langle A \rangle^2 \tag{1.43}$$

From an unbiased variance $var(A)$ we usually get an estimator of the error of our expectation value with $\Delta A = \sqrt{\frac{var(A)}{N_m}}$ where N_m is the number of measurements. The unbiased variance can be shown to be $var(A) = \frac{N_m}{N_m - 1} \left(\left\langle A^2 \right\rangle - \langle A \rangle^2 \right)$ if the estimator of the expectation value $\langle A \rangle$ (1.42) is taken to calculate the variance. So we get

$$\Delta A = \sqrt{\frac{\langle A^2 \rangle - \langle A \rangle^2}{N_m - 1}} \tag{1.44}$$

for the estimator of the error ΔA.

Autocorrelation Function:

In our case of a continuous-time Markov chain, we are, in principle, measuring continuously. Thus it is not trivial to quantify the number of measurements N_m. To be able to provide N_m, we have to think about the meaning of independent random variables. If we are in a state $x(t)$ at time t, the state $x(t+dt)$ occupied an infinite amount of time dt later is presumably dependent of $x(t)$. We have to wait at least as long as the average time needed to access a new state. In this case, the number of measurements would be the number of Markov jumps that the chain performed $N_m = N_M$. Due to the Markov property, the chain is linked with its past and hence the state before and after a jump are not independent. The strength of the correlation between those two states depends on the way that the jumps are suggested. Practically it is not possible to create a totally uncorrelated Markov chain. For independent random variables, a new state would have to be chosen independent of the prior state directly from the probability distribution π.

This illustrates that our Markov chain cannot generate completely independent random variables. This is where autocorrelations come into play.

An important quantity to determine the dependence of measurements taken at different times of the same Markov chain is the normalised **autocorrelation function** $\rho(t)$. For an observable A it is defined as

$$\rho_A(t) := \frac{cov\left(A_{x(t')}, A_{x(t'+t)}\right)}{\sqrt{var(A_{x(t')}) \cdot var(A_{x(t'+t)})}} = \frac{cov(A_{x(t_0)}, A_{x(t_0+t)})}{var(A)} \tag{1.45}$$

where homogeneity in time was assumed after thermalisation $t' \geq t_0$. The covariance $cov(B,C)$ of two observables B and C is given by

$$cov(B,C) = \langle (B - \langle B \rangle)(C - \langle C \rangle) \rangle . \tag{1.46}$$

A numerically very stable estimator of the autocorrelation function is the so-called empirical autocorrelation function $\rho^E(t)$ [15]. For a continuous-time Markov chain it can be evaluated in the following way:

$$\rho_A^E(t) = \frac{\frac{1}{\Delta t - t}\int_{t_0}^{t_0+\Delta t-t} \left(A_{x(t')} - \langle A \rangle_t\right)\left(A_{x(t'+t)} - \langle A \rangle_t'\right) dt'}{\sqrt{\frac{1}{\Delta t - t}\int_{t_0}^{t_0+\Delta t-t} \left(A_{x(t')} - \langle A \rangle_t\right)^2 dt' \frac{1}{\Delta t - t}\int_{t_0}^{t_0+\Delta t-t} \left(A_{x(t'+t)} - \langle A \rangle_t'\right)^2 dt'}} \tag{1.47}$$

$$= \frac{\frac{1}{\Delta t - t}\left(\int_{t_0}^{t_0+\Delta t-t} A_{x(t')}A_{x(t'+t)} dt'\right) - \langle A \rangle_t \langle A \rangle_t'}{\sqrt{\left(\langle A^2 \rangle_t - \langle A \rangle_t^2\right)\left(\langle A^2 \rangle_t' - \left(\langle A \rangle_t'\right)^2\right)}} \tag{1.48}$$

with

$$\langle A \rangle_t = \frac{1}{\Delta t - t}\int_{t_0}^{t_0+\Delta t-t} A_{x(t')} dt' \tag{1.49}$$

$$\langle A^2 \rangle_t = \frac{1}{\Delta t - t}\int_{t_0}^{t_0+\Delta t-t} A_{x(t')}^2 dt' \tag{1.50}$$

$$\langle A \rangle_t' = \frac{1}{\Delta t - t}\int_{t_0}^{t_0+\Delta t-t} A_{x(t'+t)} dt' \tag{1.51}$$

$$\langle A^2 \rangle_t' = \frac{1}{\Delta t - t}\int_{t_0}^{t_0+\Delta t-t} A_{x(t'+t)}^2 dt' \tag{1.52}$$

From the simulated continuous-time Markov chain $\{x(t) = x(t_i), t_i \leq t < t_{i+1} \ \forall i < N_M\}$, where N_M is the number of jumps the simulation stepped through after thermalisation, the empirical autocorrelation function can be evaluated directly. Due to the step function nature of the Markov chain, the integrals can be converted to sums as in (1.42). E.g., (1.49) turns into:

$$\langle A \rangle_t = \frac{1}{\Delta t - t}\int_{t_0}^{t_0+\Delta t-t} A_{x(t')} dt' = \frac{1}{\Delta t - t}\left(\sum_{i=0}^{N_M'-1} A_{x(t_i)}\Delta t_i + A_{x(t_{N_M'})}\left(\Delta t - t - t_{N_M'}\right)\right), \tag{1.53}$$

where the variable N_M' is the highest integer for which the relation $t_{N_M'} \leq \Delta t - t$ holds. Similar relations can be formulated for all other expectation values needed to calculate $\rho_A^E(t)$. Due to the integration (considering continuous observables A only depending on the current state of the Markov chain $A_{x(t)}$), the step functions are smoothed out and our empirical autocorrelation function is a smooth and piecewise differentiable function of time. For an observable B depending on the transition $B_{x(t),x(t+dt)}$, e.g., a velocity, the observable only takes non-zero values during the immediate transition. As a consequence, Dirac-delta-functions appear in $B_{x(t),x(t+dt)}$ and the integral is only piecewise smooth with steps appearing in $\rho_B^E(t)$. Although this evaluation of $\rho^E(t)$ is the correct one and leads to the accurate time dependence in continuous Markov time, it is quite cumbersome.

An easier approach is to treat the continuous-time Markov chain $\{x(t) = x(t_i), t_i \leq t < t_{i+1} \ \forall i < N_M\}$ as if it would be a discrete-time Markov chain $\{x_i, i \in \{0,1,\ldots,N_M-1\}\}$. Calculating the empirical autocorrelation function $\rho_A^E(i)$ of an observable $A_i = A_{x_i}$ according to the definition (1.45) for a discrete-time Markov chain, is reminiscent to a continuous-time Markov chain, but the final expression is easier to handle:

$$\rho_A^E(i) = \frac{\frac{1}{N_M-i}\left(\sum_{j=0}^{N_M-1-i} A_j A_{j+i}\right) - \langle A\rangle_i \langle A\rangle_i'}{\sqrt{\left(\langle A^2\rangle_i - \langle A\rangle_i^2\right)\left(\langle A^2\rangle_i' - \left(\langle A\rangle_i'\right)^2\right)}} \tag{1.54}$$

with

$$\langle A\rangle_i = \frac{1}{N_M-i}\sum_{j=0}^{N_M-1-i} A_j \tag{1.55}$$

$$\langle A^2\rangle_i = \frac{1}{N_M-i}\sum_{j=0}^{N_M-1-i} A_j^2 \tag{1.56}$$

$$\langle A\rangle_i' = \frac{1}{N_M-i}\sum_{j=0}^{N_M-1-i} A_{j+i} \tag{1.57}$$

$$\langle A^2\rangle_i' = \frac{1}{N_M-i}\sum_{j=0}^{N_M-1-i} A_{j+i}^2 \tag{1.58}$$

Observables $B_{x_i,x_{i+1}}$ depending on the transition $x_i \to x_{i+1}$ now also just have certain values $B_i = B_{x_i,x_{i+1}}$ for the currently present discrete Markov time i. Hence, they can be treated in exactly the same way as observables A_i that only depend on the current state x_i.

Our discrete-time empirical autocorrelation function $\rho_A^E(i)$ no longer gives us the correct time dependence in continuous Markov time, but rather tells us how correlated the measured values of the observable A for the individually visited states are. With this simplification it seems that we completely neglect the continuous-time dependence. Fortunately this is not the case as we still have the retention times, which can be seen as an observable Δt_i. So by looking at both, the discrete-time empirical autocorrelation function of the observable A and of the retention times Δt_i, we should get the same information. Nevertheless it would be more accurate to directly implement the continuous-time empirical autocorrelation function.

Autocorrelation Times:
The shape of the autocorrelation function $\rho_A(t)$ for a certain observable A is reflecting the behaviour of the Markov chain and holds information about the system and the observable A. So it is worthwhile having a more detailed look at it. In the following the time t is assumed to be discrete, but, if the notation is changed appropriate, the content also holds for continuous times t.

For the analytical autocorrelation function of an observable A as defined in (1.45), it can be shown that

$$\rho_A(t) = \sum_{i=1}^{N_{ac}} c_i(A) e^{-\frac{t}{\tau_i}} \tag{1.59}$$

with N_{ac} being the number of autocorrelation times τ_i (for a finite dimensional state space N_{ac} is finite) and corresponding positive coefficients $c_i(A)$, where the autocorrelation times are fixed by the transition matrix (or the transition function and hence the q-matrix in the continuous case) and the coefficients are depending on the observable A. Note that it is possible that for certain observables A some autocorrelation times τ_i are not present as the coefficients might be zero $c_i(A) = 0$.

The largest autocorrelation time for which the coefficient is non-zero, $c_{exp}(A) > 0$, is called the asymptotic autocorrelation time $\tau_{exp}(A)$ as it determines the asymptotic behaviour of the autocorrelation function $\lim_{t\to\infty} \rho_A(t) = c_{exp}(A) e^{-t/\tau_{exp}(A)}$. Additionally, it sets the time-scale for thermalisation as after a few

asymptotic autocorrelation times the Markov chain has more or less completely lost the information of the starting state.

Another important autocorrelation time is the integrated autocorrelation time $\tau_{int}(A)$:

$$\tau_{int}(A) = \frac{1}{2} + \sum_{t=1}^{N_M} \rho_A(t) \left(1 - \frac{t}{N_M}\right) \tag{1.60}$$

which can be calculated for a Markov chain having N_M links. With this integrated autocorrelation time an improved estimator of the error of an observable can be given:

$$\Delta A = \sqrt{\frac{var(A)}{N_M} 2\tau_{int}} \tag{1.61}$$

Compared to the case of independently generated random variables, where $\Delta A = \sqrt{\frac{var(A)}{N_m}}$, the number of measurements N_m is efficiently reduced to $\frac{N_M}{2\tau_{int}}$. This observation seems to answer the question some pages before: How 'correlated' are correlated random variables? From (1.61) one could interpret that after a time of $2\tau_{int}$ the measurements are independent.

For an autocorrelation function just having one dominating autocorrelation time $\rho_A(t) = e^{-t/\tau_{exp}}$ the integrated autocorrelation time τ_{int} is given by

$$\begin{aligned} \tau_{int} &= \frac{1}{2} + \sum_{t=1}^{N_M} e^{-\frac{t}{\tau_{exp}}} \left(1 - \frac{t}{N_M}\right) \approx -\frac{1}{2} + \sum_{t=0}^{N_M} \left(e^{-\frac{1}{\tau_{exp}}}\right)^t = -\frac{1}{2} + \frac{1 - \left(e^{-\frac{1}{\tau_{exp}}}\right)^{N_M+1}}{1 - e^{-\frac{1}{\tau_{exp}}}} && (1.62) \\ &\approx -\frac{1}{2} + \frac{1}{1 - \left(1 - \frac{1}{\tau_{exp}}\right)} = -\frac{1}{2} + \tau_{exp} \approx \tau_{exp} && (1.63) \end{aligned}$$

where $N_M \gg \tau_{exp} \gg 1$ was assumed. This shows that the integrated autocorrelation time τ_{int} and the asymptotic autocorrelation time τ_{exp} are related. For an autocorrelation function dominated only by τ_{exp}, they are approximately the same $\tau_{int} \approx \tau_{exp}$, provided that $N_M \gg \tau_{exp} \gg 1$.

Jackknife:

Autocorrelation times are important to be able to estimate the number of necessary jumps to get simulation results that have acceptable error bars, but the calculation of error bars itself is rather done with different methods. A very important one, and the one of choice here, is a method called Jackknife. In a first step, the data series (in our case the Markov chain) is divided into N_B blocks. The data series in two different blocks should be uncorrelated to assure that the estimator of the error is trustworthy. Finally we want to get an estimator and the corresponding error of a certain measure M. This measure could be the expectation value of an observable A or the fit to an autocorrelation function or any information that can be extracted out of the data series. Now the whole data series is analysed and yields to the measure $M^{(0)}$ which is the estimator of the measure. Next the data series is analysed another N_B times $\{i = 1,2,3,\ldots,N_B\}$ while in each step i we leave out block i and analyse the remaining data series to get the measure $M^{(i)}$. The average of those measures $M^{(av)} = \frac{1}{N_B}\sum_{i=1}^{N_B} M^{(i)}$ is calculated as well and the estimator of the error of M is given by

$$\Delta M = \sqrt{\frac{N_B - 1}{N_B} \sum_{i=1}^{N_B} \left(M^{(i)} - M^{(av)}\right)^2} \tag{1.64}$$

Compared to the estimator of the error for uncorrelated data, the error is increased by a factor of $(N_B - 1)$. This factor appears due to the fact that the measures $M^{(i)}$ always evaluate the same data set except one block i. Hence, the measures $M^{(i)}$ are highly correlated.

Jackknife can also be used to get an estimator of the so-called bias b to get an even better estimator of the measure M:

$$M = M^{(0)} - b \quad \text{with} \quad b = (N_B - 1)\left(M^{(av)} - M^{(0)}\right) \tag{1.65}$$

In our case, the bias is assumed to be much lower than the statistical error. Hence it plays a minor role and the correction is not done.

The most remarkable benefit of Jackknife is, that also estimators of errors for measures involving highly non-linear data processing like fits of autocorrelation functions can easily be performed. For each measure $M^{(i)}$ nearly the complete data series is used which leads to very stable values of $M^{(i)}$. As long as the block size is much larger than each of the autocorrelation times (especially $2\tau_{int}$), the estimator of the error is a very reliable value for the actual error.

1.1.5 Relation Between Physical and Markov Time

On the last pages we mathematically constructed a method to simulate a stochastic process (the continuous-time Markov chain) in which the only input is a q-matrix Q that contains certain rates $q_{x,x'} = R(x \to x')$. We have spent a lot of time to show that in this simulation we end up with a certain limiting distribution π that can be determined by the detailed balance equation $\pi_{x'} q_{x',x} = \pi_x q_{x,x'}$. Now it is time to think about the physical system that is related to this stochastic process and, hence, to have a look at the differential equation that describes the time evolution of our continuous-time Markov chain.

Master Equation

At a certain time t, our system is in a state given by the probability distribution row vector $\hat{\pi}(t)$ which holds the probability that a state $x \in \mathcal{X}$ is occupied at time t. Note that $\hat{\pi}$ can be any probability distribution vector satisfying $\hat{\pi}_x \geq 0 \ \ \forall x \in \mathcal{X}$ and $\sum_{x\in\mathcal{X}} \hat{\pi}_x = 1$. It has generally nothing to do with the stationary distribution π although for a long enough time $t \to \infty$ it will converge to it. When we start with a probability distribution vector $\hat{\pi}(0)$ at time $t = 0$, the probability distribution vector $\hat{\pi}(t)$ can be calculated by

$$\hat{\pi}_x(t) = \sum_{x'\in\mathcal{X}} P\left(x(t)|x'(0)\right)\hat{\pi}_{x'}(0) \tag{1.66}$$

The probability $P(x(t)|x'(0))$ was called the transition function $\mathbf{T}_{x',x}(t)$ and in matrix representation this equation simply writes as $\hat{\pi}(t) = \hat{\pi}(0)\mathbf{T}(t)$. Taking the time derivative of this equation leads to

$$\frac{d}{dt}\hat{\pi}(t) = \hat{\pi}(0)\frac{d}{dt}\mathbf{T}(t) = \hat{\pi}(0)\mathbf{T}(t)Q = \hat{\pi}(t)Q \tag{1.67}$$

where the Kolmogorov forward equation $\mathbf{T}'(t) = \mathbf{T}(t)Q$ was used. Equation (1.67) is already a form of the master equation. This is getting more obvious if we write the equation component-wise, replace the off-diagonal matrix elements by the rates $q_{x,x'} = R(x \to x')$ and use the definition of our conservative q-matrix to get the diagonal matrix elements $q_{x,x} = -\sum_{x'\in\mathcal{X}\setminus\{x\}} q_{x,x'}$:

$$\frac{d}{dt}\hat{\pi}_x(t) = \sum_{x'\in\mathcal{X}} \hat{\pi}_{x'}(t)q_{x',x} = \sum_{x'\in\mathcal{X}\setminus\{x\}} \hat{\pi}_{x'}(t)R(x' \to x) - \sum_{x'\in\mathcal{X}\setminus\{x\}} \hat{\pi}_x(t)R(x \to x') \tag{1.68}$$

This equation is called the master equation and shows that a change of each element of the probability distribution vector $\hat{\pi}_x(t)$ over time is happening as long as the probability flux $\sum_{x'\in\mathcal{X}\setminus\{x\}} \hat{\pi}_{x'}(t)R(x' \to x)$ into state x is not balanced with the probability flux $\sum_{x'\in\mathcal{X}\setminus\{x\}} \hat{\pi}_x(t)R(x \to x')$ out of state x. As we already know, this balance is only reached when the probability distribution vector converged to the stationary distribution $\hat{\pi} = \pi$. So we see that the time dependence of our simulation is the one of the master equation which is commonly used to describe physical systems.

Poisson Process and Physical Time

The last thing that we need to establish now is a relation between Markov time and physical time. So we first have to think about what gives us the physical time. We start from a physically motivated rate R which states that the probability that an event E is happening in an infinitesimal time interval dt is given by $P(E|dt) = Rdt$. This event can either happen or not, so the number of events $N_{event}(t)$ that happened in a certain time t can only take integer values, i.e., the infinitesimal change of $N_{event}(t)$ can only be 0 or 1: $dN \in \{0,1\}$. Such a stochastic process is called Poisson process. The probability $P(n,t) = P(N(t) = n)$ denotes the probability that at a certain time t the process has experienced n events E. To calculate the average time between two events we need the probability $P(0,t)$. We divide the interval $[0,t]$ into M subintervals of length $\Delta t_M = \frac{t}{M}$. The probability that there is an event happening in a subinterval for very short interval lengths is

$$\lim_{M\to\infty} P(E|\Delta t_M) = P(E|dt) = Rdt = \lim_{M\to\infty} R\Delta t_M \tag{1.69}$$

where $\lim_{M\to 0} \Delta t_M = dt$ is an infinitesimal time interval and we used the definition of the rate R. The probability that no event is happening in this infinitesimal time interval is $(1 - \lim_{M\to\infty} R\Delta t_M)$, provided that the time interval is short enough so that at most one event can happen in this interval. Two events happening at the same time are restricted. As the number of events in one interval does not effect the number of events in any other interval and probabilities of independent events E_1 and E_2 are multiplicative $P(E_1 \wedge E_2) = P(E_1)P(E_2)$, we can calculate the probability that no event is happening in a finite interval $[0,t]$:

$$P(0,t) = \lim_{M\to\infty} (1 - R\Delta t_M)^M = \lim_{M\to\infty} \left(1 - R\frac{t}{M}\right)^M = e^{-Rt} \tag{1.70}$$

The probability $p_E(t)dt$ that the first event is happening exactly at a time t is combining the probability $P(0,t)$ that nothing is happening up to time t and the probability Rdt that an event is happening in the infinitesimal time interval dt at time t: $p_E(t)dt = P(0,t)Rdt$. At any time this first event has to happen, so the probability density should be normed and indeed

$$\int_0^\infty p_E(t)dt = \int_0^\infty Re^{-Rt}dt = R\left(-\frac{1}{R}\right)\left(e^{-Rt}\right)\Big|_0^\infty = -(0-1) = 1 \tag{1.71}$$

With the probability density $p_E(t)$ we can calculate the expectation value

$$\langle t_E\rangle = \int_0^\infty t\cdot p_E(t)dt = \int_0^\infty tR\cdot e^{-tR}dt = \left(-t\cdot e^{-tR}\right)\Big|_0^\infty - \int_0^\infty -e^{-tR}dt = -\frac{1}{R}\left(e^{-tR}\right)\Big|_0^\infty = \frac{1}{R} \tag{1.72}$$

which shows us that the physical time-scale corresponding to the rate R is fixed by the average time between two events $\langle t_E\rangle = \frac{1}{R}$.

Considering multiple events $E_1, E_2, E_3, \ldots, E_n$ with corresponding rates $R_1, R_2, R_3, \ldots, R_n$ that can happen independent of each other in parallel, every event i has its own probability density $p_{E_i}(t) = R_i e^{-tR_i}$ for the first event. The overall rate $R^{(n)}$ is the sum of all individual rates $R^{(n)} = \sum_{i=1}^n R_i$. With all those competing processes, one has to trigger the first event. We call the time when the first event of all events happens t_{min}. Furthermore we call the first event that happens E_{min}. We write $p(t = t_{min})$ for the probability density that the first event happens at time t. Below we will show that $p(t = t_{min}) = R^{(n)}e^{-tR^{(n)}}$ and that the probability that E_i is the first event happening is $P(E_i = E_{min}) = \frac{R_i}{R^{(n)}}$. Comparing this result to the sample-path behaviour of our Markov chain (see theorem 1.6), we see that our physical process corresponds exactly to our Markov chain if we equalise the q-matrix with the corresponding physical rates $q_{x,x'} = R(x \to x')$. Given this equality, physical time and Markov time are the same and the time evolution of the Markov chain can be directly interpreted as the time evolution of a physical system obeying the master equation. This also implies that non-equilibrium properties of a system could also be studied with continuous-time Markov chains, as long as it is assured that the system converges to a unique equilibrium configuration. The term 'equilibrium configuration' for the physical system corresponds to the term 'stationary distribution' for the Markov chain.

Prove of Equivalence of the Two Sample-Path Behaviours

For convenience we define the set of numbers $\mathcal{N}_n = \{1,2,3,\ldots,n\}$. We start with assuming that $E_j = E_{min}$ which means that j is the fastest process and $t_j < t_i \ \forall i \in \mathcal{N}_n \setminus \{j\}$. The probability distribution $p(t_j|t_j = t_{min})$ of the time t_j given that it is the shortest time is supposed to be

$$p(t_j|t_j = t_{min}) = p(t_j|t_j < t_i \ \ \forall i \in \mathcal{N}_n \setminus \{j\}) = R^{(n)} e^{-R^{(n)} t_j}, \tag{1.73}$$

a relation that we will prove by mathematical induction. We start with the (trivial) basis for only one process $n = 1$

$$p(t_1|t_1 = t_{min}) = R_1 e^{-R_1 t_1} = R^{(1)} e^{-R^{(1)} t_1} \tag{1.74}$$

The step from n to $n+1$ is less trivial:

$$p(t_j|t_j < t_i \ \ \forall i \in \mathcal{N}_{n+1} \setminus \{j\}) = \\ \frac{P(t_j < t_{n+1}|t_j \wedge t_j < t_i \ \ \forall i \in \mathcal{N}_n \setminus \{j\}) p(t_j|t_j < t_i \ \ \forall i \in \mathcal{N}_n \setminus \{j\})}{\int_0^\infty P(t_j < t_{n+1}|t_j \wedge t_j < t_i \ \ \forall i \in \mathcal{N}_n \setminus \{j\}) p(t_j|t_j < t_i \ \ \forall i \in \mathcal{N}_n \setminus \{j\}) dt_j} \tag{1.75}$$

where the Bayes' theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1.76}$$

and marginalisation

$$P(A) = \int P(A|x)p(x)dx \tag{1.77}$$

was used. The functions constituting equation (1.75) can be evaluated separately:

$$\begin{aligned} P(t_j < t_{n+1}|t_j \wedge t_j < t_i \ \ \forall i \in \mathcal{N}_n \setminus \{j\}) &= \int_{t_j}^\infty p(t_{n+1}) dt_{n+1} && (1.78)\\ &= \int_{t_j}^\infty R_{n+1} e^{-t_{n+1} R_{n+1}} dt_{n+1} && (1.79)\\ &= e^{-t_j R_{n+1}} && (1.80) \end{aligned}$$

$$\int_0^\infty P(t_j < t_{n+1}|t_j \wedge t_j < t_i \ \ \forall i \in \mathcal{N}_n \setminus \{j\}) p(t_j|t_j < t_i \ \ \forall i \in \mathcal{N}_n \setminus \{j\}) dt_j = \\ = \int_0^\infty e^{-t_j R_{n+1}} R^{(n)} e^{-t_j R^{(n)}} dt_j = \frac{R^{(n)}}{R_{n+1} + R^{(n)}} = \frac{R^{(n)}}{R^{(n+1)}} \tag{1.81}$$

Combining those results concludes our mathematical induction

$$\begin{aligned} p(t_j|t_j < t_i \ \ \forall i \in \mathcal{N}_{n+1} \setminus \{j\}) &= \frac{e^{-t_j R_{n+1}} R^{(n)} e^{-t_j R^{(n)}}}{\frac{R^{(n)}}{R^{(n+1)}}} && (1.82)\\ &= R^{(n+1)} e^{-t_j \left(R_{n+1} + R^{(n)}\right)} && (1.83)\\ &= R^{(n+1)} e^{-t_j R^{(n+1)}}. && (1.84) \end{aligned}$$

With the help of marginalisation, this result can be used to calculate the desired probability density $p(t = t_{min})$:

$$\begin{aligned} p(t = t_{min}) &= \sum_{j=1}^n p(t = t_{min}|t_j = t_{min}) P(t_j = t_{min}) && (1.85)\\ &= \sum_{j=1}^n R^{(n)} e^{-tR^{(n)}} P(t_j = t_{min}) && (1.86)\\ &= R^{(n)} e^{-tR^{(n)}} \sum_{j=1}^n P(t_j = t_{min}) && (1.87)\\ &= R^{(n)} e^{-tR^{(n)}} && (1.88) \end{aligned}$$

with $\sum_{j=1}^{n} P(t_j = t_{min}) = 1$ because one of the events E_j has to be the first. Now we have proven that $p(t = t_{min}) = R^{(n)} e^{-tR^{(n)}}$ is correct, so let's continue with proving $P(E_i = E_{min}) = \frac{R_i}{R^{(n)}}$. The probability that the event E_i is the first that happens is equivalent to the probability that the time when event E_i happens is the shortest time t_{min}.

$$P(E_i = E_{min}) = P(t_i = t_{min}) = \int_0^\infty P(t_i = t_{min}|t_1) p(t_1) dt_1 = \cdots = \tag{1.89}$$

$$= \int \cdots \int_{\mathbb{R}^{n+}} P\left(t_i = t_{min} | \{t_j\} \ \ \forall j \in \mathcal{N}_n\right) \prod_{j \in \mathcal{N}_n} p(t_j) dt_j \tag{1.90}$$

The probability that t_i is the lowest time t_{min}, provided that all times $t_j \ \ \forall j \in \mathcal{N}_n$ have a specified value, is simply either unity if it is the minimum time or zero if not. With the help of the Heaviside function

$$\Theta(t) = \begin{cases} 0 & \text{for } t < 0 \\ 1 & \text{for } t \geq 0 \end{cases} \tag{1.91}$$

it can be written as

$$P\left(t_i = t_{min} | \{t_j\} \ \ \forall j \in \mathcal{N}_n\right) = \prod_{j \in \mathcal{N}_n \setminus \{i\}} \Theta\left(t_j - t_i\right) \tag{1.92}$$

With this we can continue the evaluation of $P(E_i = E_{min})$

$$P(E_i = E_{min}) = \int \cdots \int_{\mathbb{R}^{n+}} \left(\prod_{j \in \mathcal{N}_n \setminus \{i\}} \Theta(t_j - t_i) p(t_j) dt_j \right) p(t_i) dt_i \tag{1.93}$$

$$= \int_0^\infty \left(\prod_{j \in \mathcal{N}_n \setminus \{i\}} \int_{t_i}^\infty R_j e^{-t_j R_j} dt_j \right) R_i e^{-t_i R_i} dt_i \tag{1.94}$$

$$= \int_0^\infty \left(\prod_{j \in \mathcal{N}_n \setminus \{i\}} R_j \left(-\frac{1}{R_j} \right) \left(e^{-t_j R_j} \right) \Big|_{t_i}^\infty \right) R_i e^{-t_i R_i} dt_i \tag{1.95}$$

$$= \int_0^\infty \left(\prod_{j \in \mathcal{N}_n \setminus \{i\}} e^{-t_i R_j} \right) R_i e^{-t_i R_i} dt_i = \int_0^\infty R_i e^{-t_i \left(\sum_{j \in \mathcal{N}_n} R_j \right)} dt_i \tag{1.96}$$

$$= \int_0^\infty R_i e^{-t_i R^{(n)}} dt_i = -\frac{R_i}{R^{(n)}} \left(e^{-t_i R^{(n)}} \right) \Big|_0^\infty = \frac{R_i}{R^{(n)}} \tag{1.97}$$

So we were able to show that $P(E_i = E_{min}) = \frac{R_i}{R^{(n)}}$ for which indeed the relation $\sum_{i \in \mathcal{N}_n} P(E_i = E_{min}) = \frac{1}{R^{(n)}} \sum_{i \in \mathcal{N}_n} R_i = 1$ is fulfilled. Of course this relation must be fulfilled as one event must be the first.

1.2 The Model System

In this section the model system itself is introduced. It is shown that it fulfils the mathematical requirements so that the simulations lead to proper results and the physical motivation behind this model system is discussed. A very appropriate paper for charge transport in general, where also a lot of information provided in this section is taken out, is [1]. The kinetic Monte Carlo (KMC) method we implemented can be found in literature many times like in [16–19].

The first important requirement of the physical system is, that it can be described with a Markov process at all. As a reminder, the Markov property tells us that the system immediately forgets its past history after a transition. Thinking of a classical trajectory of a particle, this could never be the case because, given the initial conditions and the potential energy surface, the trajectory of the particle is determined for all times. So it never forgets where it came from. For a quantum-mechanical particle things are different. Assuming that a quantum-mechanical particle is undergoing a transition from one state to another, driven e.g. by the absorption of a photon, it needs some time to relax into a final state. But when

this relaxation process is finished, the particle looses its memory of the past. For classical systems that are behaving chaotically, loosing memory is also observed as incredibly tiny changes in the initial and boundary conditions can change the trajectory of the system dramatically. This means that we need a stochastic system in which the relaxation times (the times in which the system looses the memory of the prior state) are happening on a much shorter time-scale than the transition times (the times in which the system changes from one state to another). This is satisfied for charge carriers hopping in an organic semiconductor.

1.2.1 The State Space

We start with dividing our real space into finite volumes. Each volume is assigned a point inside the volume, this could, e.g., be the middle of it. With this points we end up with a discrete real space. Usually those volumes are cubes and so the points describe a simple cubic lattice. Each volume can host at most one charge carrier. Below we refer to the structure behind this volume as a cell. The position of a cell is referred to the point in our discrete real space of the corresponding volume, the size of the cell is the volume itself, and the edge length of a cubic cell will be called lattice constant. All currently known organic semiconducting materials have in common that their molecular building blocks have π-orbitals delocalised across the molecule or a segment of it. The concept of cells that can be occupied by charge carriers is physically inspired by the fact that the highest occupied molecular orbital (HOMO) and/or the lowest unoccupied molecular orbital (LUMO) is delocalised over a certain volume according to the size of the cell. For an electron conducting material the LUMO has to be a delocalised π-orbital and for a hole conducting material the HOMO. In our simulations we only consider one type of charge carriers, so either electrons or holes.

Now we have cut our organic semiconductor into cells that are arranged in a simple cubic lattice and each cell can house at most one charge carrier. The size of the cell is approximately the size of the molecular orbital that the charge carrier occupies in the organic semiconducting material. Typically an edge length in the order of 1 nm is chosen.

In a perfectly crystalline organic semiconductor, all molecules would align in the same way and all energy levels of the HOMOs and LUMOs of the molecules would be the same. In this case the well known Bloch's theorem would be applicable and the eigenstates of the Hamilton operator would be bands delocalised over the entire crystal. With our KMC approach, we assume that the charge carrier is localised within our cell and not delocalised over the whole simulation volume. So as long as there are no other interactions that cause localised eigenstates (e.g. strong phonon interactions), KMC cannot be used to physically interpretable simulate organic semiconductors composed of molecular crystals. If the molecules or polymers do not form a crystalline structure but rather assemble randomly, an energetic disorder is induced because the energy levels of a molecule (or a segment of a polymer) are influenced by their surrounding. Energetic disorder is known to lead to a localisation of the eigenstates, depending on the dimensionality this localisation is stronger or weaker (see [20] chapters 8.7, 9.9, 10.10 and 13). As a consequence, a disordered organic semiconductor is suited quite well to be mapped onto a KMC simulation. Due to the localisation of the eigenstates, the assumption that a cell can be either occupied or unoccupied becomes a physical foundation. Additionally our Markov chain, which in principle simulates a classical system, is also valid for quantum-mechanical systems where the states that are simulated are the eigenstates of the Hamilton operator (see chapter 1.1.1). So the higher the disorder, the better the physical system is represented by our KMC simulation.

In the simulation, the disorder is introduced by assigning an energy level ε_i to each cell i. Without disorder we would get $\varepsilon_i = 0$ for all cells i. As disorder has a stochastic nature, we have to provide a certain probability density $p(\varepsilon)$ which gives us the probability $P(\varepsilon_i = \varepsilon) = p(\varepsilon)d\varepsilon$ that the energy level ε_i of a cell i is given by ε. In practice, this probability distribution is often chosen to be a Gaussian

$$p(\varepsilon) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\varepsilon^2}{2\sigma^2}} \tag{1.98}$$

centred at $\langle \varepsilon \rangle = 0$ with the so-called energetic disorder σ. Exponential probability distributions are also in use. Likewise, an experimentally evaluated probability distribution could be considered.

Up till now we have mapped our physical system to a state space in which each state is one possibility of placing N_{cc} charge carriers into N_{cells} cells where the number of charge carriers can be either fixed or assume an integer number between 0 and N_{cells}, $N_{cc} \in \{0,1,2,\ldots,N_{cells}\}$. The $\sum_{n=0}^{N_{cells}} \frac{N_{cells}!}{N_{cc}!(N_{cells}-N_{cc})!} = 2^{N_{cells}}$ dimensional state space is incredibly huge, but still it is finite dimensional (for a fixed number of charge carriers it is 'only' $\frac{N_{cells}!}{N_{cc}!(N_{cells}-N_{cc})!}$).

1.2.2 The Rates

To further establish a continuous-time Markov Chain Monte Carlo method, we need the q-matrix, i.e., we need rates. We only allow one charge carrier to move during a transition, so a certain charge carrier could move from its current cell i to a cell j within one transition. With this the irreducibility of our Markov process is already assured, as, (i), any charge carrier can move to any cell within a finite time and, (ii), all states of the state space are communicating. There are two commonly employed rates that are associated to those transitions, the Marcus rate [21] and the Miller-Abrahams rate [22]. The physical interpretation of those rates will be delivered below, but at the moment we are interested in proving the conceptual correctness of the method. The Marcus rate $R^M(i,j)$ for the hop of a charge carrier from cell i to cell j is given by

$$R^M(i,j) = \nu_0 e^{-2\alpha r_{ij}} \exp\left(-\frac{(\Delta E_{ij} + E_r)^2}{4E_r k_B T}\right) \tag{1.99}$$

and the Miller-Abrahams rate $R^{MA}(i,j)$ is

$$R^{MA}(i,j) = \nu_0 e^{-2\alpha r_{ij}} \begin{cases} \exp\left(-\frac{\Delta E_{ij}}{k_B T}\right) & \text{for } \Delta E_{ij} > 0 \\ 1 & \text{for } \Delta E_{ij} \leq 0. \end{cases} \tag{1.100}$$

Both rates are consisting of three parts: the hopping prefactor ν_0, a spatial decay term $e^{-2\alpha r_{ij}}$ and a term determined by the difference between the total energy of the system E^{tot} where the charge carrier is occupying cell j and cell i respectively $\Delta E_{ij} = E^{tot}(\text{charge carrier at cell } j) - E^{tot}(\text{charge carrier at cell } i)$. The hopping prefactor, in principle, chooses the time-scale and, as it appears linear in the rates and hence in time, it can be easily changed after the simulation without affecting any measured observable except the time (and observables involving the time). Those affected observables only have to be multiplied by a constant factor when changing ν_0. From a physical point of view, ν_0 tells us how fast a transition can be undergone if the spatial and energetic damping factor, that are both decreasing the rate, are neglected. The spatial decay term $e^{-2\alpha r_{ij}}$ includes the charge delocalisation constant α and the distance between the two cells $r_{ij} = |\vec{r}_i - \vec{r}_j|$ with the positions of the cells $\vec{r}_i$ and $\vec{r}_j$. This term takes into account that hops over larger distances are less likely. As tunnelling is known to have an exponential dependence of the tunnelling distance, this term is presumed to adopt an exponential function of the distance between the two cells. When looking at the two rate equations (1.99) and (1.100), they evidently only differ in the energetic term. Of course, in both expressions the product of the Boltzmann constant k_B and the temperature T is present, as this product $k_B T = \frac{1}{\beta}$ dictates the energy scale. But that's it with the similarities. In the Miller-Abrahams rate equation all hops downwards in energy are equiprobable and the hops upwards in energy are damped by a Boltzmann factor $e^{-\beta \Delta E}$. In contrast, for Marcus rates we have a Gauss peak centred at the negative reorganisation energy $-E_r$ which means that hops downwards in energy, for which the energy difference is exactly the reorganisation energy $\Delta E = -E_r$, are the fastest and all hops for other energy differences are slower. For small positive or negative energy differences in the order of the reorganisation energy, the quantitative behaviour of the two rates is quite similar. On the other hand, for high negative energy differences the rates obviously disagree. The reorganisation energy is the energy barrier that has to be overcome to get from one cell to another and it corresponds to twice

the polaronic binding energy (see later).

Except the energy, all variables constituting the rates are clear. To continue proving the conceptual correctness of the method, we need to know those energies. The total energy of the system with N_{cc} charge carriers occupying cells $k_1, k_2, \ldots, k_{N_{cc}}$ is given by

$$E^{tot} = E^{\text{cells}} + E^{\text{field}} + E^{\text{interaction}} + E^{\text{own image charge}} + E^{\text{image charge interaction}} \tag{1.101}$$

for a simulation with a metal contact present at one side of the organic semiconductor, or, in other words, an injection simulation. For a bulk simulation with an organic semiconductor only and no metal surface, only the first three terms are needed to get the total energy of the system.

The first term is the energy that is given to the N_{cc} charge carriers by occupying the cells.

$$E^{\text{cells}} = \sum_{l=1}^{N_{cc}} \varepsilon_{k_l} \tag{1.102}$$

An externally applied electric field $\vec{F}$ leads to the energy

$$E^{\text{field}} = \sum_{l=1}^{N_{cc}} q\vec{F}\vec{r}_{k_l} \tag{1.103}$$

where $\vec{r}_{k_l}$ is the position of cell k_l which is occupied by charge carrier l and the charge is $q = \pm e$ depending on whether holes or electrons are the simulated charge carriers (e is the elementary charge). The Coulomb interaction energy is given by

$$E^{\text{interaction}} = \sum_{l=1}^{N_{cc}} \sum_{m=l+1}^{N_{cc}} \frac{e^2}{4\pi\varepsilon_0\varepsilon_r |\vec{r}_{k_l} - \vec{r}_{k_m}|} = \frac{1}{2}\sum_{l=1}^{N_{cc}} \sum_{\substack{m=1 \\ m\neq l}}^{N_{cc}} \frac{e^2}{4\pi\varepsilon_0\varepsilon_r |\vec{r}_{k_l} - \vec{r}_{k_m}|} \tag{1.104}$$

with the vacuum permittivity ε_0 and the relative permittivity ε_r. In the summation, double counting has to be prevented. As soon as a metal contact is present, contributions to the energy from the own image charge [23]

$$E^{\text{own image charge}} = -\sum_{l=1}^{N_{cc}} \frac{e^2}{16\pi\varepsilon_0\varepsilon_r d_{k_l}} \tag{1.105}$$

and the image charges of all other charge carriers

$$E^{\text{image charge interaction}} = -\sum_{l=1}^{N_{cc}} \sum_{m=l+1}^{N_{cc}} \frac{e^2}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}_{k_l} - \vec{r}^{\,img}_{k_m}\right|} = -\frac{1}{2}\sum_{l=1}^{N_{cc}} \sum_{\substack{m=1 \\ m\neq l}}^{N_{cc}} \frac{e^2}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}_{k_l} - \vec{r}^{\,img}_{k_m}\right|} \tag{1.106}$$

have to be considered in which double counting has to be prevented for the evaluation of the Coulomb interaction with the image charges as well. The minimum distance of the cell k_l occupied by charge carrier l to the metal contact is given by d_{k_l}. The variable $\vec{r}^{\,img}_{k_m}$ is the position of the image cell of cell k_m where the image charge of charge carrier m is found.

From this total energy E^{tot}, the change in energy when a charge carrier is moving from cell i to cell j can be calculated:

$$\Delta E_{ij} = E^{tot}(\text{charge carrier at cell } j) - E^{tot}(\text{charge carrier at cell } i) \tag{1.107}$$

$$\begin{aligned} = {} & \varepsilon_j - \varepsilon_i + q\vec{F}(\vec{r}_j - \vec{r}_i) + \sum_{\substack{l=1 \\ k_l\neq j}}^{N_{cc}} \frac{e^2}{4\pi\varepsilon_0\varepsilon_r |\vec{r}_j - \vec{r}_{k_l}|} - \sum_{\substack{l=1 \\ k_l\neq i}}^{N_{cc}} \frac{e^2}{4\pi\varepsilon_0\varepsilon_r |\vec{r}_i - \vec{r}_{k_l}|} \\ & - \frac{e^2}{16\pi\varepsilon_0\varepsilon_r d_j} + \frac{e^2}{16\pi\varepsilon_0\varepsilon_r d_i} - \sum_{\substack{l=1 \\ k_l\neq j}}^{N_{cc}} \frac{e^2}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}_j - \vec{r}^{\,img}_{k_l}\right|} + \sum_{\substack{l=1 \\ k_l\neq i}}^{N_{cc}} \frac{e^2}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}_i - \vec{r}^{\,img}_{k_l}\right|} \end{aligned} \tag{1.108}$$

The cells of the charge carriers are again labelled by $\{k_1, k_2, \ldots, k_{N_{cc}}\}$ and symmetry was used for interactions $|r_j - r_{k_l}| = |r_{k_l} - r_j|$ and image charge interactions $\left|r_j - r_{k_l}^{img}\right| = \left|r_{k_l} - r_j^{img}\right|$. All terms, in which the moving charge carrier does not appear, cancel out. So we can define an energy

$$E_i = \varepsilon_i + q\vec{F}\vec{r}_i + \sum_{\substack{l=1 \\ k_l \neq i}}^{N_{cc}} \frac{e^2}{4\pi\varepsilon_0\varepsilon_r |\vec{r}_i - \vec{r}_{k_l}|} - \frac{e^2}{16\pi\varepsilon_0\varepsilon_r d_i} - \sum_{\substack{l=1 \\ k_l \neq i}}^{N_{cc}} \frac{e^2}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}_i - \vec{r}_{k_l}^{img}\right|} \tag{1.109}$$

which is the energy needed to place a new charge carrier at cell i and, as the difference in total energy, when a charge carrier is hopping from cell i to cell j, is the same as taking a charge carrier at cell i out of the simulation and insert a new one at cell j, the difference in total energy is

$$\Delta E_{ij} = E_j - E_i \tag{1.110}$$

Note that the total energy of the system is not given by $E^{tot} \neq \sum_{i=1}^{N_{cc}} E_i$ because to get the correct total energy of the system one would have to place the charge carriers one after the other starting with an empty system and hence all interactions are erroneously double counted in $\sum_{i=1}^{N_{cc}} E_i$.

By knowing the energies and consequently the rates, we can feed the q-matrix with this information. As a reminder, we have to show that the q-matrix is conservative, finite dimensional, stable (and hence uniformly bounded) and irreducible. The dimensionality $d \leq 2^{N_{cells}}$ with N_{cells} being the number of cells was already shown and we already discussed that the chain is irreducible.

To ensure that the q-matrix is conservative, we simply have to take the total rate $R^{(N)} = \sum_{i=1}^{N} R_i$ for each Markov jump to evolve the system in time, where R_i are all rates of the transitions allowed in the current state and N is the number of those transitions. So the fact that our q-matrix is conservative is guaranteed by implementing the method in a correct way as described in theorem 1.6.

Next we want to show that the q-matrix is stable. Having a look at the rates we see that any rate of any transition cannot be larger than ν_0. So it holds that $q_{x,x'} \leq \nu_0 \ \forall x, x' \in \mathcal{X}$ with $x \neq x'$ and $q_x = \sum_{x' \in \mathcal{X}\setminus\{x\}} q_{x,x'} \leq \nu_0 \sum_{x' \in \mathcal{X}\setminus\{x\}} 1 \leq \nu_0 2^{N_{cells}} < \infty$ which shows that Q is stable and uniformly bounded. To finish the considerations about the correctness of our method, we will show that for both rates the stationary distribution, which is also the limiting distribution of our Markov chain, is given by the Boltzmann statistics

$$\pi_x = \frac{1}{Z} e^{-\beta E_x^{tot}} \ \forall x \in \mathcal{X} \quad \text{with} \quad Z = \sum_{x \in \mathcal{X}} e^{-\beta E_x^{tot}} \tag{1.111}$$

where E_x^{tot} is the total energy of the system being in state x. Due to $\pi_x \geq 0$ and $\sum_{x \in \mathcal{X}} \pi_x = 1$ the row vector π is a probability distribution and by showing that it satisfies the detailed balance equation (1.40) it is ensured that the Markov chain indeed tends towards the probability distribution π in the limiting behaviour (see theorem 1.5). We have to prove that $\pi_{x'} q_{x',x} = \pi_x q_{x,x'} \ \forall x, x' \in \mathcal{X}$ so we have to think about allowed transitions. The matrix element $q_{x,x'}$ is only non-zero if the two states only differ in the position of one charge carrier. All the other rates are zero and the detailed balance equation is fulfilled trivially for those. So we consider two states x_i and x_j that differ only in the position of one charge carrier, i.e., in x_i a charge carrier is occupying cell i and cell j is empty while in x_j it is the other way round. The position of all the other charge carriers stays the same, no matter if there are 0 or $N-2$ remaining charge carriers present. An important remark at this point is, that the allowed transitions have to be chosen symmetric which means that if a charge carrier is allowed to hop from cell i to cell j, a hop from cell j to cell i has to be allowed too. This being provided, the two entries in the q-matrix are given by $q_{x_i,x_j} = R^{M \text{ or } MA}(i,j)$ and $q_{x_j,x_i} = R^{M \text{ or } MA}(j,i)$. The fact that the detailed balance equation has to be fulfilled for all $x, x' \in \mathcal{X}$ is equivalent to the requirement that it has to be fulfilled for all $x_i, x_j \in \mathcal{X}$ and for all cells $i, j \in \{1, 2, \ldots, N_{cells}\}$. All the other combinations of x and x' that are not covered with x_i and x_j are not allowed and hence anyway fulfilled.

We start with the Marcus rate:

$$\pi_{x_i} q_{x_i,x_j} = \pi_{x_j} q_{x_j,x_i} \tag{1.112}$$

$$\frac{1}{Z} e^{-\beta E^{tot}_{x_i}} R^M(i,j) = \frac{1}{Z} e^{-\beta E^{tot}_{x_j}} R^M(j,i) \tag{1.113}$$

$$\frac{1}{Z} e^{-\beta E^{tot}_{x_i}} \nu_0 e^{-2\alpha r_{ij}} \exp\left(-\frac{(\Delta E_{ij}+E_r)^2}{4E_r k_B T}\right) = \frac{1}{Z} e^{-\beta E^{tot}_{x_j}} \nu_0 e^{-2\alpha r_{ji}} \exp\left(-\frac{(\Delta E_{ji}+E_r)^2}{4E_r k_B T}\right) \tag{1.114}$$

We use $r_{ji} = r_{ij}$ and $\Delta E_{ji} = -\Delta E_{ij}$ and remove all terms obviously fulfilling equality

$$e^{-\beta E^{tot}_{x_i}} \exp\left(-\beta \frac{\Delta E_{ij}^2 + 2\Delta E_{ij} E_r + E_r^2}{4E_r}\right) = e^{-\beta E^{tot}_{x_j}} \exp\left(-\beta \frac{\Delta E_{ij}^2 - 2\Delta E_{ij} E_r + E_r^2}{4E_r}\right) \tag{1.115}$$

$$e^{-\beta E^{tot}_{x_i}} \exp\left(-\beta \frac{2\Delta E_{ij} E_r}{4E_r}\right) = e^{-\beta E^{tot}_{x_j}} \exp\left(\beta \frac{2\Delta E_{ij} E_r}{4E_r}\right) \tag{1.116}$$

$$e^{-\beta E^{tot}_{x_i}} \exp\left(-\beta \frac{\Delta E_{ij}}{2}\right) = e^{-\beta E^{tot}_{x_j}} \exp\left(\beta \frac{\Delta E_{ij}}{2}\right) \tag{1.117}$$

$$e^{-\beta\left(E^{tot}_{x_i} + \Delta E_{ij}\right)} = e^{-\beta E^{tot}_{x_j}} \tag{1.118}$$

which is fulfilled, as ΔE_{ij} was introduced as

$$\Delta E_{ij} = E^{tot}(\text{charge carrier at cell } j) - E^{tot}(\text{charge carrier at cell } i) = E^{tot}_{x_j} - E^{tot}_{x_i}.$$

For the Miller-Abrahams rate, the procedure is exactly the same with the slight difference that we have to distinguish between $\Delta E_{ij} > 0$, $\Delta E_{ij} < 0$ and $\Delta E_{ij} = 0$. All terms containing no energies vanish like for the Marcus rate and we end up with

$$\pi_{x_i} q_{x_i,x_j} = \pi_{x_j} q_{x_j,x_i} \tag{1.119}$$

$$\frac{1}{Z} e^{-\beta E^{tot}_{x_i}} R^{MA}(i,j) = \frac{1}{Z} e^{-\beta E^{tot}_{x_j}} R^{MA}(j,i) \tag{1.120}$$

$$e^{-\beta E^{tot}_{x_i}} \exp\left(-\frac{\Delta E_{ij}}{k_B T}\right) = e^{-\beta E^{tot}_{x_j}} \qquad \text{for} \quad \Delta E_{ij} > 0, \tag{1.121}$$

$$e^{-\beta E^{tot}_{x_i}} = e^{-\beta E^{tot}_{x_j}} \exp\left(-\frac{\Delta E_{ji}}{k_B T}\right) \qquad \text{for} \quad \Delta E_{ij} < 0, \text{ and} \tag{1.122}$$

$$e^{-\beta E^{tot}_{x_i}} = e^{-\beta E^{tot}_{x_j}} \qquad \text{for} \quad \Delta E_{ij} = 0. \tag{1.123}$$

where equality holds for all three cases.

This shows us that both rates are tending to the same equilibrium configuration (in mathematical words limiting or stationary distribution). As a consequence, measures only depending on the stationary distribution like charge carrier densities should give the same values for simulations done with Marcus rates and Miller-Abramahs rates respectively. This is very counter-intuitive as especially for high negative energy differences ΔE_{ij} the rates have nothing to do with each other. So how can we interpret this fact? Of course the Markov chains themselves are looking completely different and the time evolution of the systems as well. But the time evolution itself is not uniquely determining the limiting behaviour. And exactly this fact that different time evolutions can have the same limiting behaviour and furthermore the same stationary distribution, is responsible for the situation that for both rates measures only depending on the limiting distribution are the same. As an example, a simulation done with Marcus rates and another one done with Miller-Abrahams rates can have completely different autocorrelation functions but still the estimators of charge carrier densities have to be the same within their differing error bars.

This consequence also holds for considering different hopping regions. E.g. for a spherical hopping region, the hopping radius $r_{hop,cc}$ is the maximum distance for which a hop of a charge carrier is allowed. This means that rates $R(i,j)$ for cells separated by more than the hopping radius $r_{ij} > r_{hop,cc}$ are set to

$R(i,j) = 0$. It is a numerical parameter which reduces the computational cost of our simulation if it is chosen to be low. Depending on the charge delocalisation constant α, the hopping range has to be selected higher or lower to get a reliable time evolution. As long as the hopping region is taken in a way that the Markov chain is still irreducible and the allowed hops are symmetric, the detailed balance equation is fulfilled and the limiting behaviour does not change. Symmetric means if the rate for a charge carrier hopping from cell i to cell j is non-zero, then the rate from j to i has to be non-zero as well. Irreducibility is assured as long as the hopping region is big enough so that at least nearest neighbour hopping is allowed. So we see that the choice of the hopping region does not affect measures that only depend on the stationary distribution.

Measures depending on the time evolution of the Markov chain like velocities, mobilities and current densities are changing when we are changing the rates or the hopping region. So we have to be careful when we say that two different simulations are sampling from the same stationary distribution. Getting the same results for measures only depending on the stationary distribution does not imply getting the same results for all measures. But still it is remarkable at this point to state that charge carrier densities do not depend on the chosen rate or the hopping region while velocities, mobilities and current densities do.

This leads us to the next question: What is the difference between Marcus rates and Miller-Abrahams rates? Charge transport, in general, is governed by the interplay between electrons and phonons. Phonons are quasiparticles of vibrational modes. In an organic semiconductor intramolecular vibrations of single molecules (or segments of a polymer) and intermolecular vibrations of the whole material have to be considered. Including all electronic and vibrational degrees of freedom in a single calculation, one ends up with quasiparticles called polarons. Those polarons can be interpreted as electrons coated by a cloud of phonons which means that an electron is creating a distortion of the material as soon as it is placed anywhere. The energy associated with this distortion is called polaronic binding energy. Coming back to the electron-phonon picture, the charge transport is mainly influenced by two effects; the electronic coupling and the electron-phonon interactions. The electronic coupling in principle means that due to a wave function overlap of two localised states, the charge carrier can move through the material. In an organic semiconductor this overlap is rather small as the mainly Van der Waals-driven packing density of molecules is quite low compared to covalently bound inorganic semiconductors. The electron-phonon interactions result from the fact that a change in the position of an atom, caused by a phonon, changes the electronic structure of the whole molecule. As a consequence, phonons can raise and lower the energy levels of the electrons and induce hops. In organic semiconductors both is observed, strong and weak electron-phonon interactions depending on the molecule itself and the alignment of the molecules. For loosely packed molecules with intensively swinging intramolecular vibrations one could probably assume a strong electron-phonon interaction. Of course this simple picture does not hold in reality as many effects concurrently govern this interaction strength. To get an estimator for the electron-phonon interactions, molecular dynamics simulations or density functional theory are suitable. In summary we can say that for an organic semiconductor the electronic coupling is weak and the electron-phonon interaction can be weak or strong. With the assumption of weak electronic coupling a general expression for the rates can be obtained by means of time-dependent perturbation theory. To get the Miller-Abrahams rates from this expression, weak electron-phonon interactions and low temperatures are considered. Only one phonon can be absorbed or emitted during a hop and hence the energy difference that is overcome during a hop should not exceed the maximum energy of the acoustical phonons (the Debye energy) and the energy of the optical phonons. In contrast, the Marcus rate is received when assuming strong electron-phonon coupling and high temperatures.

1.2.3 The Simulation

Performing a Hop:
With the given rates (1.99) or (1.100) and the sample-path behaviour given in theorem 1.6 we can construct our Markov chain. Starting from a given state in which charge carriers occupy certain cells, we can

calculate the energy differences for hops of all charge carriers to all nearby cells within a distance of the hopping region. With those energy differences the corresponding rates can be calculated. Assuming that N_{cc} charge carriers are in the simulation and N_{hop} is the number of neighbouring cells within the hopping region, a number of $N_R = N_{cc} \cdot N_{hop}$ rates R_i have to be calculated. Additionally we need the total rate $R^{(N_R)} = \sum_{i=1}^{N_R} R_i$ and with this the next hop is chosen with probability $\frac{R_i}{R^{(N_R)}}$. This is done by taking a uniformly distributed random number ξ out of the interval $(0,1]$ and multiply it by $R^{(N_R)}$ to get a random number $\xi^R = \xi \cdot R^{(N_R)}$ out of the interval $(0, R^{(N_R)}]$. Now we are summing up the individual rates R_i and as soon as the condition $\xi^R \leq \sum_{i=1}^{j} R_i$ is fulfilled, j is our chosen hop. The time that the system stays in a certain state was said to be exponentially distributed with rate parameter $R^{(N_R)}$. So the retention time Δt is generated by taking a random number ξ uniformly distributed over the interval $(0,1)$ and calculate

$$\Delta t = -\frac{\log(\xi)}{R^{(N_R)}} \tag{1.124}$$

where $log(e) = 1$ is the natural logarithm. The fact that Δt is exponentially distributed with rate parameter $R^{(N_R)}$ can be easily shown by means of the inverse transformation method (see e.g. [24]).

Now the Markov time is evolved by the amount Δt and the hop of the charge carrier corresponding to rate j is performed. In this new state we recalculate the energy differences and the rates, choose a hop and assign a time and repeat this process again and again and again...

Initial State:
The initial state of our system can be chosen randomly as it has no effect on the simulation at all as long as it cannot be directly drawn out of the stationary distribution. If it could be drawn out of the stationary distribution, we would not have to thermalise the system. But as long as one cannot be sure whether the starting state is drawn from the stationary distribution or not, we have to thermalise the system according to the asymptotic autocorrelation time τ_{exp}. A commonly taken value for the amount of Markov time to spend with thermalisation is about 10% of the total Markov time. If thermalisation is not finished within 10% of the total Markov time, it can be assumed that most of the Markov chain is correlated and the measures taken from this chain are not representative anyway. On the other hand taking less than 10% does not really effect the computational effort. Coming back to the starting configuration, note that if we are able to draw a state directly out of the stationary distribution, we would not need to simulate it with a Markov Chain Monte Carlo method. So this consideration is needless anyway and the take home message of this passage is: Thermalisation is important!

Measurement:
After thermalisation we can start measuring. The most important measures are charge carrier densities, velocities, mobilities and current densities. Those quantities and the relations between them will be discussed in the following.

The most obvious measure is the charge carrier density $n_{cc}(V)$ where V is the volume with respect to which averaging is performed. At a certain time t, a number of $N_{cc}(V,t)$ charge carriers is in the volume V which leads to the time averaged estimator of $n_{cc}(V)$

$$n_{cc}(V) = \frac{1}{V\,(t_{N_M} - t_0)} \int_{t_0}^{t_{N_M}} N_{cc}(V,t)dt = \frac{1}{V\,(t_{N_M} - t_0)} \sum_{i=0}^{N_M-1} N_{cc}(V,t_i)\Delta t_i \tag{1.125}$$

where (as introduced in chapter 10) t_0 is the time after thermalisation, t_{N_M} is the final Markov time after N_M hops and $\Delta t_i = t_{i+1} - t_i$ is the retention time for hop i. In this master thesis, the charge carrier density is often given in charge carriers per cell (ccpc), rather than in charge carriers per volume, as this is more meaningful for the simulations:

$$n_{cc}^{cell}(V) = \frac{1}{\frac{V}{V_{cell}}\,(t_{N_M} - t_0)} \int_{t_0}^{t_{N_M}} N_{cc}(V,t)dt = \frac{V_{cell}}{V\,(t_{N_M} - t_0)} \sum_{i=0}^{N_M-1} N_{cc}(V,t_i)\Delta t_i. \tag{1.126}$$

The volume of a cell is written as V_{cell}.

The other measures mentioned above are dynamical properties which have to be calculated with more care. First we will calculate an estimator of the velocity and start with only one charge carrier in our simulation. It performs hops at times t_i and stays in a state for an amount of time Δt_i until the next hop is performed. We can expect that at any time in this time interval the charge carrier has to move. The knowledge of the exact time dependence of the velocity $\vec{v}(t)$ is not necessary, as we are only interested in the average with respect to time. So whether it is a Dirac delta function describing an instantaneous hop, a constant velocity in the time interval Δt_i, or anything else does not effect the time average:

$$\vec{v}_i = \frac{1}{\Delta t_i}\int_{t_i}^{t_{i+1}} \vec{v}(t)dt = \frac{\Delta\vec{s}_i}{\Delta t_i}. \tag{1.127}$$

$\Delta\vec{s}_i$ is the distance that was overcome during this hop. Averaging this velocity over the time interval $[t_0, t_{N_M}]$ of the whole Markov chain after thermalisation, we get

$$\vec{v}^{\,av} = \frac{1}{t_{N_M}-t_0}\int_{t_0}^{t_{N_M}} \vec{v}(t)dt = \frac{1}{t_{N_M}-t_0}\sum_{i=0}^{N_M-1}\frac{\Delta\vec{s}_i}{\Delta t_i}\Delta t_i = \frac{\sum_{i=0}^{N_M-1}\Delta\vec{s}_i}{t_{N_M}-t_0} = \frac{\Delta\vec{s}^{\,tot}}{t_{N_M}-t_0} \tag{1.128}$$

So the time-averaged velocity of a single charge carrier is simply the total distance $\Delta\vec{s}^{\,tot}$ that it travelled over the total time $(t_{N_M}-t_0)$ elapsed. Taking into account that a constant number of N_{cc} charge carriers is present in the simulation, our time dependent velocity $\vec{v}^{\,j}(t)$ gets an additional index for the charge carrier j. Averaging over those charge carriers leads to

$$\begin{aligned}\vec{v}^{\,av} = \frac{1}{N_{cc}}\sum_{j=1}^{N_{cc}}\frac{1}{t_{N_M}-t_0}\int_{t_0}^{t_{N_M}}\vec{v}^{\,j}(t)dt = \frac{1}{N_{cc}\,(t_{N_M}-t_0)}\sum_{j=1}^{N_{cc}}\sum_{i=0}^{N_M-1}\frac{\Delta\vec{s}_i^{\,j}}{\Delta t_i}\Delta t_i = \frac{1}{N_{cc}\,(t_{N_M}-t_0)}\sum_{i=0}^{N_M-1}\Delta\vec{s}_i = \\ = \frac{1}{t_{N_M}-t_0}\sum_{i=0}^{N_M-1}\frac{\Delta\vec{s}_i}{N_{cc}\Delta t_i}\Delta t_i = \frac{1}{t_{N_M}-t_0}\sum_{i=0}^{N_M-1}\vec{v}_i\Delta t_i\end{aligned} \tag{1.129}$$

where we see that we can define an average velocity $\vec{v}_i = \frac{\Delta\vec{s}_i}{N_{cc}\Delta t_i}$ for each time step. The distance travelled by charge carrier j in time interval i is given by $\Delta\vec{s}_i^{\,j}$. Summing over all times for one charge carrier j gives the total distance $\Delta\vec{s}^{\,j,tot}$ that this charge carrier travelled during the simulation. Additionally summing over all charge carriers leads to the total distance travelled by all charge carriers $\Delta\vec{s}^{\,tot}$ and is the same as summing over all hopping distances $\Delta\vec{s}_i$ belonging to the time interval Δt_i not depending on which charge carrier performed this hop:

$$\sum_{i=0}^{N_M-1}\sum_{j=1}^{N_{cc}}\Delta\vec{s}_i^{\,j} = \sum_{j=1}^{N_{cc}}\left(\sum_{i=0}^{N_M-1}\Delta\vec{s}_i^{\,j}\right) = \sum_{j=1}^{N_{cc}}\Delta\vec{s}^{\,j,tot} = \Delta\vec{s}^{\,tot} = \sum_{i=0}^{N_M-1}\Delta\vec{s}_i. \tag{1.130}$$

As soon as the number of charge carriers is not constant, it is not straight forward to give an ensemble average for the velocity as it is not really clear how to weight different numbers of charge carriers $N_{cc,i}$ for different times t_i. If we do it straight forwardly by using the definition of the average velocity for a time step i introduced above

$$\vec{v}_i = \frac{\Delta\vec{s}_i}{N_{cc,i}\Delta t_i} \tag{1.131}$$

we end up with an effective number of charge carriers $N_{cc,eff}$ in the simulation given by the relation

$$\vec{v}^{\,av} = \frac{\Delta\vec{s}^{\,tot}}{N_{cc,eff}\,(t_{N_M}-t_0)} = \frac{1}{t_{N_M}-t_0}\sum_{i=0}^{N_M-1}\frac{\Delta\vec{s}_i}{N_{cc,i}} \tag{1.132}$$

where $\frac{1}{N_{cc,eff}}$ in general has to be a matrix to fulfil the relation component-wise. This matrix is difficult to interpret from a physical point of view. The time averaged velocity of a single charge carrier is still

physically interpretable. So probably by averaging reasonable over those time averaged velocities of all charge carriers, respecting the time that they spent in the simulation, could lead to a proper average velocity. Anyhow, we will not calculate velocities for systems in which the number of charge carriers can change, so we do not have to care about this right now.

The mobility μ is simply defined as the ratio between the velocity in field direction v_E and the absolute value of the electric field strength F

$$\mu = \frac{v_E}{F} \tag{1.133}$$

and can be calculated directly from the measured velocity discussed above (at least for a constant number of charge carriers).

Considering fluctuating numbers of charge carriers, the current density $\vec{j}$ is a more representative measure for the system than the velocity. Additionally it can be spatially resolved in an easy way as it is not necessarily referred to the trajectory of a charge carrier. There are two useful and, of course, theoretically equivalent ways to define the current density. The first one is to count the number of charge carriers ΔN_{cc} with charge $q = \pm e$ that go through a certain area A in a time interval $\Delta t = t_{N_M} - t_0$. With this the current density j_A averaged over this area is given by

$$j_A = \frac{q\Delta N_{cc}}{A\Delta t} \tag{1.134}$$

This definition is a very useful measure for the simulation as we just have to count the amount of charge carriers going through the desired area. For a spatial resolution of the current density $\vec{j}(\vec{r},t)$, we need to take a more fundamental definition

$$\vec{j}(\vec{r},t) = qn_{cc}(\vec{r},t)\,\vec{v}(\vec{r},t) \tag{1.135}$$

which is the product of the charge $q = \pm e$, the charge carrier density $n_{cc}(\vec{r},t)$ and the velocity $\vec{v}(\vec{r},t)$ at a given position and time. For a certain time interval i starting at time t_i and a certain cell k with position $\vec{r}_k$, we can find a charge carrier density

$$n_{cc}(\vec{r}_k,t_i) = \frac{N_{cc}(k,i)}{V_{cell}} \tag{1.136}$$

where $N_{cc}(k,i)$ is the number of charge carriers occupying cell k at time t_i which is 0 if it is not occupied and 1 if it is occupied. Next we find an expression for the velocity of a hop performed by a charge carrier sitting in cell k at time t_i

$$\vec{v}(\vec{r}_k,t_i) = \frac{\Delta\vec{s}(k,i)}{\Delta t_i} \tag{1.137}$$

where $\Delta\vec{s}(k,i) = \Delta\vec{s}_i$ if the hop at time t_i started at cell k and $\Delta\vec{s}(k,i) = 0$ if not. With this the current density gets

$$\vec{j}(\vec{r}_k,t_i) = \frac{q}{V_{cell}} N_{cc}(k,i)\frac{\Delta\vec{s}(k,i)}{\Delta t_i} \tag{1.138}$$

and we can average it over time. As the cell has to be occupied before a hop, $N_{cc}(k,i)$ and $\Delta\vec{s}(k,i)$ are zero exactly for the same times and, when $N_{cc}(k,i) = 1$, also $\Delta\vec{s}(k,i)$ is non-zero. So $N_{cc}(k,i)$ can be dropped and the time average of the current density $\vec{j}(\vec{r}_k)$ is given by

$$\begin{aligned}\vec{j}(\vec{r}_k) &= \frac{1}{t_{N_M}-t_0}\int_{t_0}^{t_{N_M}} \vec{j}(\vec{r}_k,t)\,dt = \frac{1}{t_{N_M}-t_0}\sum_{i=0}^{N_M-1}\vec{j}(\vec{r}_k,t_i)\,\Delta t_i = \\ &= \frac{1}{t_{N_M}-t_0}\sum_{i=0}^{N_M-1}\frac{q}{V_{cell}}N_{cc}(k,i)\frac{\Delta\vec{s}(k,i)}{\Delta t_i}\Delta t_i = \frac{q}{V_{cell}\,(t_{N_M}-t_0)}\sum_{i=0}^{N_M-1}\Delta\vec{s}(k,i) = \frac{q\Delta\vec{s}^{\,tot}(k)}{V_{cell}\,(t_{N_M}-t_0)}\end{aligned} \tag{1.139}$$

which means that the current density at cell k is calculated by summing up all distances of all hops performed from this cell $\Delta\vec{s}^{\,tot}(k) = \sum_{i=0}^{N_M-1} \Delta\vec{s}_i(k)$. When we are additionally averaging this current density over a certain volume $V = A \cdot l$ with an area A and a length l we get

$$\vec{j}_V = \frac{V_{cell}}{V} \sum_j^{r_j \in V} \frac{q\Delta\vec{s}^{\,tot}(k)}{V_{cell}\,(t_{N_M} - t_0)} = \frac{q}{A \cdot l\,(t_{N_M} - t_0)} \sum_j^{r_j \in V} \Delta\vec{s}^{\,tot}(k) = \frac{q}{A \cdot l\,(t_{N_M} - t_0)} \Delta\vec{s}^{\,tot}(V) \tag{1.140}$$

By assuming that the charge carriers are mainly travelling normal to the area A so that the number of charge carriers ΔN_{cc} which were going through area A can be estimated by $\Delta N_{cc} \approx \frac{|\Delta\vec{s}^{\,tot}(V)|}{l}$ we get

$$\left|\vec{j}_V\right| = \frac{q}{A\,(t_{N_M} - t_0)} \frac{|\Delta\vec{s}^{\,tot}(V)|}{l} \approx \frac{q\Delta N_{cc}}{A\,(t_{N_M} - t_0)} = j_A \tag{1.141}$$

Our definition of the current density $\vec{j}(\vec{r}_k, t_i)$ is most likely the easiest one, but probably not the best. By addressing all the travelled distance to the starting cell, this cell gets an overestimation of distance whereas all the other cells that are passed during the hop are getting an underestimated amount of distance, namely nothing. With this approach, a scenario in which a charge carrier is hopping between two cells for many times would correspond to a high current in both cells pointing in the opposite direction while effectively no current would pass the area between the two cells. So it might be better for some applications to distribute the travelled distance evenly amongst all the cells that were passed during the hop to get more smooth current densities. On the other hand this smoothing would not take those forth and back hopping into account. So in general both methods complement one another and it would be useful to implement both. We have only used the method in which all the distance is put into one cell.

1.2.4 The Update Mechanisms

With the term 'update mechanism' we refer to the way that the rates are recalculated after a hop. Above, one update mechanism was already introduced which we call 'Dynamic Monte Carlo' or DMC. In the DMC approach we recalculate all energy levels of all charge carriers and all energy levels that charge carriers would have if they were moving from their current cell i to an adjacent cell j within the hopping region. For the calculation of one energy level, we have to sum up the contributions of all Coulomb interactions from all other charge carriers $(N_{cc} - 1)$. As we have to do this for N_{cc} charge carriers and $N_{hop} + 1$ cells (N_{hop} is the number of neighbours within the hopping region), we have to handle the Coulomb interaction $(N_{cc} - 1)N_{cc}(N_{hop} + 1)$ times. Looking at the dependence on the number of charge carriers N_{cc} of the time that this calculation needs, we are calling it an $\mathcal{O}(N_{cc}^2)$-process. In this master thesis we are interested in constructing a method that can cope with huge amounts of charge carriers, so a DMC simulation is expected to be way to time-consuming.

An established alternative to this computationally expensive update mechanism is the so called 'First Reaction Method' or FRM. [24] We have already proven (see page 17 and following) that, for a given set of N_R rates R_i and its sum $R^{(N_R)} = \sum_{i=1}^{N_R} R_i$, the following two update mechanisms are stochastically identical:

1. Choose a process i according to the probability $\frac{R_i}{R^{(N_R)}}$ and assign a retention time Δt out of an exponential distribution with rate parameter $R^{(N_R)}$. This is exactly the update mechanism that we became familiar with above and we called DMC.

2. Assign a retention time Δt_i to each process i out of an exponential distribution with rate parameter R_i and choose the process with the fastest time.

At first sight it seems that the second option is way more intricate compared to the first one as, (i), we still have to calculate all the rates after each hop and, (ii), additionally we need much more random numbers. As generating random numbers is usually time-consuming and the risk of running into correlations of the pseudo-random-number-generator rises with the amount of required random numbers, this option seems

to be totally infeasible. But with the following property of exponentially distributed random numbers, new light is shone on this method.

A time t is drawn from an exponential distribution with rate R at time t_0. If it is known that there is a time t_1 for which $t_0 \leq t_1 < t$ holds, then the time t behaves as if it was drawn from an exponential distribution with rate R at time t_1.

To prove this statement, we will have a look at the probability distribution $p(t|t > t_1 \wedge t_1 \wedge (t_0,R))$ which holds the probability distribution of t given that $t > t_1$ for a fix t_1 and (t_i,R) stands for the fact that t was drawn out of an exponential distribution with rate R starting at time t_i. Such an exponential distribution can be written as

$$p(t|(t_i,R)) = \Theta(t-t_i)Re^{-R(t-t_i)} \tag{1.142}$$

with the Heaviside function $\Theta(t)$. As a non-relevant remark, this should be the left-continuous Heaviside step function $\Theta(0) = 0$ rather than the right-continuous one that we used elsewhere, as simultaneously happening events are restricted in a Poisson process. For the prove this does not make any difference as an integral does not change its value by changing only one discrete point.

With Bayes' theorem (1.76) and marginalisation (1.77) we can rewrite

$$p(t|t > t_1 \wedge t_1 \wedge (t_0,R)) = \frac{P(t > t_1|t \wedge t_1 \wedge (t_0,R))\, p(t|t_1 \wedge (t_0,R))}{\int_{-\infty}^{\infty} P(t > t_1|t \wedge t_1 \wedge (t_0,R))\, p(t|t_1 \wedge (t_0,R))\, dt} \tag{1.143}$$

As $p(t|t_1 \wedge (t_0,R))$ does not depend on the value of t_1 at all it simplifies to $p(t|(t_0,R))$ and the probability $P(t > t_1|t \wedge t_1 \wedge (t_0,R))$ that $t > t_1$ for given values of t and t_1 can only take on the values 0 or 1 depending on if the statement $t > t_1$ is true or false. With this we can solve the integral in the denominator

$$\begin{aligned}
\int_{-\infty}^{\infty} P(t > t_1|t \wedge t_1 \wedge (t_0,R))\, p(t|t_1 \wedge (t_0,R))\, dt &= \int_{-\infty}^{\infty} \Theta(t-t_1)\,\Theta(t-t_0)\, Re^{-R(t-t_0)} dt && (1.144)\\
&= \int_{t_1}^{\infty} Re^{-R(t-t_0)} dt && (1.145)\\
&= R\left(-\frac{1}{R}\right)\left(e^{-R(t-t_0)}\right)\Big|_{t_1}^{\infty} = e^{-R(t_1-t_0)} && (1.146)
\end{aligned}$$

where $t_0 \leq t_1$ assures that $\Theta(t-t_0)\,\Theta(t-t_1) = \Theta(t-t_1)$. Putting all those equations together leads to

$$\begin{aligned}
p(t|t > t_1 \wedge t_1 \wedge (t_0,R)) &= \frac{\Theta(t-t_1)\,\Theta(t-t_0)\, Re^{-R(t-t_0)}}{e^{-R(t_1-t_0)}} && (1.147)\\
&= \Theta(t-t_1)\, Re^{-R(t-t_1)} && (1.148)\\
p(t|t > t_1 \wedge t_1 \wedge (t_0,R)) &= p(t|(t_1,R)) && (1.149)
\end{aligned}$$

which finishes the prove. So we see that, given the time t_1 is lower than the time t, the random number $t' = t - t_1$ is behaving exactly as if it would have been drawn new from an exponential distribution with rate constant R.

How can we use this result for our alternative update mechanism? The FRM, in general, implies that we have a number of n processes labelled by i with rates R_i. At Markov time 0 we assign a retention time Δt_i^0 to each process i from an exponential distribution with rate constant R_i. The process j with the shortest retention time Δt_j^0 is performed and the Markov time is evolved to Δt_j^0. The rate for process j is recalculated and a new time Δt_j^1 is chosen from an exponential distribution with this new rate constant R_j. At this point, the fact shown above comes into play because, assumed that the other rates R_i do not change, we can use the times Δt_i^0 calculated previously to get the new times $\Delta t_i^1 = \Delta t_i^0 - \Delta t_j^0$. So we do not need to recalculate all rates and all retention times in each step. We only have to calculate the new rate of the currently performed process and get the retention time for it, i.e., only one random number is required for one Markov step.

Now this alternative update mechanism seems to be much more favourable than the original one, so why

is it not used for every simulation? The problem is, that the rates of the other processes are changing as soon as we include interactions in our simulation. This means that in FRM we are neglecting that the interactions change the rate due to which we induce a methodological error. For very low charge carrier densities, FRM is producing good results: All charge carriers are far apart of each other. Thus, the interactions are not changing in good approximation.

We are interested in developing a new update mechanism that combines the benefits of FRM and DMC, in which a reduced computational cost by only recalculating a few rates is combined with reducing the methodological error by updating the 'right' rates. As FRM is producing good results for low charge carrier densities and fails for high charge carrier densities, it is quite obvious to define, as a next step, an update radius r_{up}: For charge carriers j within this update radius around the hopping charge carrier i $|\vec{r}_i - \vec{r}_j| < r_{up}$, the rates are recalculated after a hop and for all the other charge carriers the retention times are reduced by the current retention time. The statement $p(t|t > t_1 \wedge t_1 \wedge (t_0, R)) = p(t|(t_1, R))$ proven above and the equality of the two sample-path behaviours proven on page 17 ensure that this update mechanism is correct (if the rates outside the update sphere are not changing). With this update mechanism, we can continuously change from FRM ($r_{up} \to 0$) to DMC ($r_{up} \to \infty$). This master thesis focuses on the question of how to choose the update radius to get a negligible methodological error while the computational effort is still low.

As this idea is unexceptional, it was no surprise that other researchers already had this idea. After implementing this method, we found out that Heiber *et al.* already claimed to have performed related simulations but without giving any details of the method. [25]

2 Development of the Code

One of the major goals of my master thesis was to write the code for a kinetic Monte Carlo simulation. This section contains all information needed to efficiently set up the program starting from scratch. As I have written the program in Fortran, special attention is drawn to the intricacies of implementing the program in this particular language. Nevertheless nearly all information in this section is also valid for all other programming languages.

In chapter 2.1 I present a suggestion, how to develop the code starting from non-interacting simulations via short range interacting bulk system to a fully interacting injection simulation.

In a next step, the reliability of the measurements of a kinetic Monte Carlo simulation is scrutinised in chapter 2.2. The values of not necessarily physical parameters such as simulation time, system size, interaction radius and many more can have huge effects on particular observables. Hence the choice of those parameters is critically reviewed.

All programmers know that coding always goes hand in hand with hours and hours of debugging. To make future programmers life a bit easier, problems that I was struggling with and hints for non-obvious program details are found in chapter 2.3.

2.1 Setting Up a Kinetic Monte Carlo Simulation

This chapter is not aiming to give a detailed recipe for the implementation of a KMC solver in the sense of stating that a certain variable has to be in this structure or the dependency diagram of the used routines has to look like this. Those informations are given for the implementation I developed in the appendix (see chapter 7.1). This chapter can be seen as a guideline for developing your own code. The implementation hints are always trying to guarantee the highest possible flexibility without loosing performance. Depending on the requirements of your KMC solver, some points are probably to general and you can save coding time by implementing it more adapted to your system and other points might be implemented not flexible enough and you have to think of implementing it more universal. The information given here should be seen as support and not as strict rule.

2.1.1 Single Charge Carrier Bulk Simulation

One of the easiest systems for a KMC simulation is a single charge carrier in a quasi-infinite volume. As there is nothing but the organic semiconducting material in the simulation (no metal contacts or other interfaces), it is called a bulk simulation. Although it is very simple, it already contains the core part of a KMC simulation, the hopping process. Important structures like the cells that can be occupied by charge carriers are also present. Hence, we will start here by building up the lattice of the cells.

Lattice:
A lattice of $N_{cells} = N_1 \times N_2 \times N_3$ cubic cells of a certain size l (the lattice constant) is created in which every cell is assigned a certain index to address it (see fig. 2.1). Next, a variable `neighbours(i1,i2)` is introduced, in which the neighbours of every cell `i1` into direction `i2` are stored. The directions are the allowed hopping directions, e.g. for nearest neighbour hopping only (which is mainly used in this master thesis), there are 6 nearest neighbours in our simple cubic lattice. In this neighbour network the periodic boundary conditions (pbc) can be included straight-forwardly, e.g. the nearest neighbours of cell 1 in fig. 2.1 would be 2, 4 and 10 without pbc and with pbc the cells 3, 7 and 19 would be nearest neighbours as well. The benefit of using such a neighbour network, compared to a three dimensional index like (2,1,3) for cell 20 in fig. 2.1, is, that any lattice can be simulated without the need to change much in the code. Moreover, in most programming languages (except Matlab) the addressing of a neighbouring cell via this variable `neighbours(i1,i2)` is faster than adding e.g. (1,0,0) to (2,1,3) to get from cell 20 to cell 21 which is its neighbour in `i2` = 1 direction. At this point I want to mention that for an efficient implementation of a KMC solver in Matlab, other aspects have to be considered and the method I am

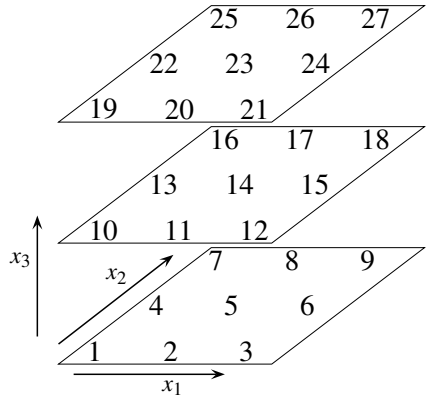

Figure 2.1: Example for the labelling of the cubic cells with a $3 \times 3 \times 3$ lattice. Counting starts in the first dimension and every time one dimension is filled up, one step into the next higher dimension is executed.

presenting here is probably not the best solution. The intricacies arising when Matlab is used for coding are discussed in the master thesis of Philipp Breitegger [26], who in parallel with me developed a KMC solver in this particular programming language.

Energy Assignment:
Having built up the lattice and a neighbour network, we can continue by randomly assigning energy levels $\varepsilon_{\mathtt{i1}}$ for the cells `i1` according to a Gaussian distribution (1.98) with a specific energetic disorder σ and writing them into a variable `energetic_landscape(i1)`. For a bulk simulation of a single charge carrier, the energy of the charge carrier reduces to the first two terms in (1.101). The external electric field $\vec{F}$ has to be applied into a certain direction, e.g. the x_3-direction. As periodicity has to be considered, the best way to include the energy contribution of the electric field is directly in the energy difference ΔE_{ij} between the charge carriers total energy occupying cell i and j respectively:

$$\Delta E_{ij} = \varepsilon_i - \varepsilon_j + qF\Delta x_3^{ij} \tag{2.1}$$

where $q = \pm e$ is the charge of the charge carriers and the distance of the hop between cell i and cell j into x_3 direction Δx_3^{ij} appears. The hop from cell i to cell j is belonging to a hopping direction `i2` in variable `neighbours(i1,i2)`. With knowing this hopping direction, the distance of the hop can be looked up from a predefined variable holding the distances for the allowed hopping directions. The distance between cells within the hopping region is a typical quantity that is needed many times in a Monte Carlo simulation. It is, thus, advisable to precalculate and store it once, rather than to calculate it some million times during the time evolution of the Markov chain. As this variable only has to hold $3 \cdot N_{hop}$ entries for a three dimensional system and N_{hop} allowed hopping directions, an insignificant amount of memory is needed. For Monte Carlo simulations in general, it is time-saving to precalculate quantities that are used in every step as long as they do not need too much storage.

Problem with Boltzmann Statistics:
With the energy difference ΔE_{ij} defined in (2.1) and the periodic boundary conditions, the charge carriers will moves, on average, into field direction (or in opposite direction of the field depending on the charge). However, in doing so they apparently loose potential energy continuously. The consequence for the total energy of the system becomes obvious, when a charge carrier exits the sample volume and is reintroduced at the opposite side due to the periodic boundary conditions. The charge carrier itself just hopped one lattice constant l into field direction and the total energy of the system was reduced by $\Delta E_{ij} = \varepsilon_i - \varepsilon_j - eFl$. But if we compare the two states of our state space, the total energy of the system is raised by $\Delta E_{ij} = \varepsilon_i - \varepsilon_j + eFl(N_{x_3} - 1)$, where N_{x_3} is the number of cells in x_3 direction. This means, that each state of our state space appears an infinite number of times with different total energies of the system due to the

periodic boundary conditions and the Boltzmann statistics (1.111) can no longer be used as stationary distribution.

The q-matrix of our Markov chain is still conservative, finite-dimensional, stable (and hence uniformly bounded) and irreducible and the system also tends to a stationary distribution $\pi_{x'} = \lim_{t\to\infty} \mathbf{T}_{x,x'}(t) \;\; \forall x \in \mathcal{X}$ as described in chapter 1.1.3. But the stationary distribution is not simply the Boltzmann statistics and it is not known by us at this point. It is also doubtful, if the stationary distribution can be found with the detailed balance criterion (1.40) or if the global balance criterion (1.39) has to be used. Nevertheless, it has to be related to the Boltzmann statistics, as locally the detailed balance criterion is fulfilled by the Boltzmann statistics. Unfortunately the insight gained in chapter 1.2.2, that the equilibrium quantities like spatially resolved charge carrier densities do not depend on the chosen rate equation or the hopping radius, cannot be assumed to hold for this unknown stationary distribution.

Hopping Process:
Having the energy differences for all the allowed hops of the charge carriers, we can calculate the rates, either Marcus (1.99) or Miller-Abrahams (1.100). With the rates, we can start the simulation by initially placing the charge carrier at a randomly chosen cell and calculate the rates for all N_{hop} hops. As all those rates become invalid after the hop, it is preferable to use the original sample-path behaviour described in theorem 1.6 and chapter 1.2.3. So we need one random number $\xi_1 \in (0,1]$ to choose the hop according to $\frac{R^i}{R^{(N_{hop})}}$ and a second $\xi_2 \in (0,1)$ to assign the retention time $\Delta t = -\frac{\log(\xi_2)}{R^{(N_{hop})}}$. After a hop, the rates are recalculated, the hop and the retention time are chosen randomly and the charge carrier evolves. After thermalisation we start to measure the displacement in field direction for each hop and sum it up. When the simulation has finished, we calculate the mobility μ

$$\mu = \frac{\Delta x_3^{tot}}{F \Delta t} \tag{2.2}$$

with the totally travelled distance, Δx_3^{tot}, into field direction, the electric field strength F, and the time, Δt, elapsed during the simulation after thermalisation. The totally travelled distance can be calculated either by, (i), remembering the starting position, x_3^{start}, and the final position x_3^{final} and calculating the total distance in x_3 direction $\Delta x_3^{tot} = x_3^{end} - x_3^{start}$ (considering the periodic boundary conditions) or, (ii), by summing up all hopping displacements in x_3 direction for each hop performed. The second possibility is slightly favourable, as, (i), the hopping distances are calculated anyway and, (ii), the convergence of the mobility can be monitored during the simulation. Additionally, the second case permits to easily estimate a Jackknife-based error (see page 14 and following) by leaving out blocks in the summation of the travelled distance. As long as the simulation time is much longer than the system inherent autocorrelation times, the errors calculated with Jackknife are reliable.

Error Determination:
Regarding errors, we have to consider that the choice of the energetic landscape already induces some degree of random ambiguity for all measurable quantities. Therefore, an averaging over multiple, randomly chosen, independent energetic landscapes is necessary to get reliable values for our measurable quantities. As the simulation results of different energetic landscapes can be assumed to be independent (as long as the random number generator is producing uncorrelated random numbers), the estimator of a quantity A_i averaged over N_{EL} energetic landscapes is

$$\langle A \rangle = \frac{1}{N_{EL}} \sum_{i=1}^{N_{EL}} A_i \tag{2.3}$$

and the estimator of the error ΔA of $\langle A \rangle$, given that the measurements are independent, is simply the standard error

$$\Delta A = \sqrt{\frac{1}{N_{EL}(N_{EL}-1)} \sum_{i=1}^{N_{EL}} (A_i - \langle A \rangle)^2} = \sqrt{\frac{\langle A^2 \rangle - \langle A \rangle^2}{N_{EL}-1}} \tag{2.4}$$

where

$$\langle A^2 \rangle = \frac{1}{N_{EL}} \sum_{i=1}^{N_{EL}} A_i^2 \tag{2.5}$$

When we are averaging over multiple energetic landscapes, it seems to be of limited use to calculate the error of the individual energetic landscapes. This is true as long as the error associated to the individual energetic landscapes is lower than the estimator of the error due to averaging over the energetic landscapes. For a thoroughly evaluated error, both the individual and the ensemble error have to be calculated and the larger of both has to be taken. In general, the ensemble error originating from averaging over multiple energetic landscapes should be the bigger one anyway. However, as the number of landscapes being simulated is limited, the ensemble error could, by chance, be lower as well.

Simulation Results:
With this a first runnable KMC simulation should be ready for testing. As a first benchmark, reference [16] is very suitable. In the first part of this paper, the bulk mobility for a single charge carrier is calculated for different electric field strengths. Hopping is restricted to nearest neighbour hopping and Marcus rates (1.99) are used to evolve the system. The parameters needed to reproduce the data given in fig. 1 in [16] are shown in tab. 2.1. The parameter E_r given in [16] is a bit misleading, as one has to take $\varepsilon \cdot E_r$ instead of just E_r to get the same results. The value for E_r given in tab. 2.1 is correct when our definition of the Marcus rate (1.99) is used.

Table 2.1: Parameters needed to reproduce the bulk mobility shown in fig.1 in [16]

E_r	0.75 eV ($12 \cdot 10^{-20}$ J)
ν_0	$6.76 \cdot 10^{11}$ s^{-1}
σ	62.4 meV ($1 \cdot 10^{-20}$ J)
T	298 K
$N_1 \times N_2 \times N_3$	$35 \times 35 \times 70$
α	0 m^{-1}
l	1 nm

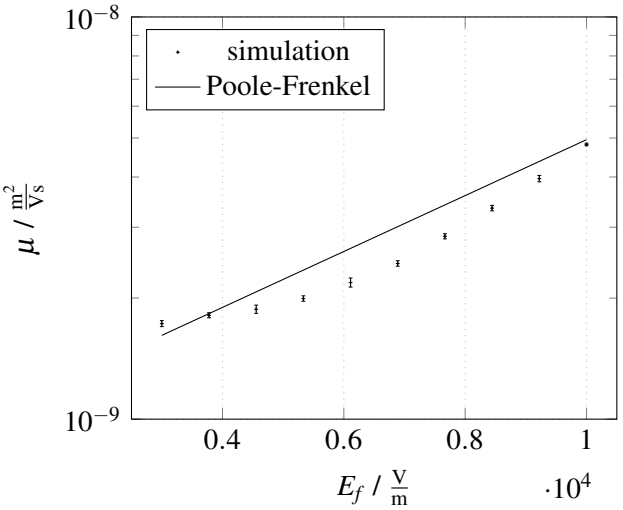

Figure 2.2: Simulation of the bulk mobility of a single charge carrier in a disordered material. The dots are our simulation results and the line shows the fitted Poole-Frenkel-type electric field dependence given in [16]

The comparison of our results with the fitted function for the Poole-Frenkel-type field dependence of the mobility performed in [16] is shown in fig. 2.2. For this simulation, the charge carrier was hopping

1,000,000 times plus 100,000 steps for thermalisation; for every electric field strength 10 randomly chosen energetic landscapes were sampled. The error is obtained by averaging the mobility values associated to those 10 independent energetic landscapes. The curves of our simulation and the corresponding ones in the paper show the same behaviour.

2.1.2 Single Charge Carrier Injection Simulation

To enhance the simulation from a single charge carrier bulk simulation to a single charge carrier injection simulation, we have to put a metal contact on one side and adapt the boundary condition on the opposite side. To get an idea of how to model the boundary condition on the opposite side of the metal contact, the work of Wolf *et al.* [27] was studied. In this work, a single charge carrier injection simulation was performed in a way discussed in the following.

A lattice of $170 \times 170 \times 20$ cubic cells with a lattice constant of $l = 0.6$ nm was simulated with periodic boundary conditions into direction x_1 and x_2. For the calculation of the energy levels (compare (1.101)), the Gaussian distributed energies of the cells E^{cells}, the energy caused by an electric field applied in x_3 direction E^{field} and the contribution of the own image charge $E^{\text{own image charge}}$ was considered. As expected for a single charge carrier simulation, no interactions are considered. Having a look at the energy needed to place a charge carrier at cell i (1.109), only three of the five terms are present in this simulation, the interaction terms do not appear since there are no other charge carriers to interact with. Miller-Abrahams rates (1.100) were used. We note here that the process was modelled in an unconventional way, probably this is due to the fact, that limited computer power in 1999 required particularly economic approaches. A charge carrier is starting in the metal at the Fermi energy E_F. For a charge carrier hopping out of the metal, the energy of the starting cell i (called injection cell) for the hop is simply the Fermi energy $E_i = E_F$ while the energy of the cell j of our simulation volume where the charge carrier hops to is $E_j = E_j^{\text{cell}} + E_j^{\text{field}} + E_j^{\text{own image charge}}$. The rates for the injection of a charge carrier from the metal to the organic semiconductor are Miller-Abrahams rates with $\Delta E_{ij} = E_j - E_F$. Injections are assumed to be only perpendicular to the metal surface and the charge carriers can only hop into the first two layers. To average over multiple energetic landscapes, the position of the metal contact is chosen randomly. This means that a neighbouring pair of layers is chosen out of the 20 possible layer pairs and exactly for those two layers the corresponding rates for injection are calculated. The distance from the contact to the first layer is the lattice constant and to the second layer twice the lattice constant. This distance of one lattice constant from the contact to the first layer is taken for the hopping distance to calculate the rates as well as for the starting point of the electric field and the image charge potential. After calculating all those $n = 170 \times 170 \times 2$ rates R_{ij}, one injection event is chosen randomly corresponding to the probability $\frac{R_{ij}}{\sum R_{ij}}$, which is exactly the same as introduced for our sample-path behaviour of the Markov chain (1.41). No time is assigned in this simulation. This is understandable, as no time dependent measures are taken. After this injection, the charge carrier can perform hops within a cube of $5 \times 5 \times 5$ cells centred around it. If the contact is within this cube, it is considered as one single cell with Fermi energy, i.e. also hops back into the metal are only permitted perpendicular to the metal surface. If the charge carrier hops back into the metal, it recombines with the contact and is not further regarded in the on-going simulation. If the charge carrier reaches the 9^{th} layer or hops even beyond it, the charge carrier is assumed to be separated far enough from the contact. In this case we call the charge carrier escaped and also take it out of the simulation. So a charge carrier is removed from the simulation either when it recombines with the metal contact or when it escapes. Simulating many charge carriers one after the other and counting the numbers of escape N_{esc} and recombination events N_{rec}, the escape probability of a charge carrier ϕ can be measured:

$$\phi = \frac{N_{esc}}{N_{esc} + N_{rec}} \tag{2.6}$$

Beyond these reasonable assumptions, the authors introduced a correction factor f to approximately account for hot charge carriers that are injected from the tail of the Fermi-Dirac distribution. This does not affect the energy of the injection cells for the simulation itself, which is still the Fermi energy E_F.

Rather, there is a competing process to injection where the charge carrier recombines within the metal due to a relaxation from the hot tail of the Fermi-Dirac distribution to the Fermi energy E_F. This latter process is assumed to have a rate of $R = \nu_0$ which corresponds to a rate with $r_{ij} = 0$ and $\Delta E_{ij} < 0$. The correction factor

$$f = \frac{\sum R_{ij}}{\nu_0 + \sum R_{ij}} \tag{2.7}$$

is easily accessible, as the sum over all injection rates $\sum R_{ij}$ is calculated in the course of the simulation. The relaxation in the metal is not directly included in the simulation. The associated correction factor is determined after the simulation and the corrected escape probability ϕ_f is defined as

$$\phi_f = \frac{N_{esc}}{N_{esc} + N_{rec}} f. \tag{2.8}$$

I.e. the denominator still contains the total number of charge carriers that were simulated, but the number of successfully escaped charge carriers N_{esc} is reduced by the factor f corresponding to the fraction of charge carriers that would not have made it out of the metal. Exactly this quantity ϕ_f is displayed in fig.1 in [27].

How can we interpret this paper to develop an injection simulation and use the results given in this paper to benchmark our simulations? As we are interested in developing a code to measure current densities for interacting charge carriers that can be continuously injected, it is evidently not helpful to implement the method exactly in the same way as in [27]. Our simulations differs in two technical aspects from the system simulated by Wolf *et al.*: (i), the sampling over multiple energetic landscapes is done differently, (ii), our injection and hopping processes assign a time.

The creation of the energetic landscape is not a bottleneck of the simulation at all, so it seems to be useless to average over 20 correlated energetic landscapes when we can easily create 20 uncorrelated ones with not much extra computer power required. Each of those uncorrelated energetic landscapes is created as follows: As an escape happens in the layers more than 8 cells apart from the contact and the charge carriers can overcome a maximum distance of two layers per hop, our bulk simulation size is $170 \times 170 \times 10$. For the metal contact, an additional layer of cells is placed at the top in x_3 direction. This means that layer 1 and 2 are the escape layers and layer 11 is the metal contact.

Each injection cell can perform injections and let charge carriers recombine with it at any time as the huge reservoir of charge carriers in the metal is assumed to be unaffected by changing the number of charge carriers in the reservoir by 1. So the injection cells are treated as being occupied and unoccupied at the same time, which is consistent with the delocalisation and abundance of the charge carriers in the metal. The energy of the injection cells is the Fermi energy E_F for injection and recombination. One could also think of injecting hot charge carriers from the occupied tail $E > E_F$ of the Fermi-Dirac distribution and let them recombine with the contact according to the unoccupied part $E < E_F$ of the Fermi-Dirac distribution. For sure, this changes the physical results of the measurements (Note that when considering the Fermi-Dirac distribution for injection rather than a sharp Fermi energy, care has to be taken to avoid violation of detailed balance). As we are mainly interested in comparing different simulation techniques in this master thesis, we will, for the sake of simplicity, assume a sharp Fermi energy E_F for injection and recombination throughout the remainder of this thesis.

Based on the such defined processes and energy levels, we can calculate rates with the Miller-Abrahams rate equation, (1.100). For injection, all $170 \times 170 \times 2$ injection rates can be precalculated and an update mechanism has to be chosen. Without interaction, all injection rates are constant and DMC and FRM are equivalent. For DMC, an injection is chosen according to (1.41) and an exponentially distributed time with a rate parameter being the sum over all injection rates is drawn. After the injection took place, the single charge carrier evolves in the same way as described in the chapter 2.1.1 on single charge carrier bulk simulations until it either recombines or escapes. Now the next injection is performed in the same way as before and the charge carrier moves again through the bulk, and so on. In FRM, at the beginning all injections get a time drawn from an exponential distribution with a rate parameter according to the

individual rates and the fastest injection event is performed. This fastest time is, (i), subtracted from all other times and, (ii), recalculated. After the charge carrier escaped or recombined, the next fastest injection is performed. For an efficient simulation it is useful to sort the hopping times in FRM. In both cases the number of escapes and recombinations is counted and the escape probabilities ϕ (2.6) and ϕ_f (2.8) are calculated.

Simulation Results:
Now we can test our injection simulation and the required parameters are summarised in tab. 2.2. For the given values of the system size $N_{cells} = N_1 \times N_2 \times N_3$, the extension of the contact layer is not included. Note that the hopping prefactor ν_0 is not listed. As no times are measured, ν_0 does not effect the simulation and can thus be chosen arbitrary (e.g. $\nu_0 = 1\ \mathrm{s}^{-1}$).

Table 2.2: Parameters needed to reproduce the escape probabilities shown in fig.1 in [27]

σ	80 meV
T	250 K
$N_1 \times N_2 \times N_3$	$170 \times 170 \times 10$
l	0.6 nm
α	$\frac{5}{l} = 8.33 \cdot 10^9\ \mathrm{m}^{-1}$
ε_r	3.5

The results of our simulations are shown in fig. 2.3. Our escape probabilities ϕ without the correction factor f, corresponding to (2.6), are shown as dashed lines, and the escape probabilities ϕ_f with correction factor f, given by (2.8), are plotted as continuous lines. The results depicted in [27] are displayed as crosses ($\times$). Each simulation was run until 10000 injected charge carriers either recombined with the contact or escaped. For all simulations this took less than 2000000 steps (no thermalisation required in this particular case). Additionally 20 different energetic landscapes were sampled. The values for the escape probabilities were produced by averaging over the data from the 20 uncorrelated runs with different energetic landscapes. The term zero-field-energy barrier Δ in this context refers to the difference between the Fermi energy E_F and the mean value of the Gaussian distributed energy levels of the cells ε_i without an external electric field and own image charge interaction considered. So as $\langle \varepsilon_i \rangle = 0$ we get $\Delta = -E_F$.

The results fit quite well with the data from [27]. Especially for high zero-field-energy barriers Δ (lower panels in fig. 2.3) it can be seen, that the shape of the curve is not just a consequence of the escape probability ϕ, but mainly formed due to the correction factor f. The lowest zero-field-energy barrier $\Delta = 0.2$ eV (top left panel in fig. 2.3) is an outlier, our simulation does not represent the data shown in [27] at all. Only the slope for low electric fields seems to be quite similar.

We are quite convinced that out simulations are correct, also for the $\Delta = 0.2$ eV simulation. To support our results, the following discussion is held. As the correction factor f for the $\Delta = 0.2$ eV case does not change the general behaviour of the simulation, it is omitted in the further discussion. For high electric fields, all the charge carriers are forced to the escape layers and the injection efficiency will approach unity. In contrast, for low electric fields the charge carriers will be forced back into the contact by their image charge and the injection efficiency goes to 0. This means that there has to be a transition between the electric-field-dominated regime ($\phi \to 1$) and the image-charge-dominated regime ($\phi \to 0$). To find the approximate electric field where this transition takes place, we have to take a look at the energetic landscape. For field strengths above $5 \cdot 10^7\ \mathrm{Vm}^{-1}$, the energy levels beyond the second layer are decreasing with increasing distance to the contact. As our transition field strengths are above this value, the challenge for a charge carrier wanting to escape is to reach the second layer. Due to the spatial decay term, injection of charge carriers preferably occurs into the first layer rather than into the second. In a simplified picture, they can decide there if they want to move to the second layer or back into the metal. The escape process can be assumed to be favoured, if the average energy level of the

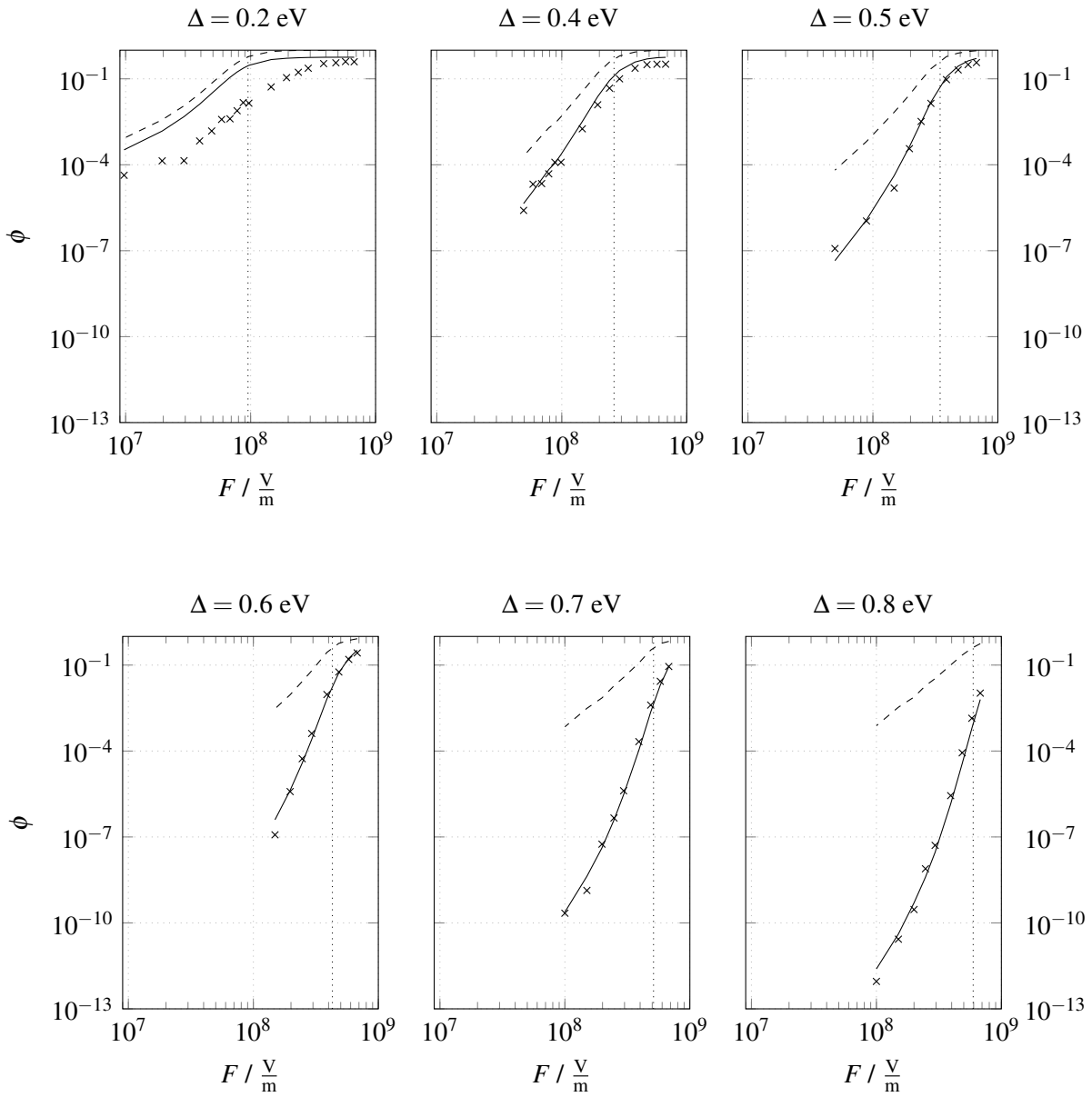

Figure 2.3: Simulated escape probability ϕ depending on the electric field strength F for different zero-field-energy barriers Δ. The dashed line shows our simulated escape probability ϕ without the correction factor f considered (calculated with (2.6)) and the solid line shows ϕ_f taking f into account (calculated with (2.8)). The results from [27] are shown as crosses (x). The estimated transition field (see tab. 2.3) are indicated with a dotted line. The results of our simulation, including the correction factor f, fit well with the data given in [27] except the $\Delta = 0.2$ eV curve (see discussion in the text).

second layer is lower than the Fermi energy. To get an estimate for the average energy level in the second layer, the energies due to the electric field, the own image charge and the energetic disorder have to be considered. To estimate the contribution from the energetic disorder, some thoughts have to be made. The injection will be preferably done to a low energy cell in the first layer, which means that the charge carrier is occupying a certain cell in the first layer and cannot move around very much. If the charge carrier wants to hop to the second layer, hops are restricted to the neighbouring cells of this low energy cell in the first layer and the charge carrier cannot choose low energy cells in the second layer. As a consequence the energy contribution of the energetic disorder is, on average, $\varepsilon_i = 0$. So the transition can be approximately calculated by equalising the Fermi energy and the energy contribution from the

electric field and the image charge in the second layer

$$E_F = 2lqF - \frac{e^2}{16\pi\varepsilon_0\varepsilon_r 2l} \tag{2.9}$$

where l is the lattice constant. Values for this transition are given in tab. 2.3 for different barriers Δ. For the $\Delta = 0.5$ eV and 0.4 eV cases, where such a transition is particularly apparent in the simulations, the estimated fields fit very well. If we have a look at the $\Delta = 0.2$ eV case, the transition field fits much better to our simulation than to [27].

Table 2.3: Electric field strength (2.9) where the transition from a field dependent to an image charge dependent regime approximately takes place as a function of the zero-field-energy barrier Δ.

Δ / eV	F / Vm^{-1}
0.2	$9.53{\cdot}10^7$
0.4	$2.62{\cdot}10^8$
0.5	$3.45{\cdot}10^8$
0.6	$4.29{\cdot}10^8$
0.7	$5.12{\cdot}10^8$
0.8	$5.95{\cdot}10^8$

This seems to favour our simulation results, but why should [27] be wrong only for the $\Delta = 0.2$ eV case? This is the only case where the first layer overall builds a trap layer, i.e., the layer average energy level is below the Fermi energy and for low electric field strengths also below the layer average of the second layer. This means that a charge carrier is injected into the first layer and travels around in the first layer for a very long time until it decides whether to recombine or to start the journey to the escape layers. The sometimes enormously long time that a charge carrier needs to pick a path is computationally very expensive and as in the year 1999 computer power was scarce, the simulations in [27] maybe did not converge. All the other simulations for the other Fermi energies were by far less intense from a computational point of view.

2.1.3 Interacting Bulk Simulation

The next step in the development of the program is to implement interactions between the charge carriers. For the energies calculated with (1.109) the first three parts are present for our interacting bulk simulation and the external electric field is favourably included in the calculation of the energy difference as described in (2.1). For the hopping process, in general, Marcus or Miller-Abrahams rates can be taken. We have decided to take Miller-Abrahams rates (1.100). To assure that only one charge carrier can be housed by one cell, we have used an array `occupied` that tells us whether a cell is currently occupied or not. As soon as interaction and, hence, multiple charge carriers in the simulation are concerned, all three update mechanisms can be implemented.

For a DMC simulation it is done exactly in the same way as it is done for one charge carrier; we calculate all rates R_i for all charge carriers N_{cc} and all allowed hopping cells of those charge carriers N_{hop}, one hop is chosen according to (1.41) and the time is advanced by an exponentially distributed random number with rate parameter being the sum over all rates $\sum R_i$.

In FRM, we combine both alternatives introduced in chapter 1.2.4. For each individual charge carrier we calculate all rates R_i to all cells i within the hopping region. We get N_{hop} rates and calculate the sum of them $R^{(N_{hop})}$. According to the probability $\frac{R_i}{R^{(N_{hop})}}$ we choose one hop and assign a retention time Δt drawn from an exponential distribution with rate $R^{(N_{hop})}$. This is done for every charge carrier j and so we end up with N_{cc} times Δt_j and hopping destinations i_j where N_{cc} is the number of charge carriers. The hops are sorted in a hopping queue according to their time and the charge carrier k with the fastest time Δt_k is allowed to hop to cell i_k. All times Δt_j of all other charge carriers $j \neq k$ are reduced by Δt_k

and the Markov time is advanced by Δt_k. The hopping time of the currently hopped charge carrier k is recalculated as described above and sorted back into the hopping queue.

For our new method it works exactly in the same way as for FRM with the only difference that after a hop not only the hopping time for the currently hopped charge carrier is recalculated, but the hopping times of all charge carrier within a distance of the update radius $r_{up,cc}$ to the new cell where the charge carrier hopped to. Searching for the charge carriers to update can either be done by looking through all charge carriers in the simulation or by scanning the region that has to be updated. For the second alternative, the array `occupied`, introduced above, which tells us if a cell is occupied or not is very useful, especially if it is not just a logical array but one that holds the index of the charge carrier that is occupying the according cell. Depending on the number of charge carriers that are in the simulation and the update radius, one of those two searching methods is favourable. If the number of charge carriers is higher than the number of cells within the update sphere the second method is chosen. If it is the other way round the first one is preferred. In both cases the periodic boundary conditions have to be taken into account to get the correct distances between the cells.

Evaluation of the Coulomb Interactions:
A very crucial topic that is raised at this point is the calculation of the Coulomb interaction. As the $\frac{1}{r}$ dependence of the interaction strength is known to be very long ranging and we are considering periodic boundary conditions, we, in principle, would have to use the Ewald summation method (see chapter 2.2.3) to calculate the interactions. A commonly made approximation is to cut-off the potential at a radius given by the so-called thermal capture radius r_{tc} used e.g. in [16]

$$r_{tc} = \frac{e^2}{4\pi\varepsilon_0\varepsilon_r k_B T} \tag{2.10}$$

From a mathematical point of view, cutting off a long ranging $\frac{1}{r}$ potential leads to a methodological error that can be huge. However, from a physical point of view, the cut-off can be interpreted. A cut-off radius may occur due to the screening of the Coulomb potential in the organic semiconductor. This lends the justification to use a cut-off radius, at least in the very first implementation. To avoid high virtual forces at the border of the cut-off region, the Coulomb potential for the cut-off radius should be subtracted from the Coulomb interaction. For the interaction of two charge carriers at cell i and j respectively, the corrected cut-off Coulomb potential $E_{ij}^{\text{cut-off Coulomb}}$ is given by

$$E_{ij}^{\text{cut-off Coulomb}} = \frac{e^2}{4\pi\varepsilon_0\varepsilon_r r_{ij}} - \frac{e^2}{4\pi\varepsilon_0\varepsilon_r r_c} \tag{2.11}$$

with the distance r_{ij} between the two cells and the cut-off radius r_c which can be chosen such that it is larger than the thermal capture radius $r_c > r_{tc}$. Note that, with this cut-off radius, our new update mechanism is exact as soon as the update radius is greater than the cut-off radius plus twice the hopping radius $r_{up,cc} > r_c + 2r_{hop,cc}$.

Including Coulomb interactions in the code can be done in two ways: (i), Either the sum in the third term of (1.109) is directly calculated according to the distances of the charge carriers for each energy level that needs to be calculated during an update. Or, (ii), an array is used where this sum is stored and corrected for each hop, i.e., the Coulomb potential of the hopped charge carrier centred at the previously occupied cell is subtracted and the Coulomb potential centred at the current cell, where the charge carrier hopped to, is added to this array. Note that for using the latter, a dipole potential for each hopping direction could reduce the computational effort but may overload the memory. The first option (i) is preferable for low amounts of charge carriers, a small amount of allowed hopping directions, a small update radius, and a large system size whereas the second one (ii) is used for high charge carrier densities, a large number of allowed hopping directions and a large update radius.

Simulation Results:
To benchmark the simulation of interacting charge carriers, the work done by Zhou *et al.* [28] can be

used. In this work bulk mobilities were measured depending on the charge carrier density considering Coulomb interactions. They utilised Miller-Abrahams rates for nearest neighbour hopping only and calculated the interactions with a direct summation method for the difference in Coulomb interaction energy between two cells. All energies, like the energetic disorder or the potential due to the external electric field, were given in k_BT. Unfortunately the Coulomb interaction energy was not given in this energy scale, as they used $\varepsilon_r = 4$. This choice of the Coulomb interaction energy fixes a total energy scale and vanishes the benefit of reduced quantities. Changing the temperature now changes the strength of the Coulomb interactions for the simulated system and without knowing the temperature, we are not able to reproduce the exact numbers of their simulations. Considering those missing or inconsistently given reduced values, we focused on the trends shown in [28] rather than on reproducing the exact numbers. For this we used a set of parameters that is comparable to that given in [28] (see tab. 2.4). The hopping prefactor was chosen to give a mobility of $\mu = 10^{-8}$ Vm^{-1} for the isoenergetic ($\sigma = 0$ meV) case.

Table 2.4: Used parameters to get the trend of the bulk mobility shown in fig.1 of [28]

ν_0	$2.25 \cdot 10^{13}$ s^{-1}
ε_r	4.0
F	$2.59 \cdot 10^{7}$ Vm^{-1}
T	300 K
$N_1 \times N_2 \times N_3$	$51 \times 51 \times 51$
l	1.0 nm
α	$\frac{5}{l} = 5 \cdot 10^{9}$ m^{-1}
r_c	20.0 nm

As this computationally expensive simulation should just serve as a benchmark, we were not spending too much time with sampling over multiple energetic landscapes to get error bars. Rather, the convergence check was just done by looking at the time series of the cumulatively measured mobility. About 10 to 20 million hops (+10% thermalisation) were needed to get acceptable convergence. Our measured mobilities can be seen in fig. 2.4, in which the results for FRM, DMC and the update radius method are shown. DMC simulations we carried out up to charge carrier densities of $n_{cc}^{cell} = 10^{-3}$ ccpc (charge carriers per cell), as for going beyond these densities the computational effort would not be proportional to the gained knowledge. An in-depth comparison between those three methods will be given in chapter 3.

In general, the trends for all simulations in fig. 2.4 behave in the same way as shown in [28]. From the viewpoint of a benchmark, our approach is not entirely wrong. Furthermore, we can already get a first impression of the performance of our new method. First we will take a look at the discrepancies of the three methods in the bottom left corner. The lowest lying curves refer to the highest disorder of 87.4 meV. At very low charge carrier densities (10^{-5} to 10^{-4} ccpc) the mobilities for all three update mechanisms differ. This discrepancy is coming from the fact that we were not averaging over multiple energetic landscapes. For each method and each energetic disorder, one energetic landscape was drawn according to the disorder. This landscape was used for all charge carrier densities to get comparable results for one curve. Assuming a high energetic disorder and only few charge carriers in the simulation, the results are dominated by the randomly chosen, very deep traps of our landscape. Hence, the dependence of the mobility on the energetic landscape is strongest for high energetic disorder and low charge carrier densities. It is remarkable that for higher charge carrier densities, the dependence on the deep traps is reduced and all three methods lead to the same mobilities for charge carrier densities between 10^{-4} and 10^{-3} ccpc for the highest energetic disorder.

Apart from the special case of high disorder and very low charge carrier densities, the results only differ for very high charge carrier densities. Here the fact that we are incorrectly considering the interactions in FRM (left panel in fig. 2.4) and also in the case of a small update radius (right panel in fig. 2.4) comes into play. The reasons that we hold responsible for this behaviour will be discussed in more detail in chapter 3.1. Here we just mention that FRM shows a peculiar down bending for high charge carrier

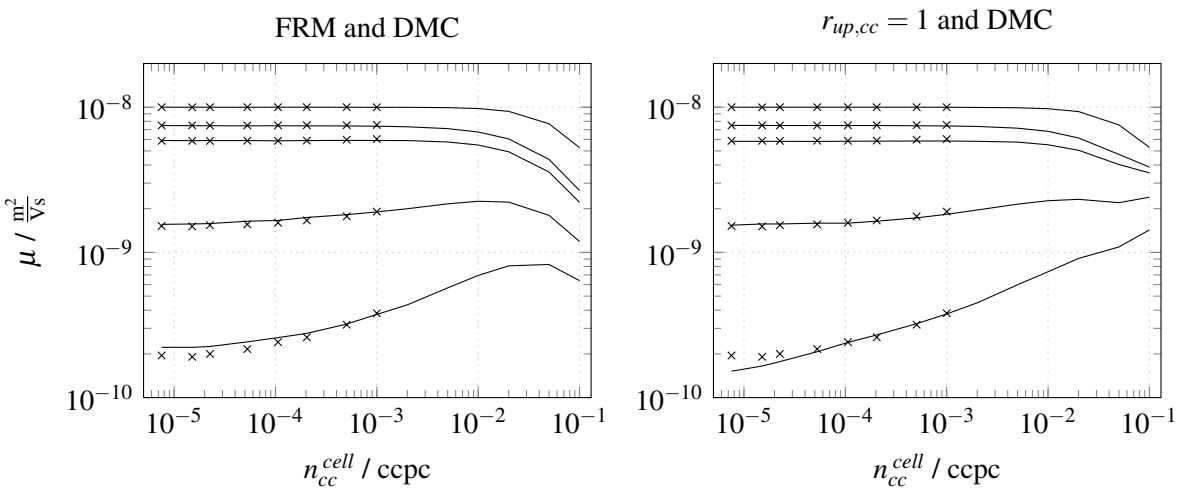

Figure 2.4: Bulk mobility μ depending on the overall charge carrier density n_{cc}^{cell} given in charge carrier per cell (ccpc) for different Gaussian disorders σ including Coulomb interaction between the charge carriers up to a cut-off radius of $r_c = 20$ nm. The energetic disorders σ are $(0.0, 43.7, 49.9, 68.7, 87.4)$ meV from top to bottom respectively. In the left plot, FRM results are shown with a solid line and DMC results with x. In the right plot, results for our new update method for an update radius of $r_{up,cc} = 1$ nm are drawn as solid line and the x are again the same DMC results as in the left plot. The trends recover those seen in [28].

densities being seemingly independent of the energetic disorder. On the other hand, the new method with a very low update radius $r_{up,cc} = 1$ nm already fits much better to the curves given in [28]. It is conjecturable that this low update radius does not lead to very accurate results compared to correct DMC calculations, but at least it is an improvement compared to FRM.

2.1.4 Interacting Contact Simulation

The obvious next step is to add all up to one program which performs injections and considers interactions. With all the work done previously this is rather simple as one just has to take a look at if the fastest injection time or the fastest hopping time is the lowest and perform this fastest process. The evaluation of injection times and hopping times will always be done separately to have a modular program that can be easily changed from injection to bulk simulations.

Update Mechanisms:
The calculation of the injection rates in DMC is done like for single charge carrier injection simulations as described in chapter 2.1.2.

For our new update mechanism, in which the update of hopping rates is confined to a region around the previously hopped charge carrier, we can distinguish between an update radius $r_{up,cc}$ for the charge carriers in bulk and $r_{up,inj}$ for injections. In the case of a bulk hop or injection, each of the two update spheres is centred at the cell in which the charge carrier has landed. However, for escape and recombination events, the charge carrier is removed from the simulation. Hence the two update spheres will be centred at the position prior the hop. The recalculation of rates is performed for all charge carriers within $r_{up,cc}$ and for all injection cells within $r_{up,inj}$. It is advisable to select hops from an injection cell reminiscent to bulk hops. I.e., the rates to all bulk cells that are allowed to be reached from the injection cell are determined. These include hops being perpendicular and diagonal to the contact surface. The associated probabilities are assigned according to (1.41). One of those injection events is randomly chosen corresponding to its probability. The duration of this event is associated to an exponential distribution in time (1.124), whose rate parameter corresponds to the sum of all hopping rates associated to this injection

cell. So, like charge carriers in bulk, each injection cell assigns an injection event to a certain bulk cell and a corresponding injection time. The injection times of all injection cells are sorted and the fastest injection time competes against the fastest hopping time.

Including injection in FRM simulations is less straight forward. The following problem readily illustrates, why an FRM implementation must differ from our update method described above. After a completed injection event, a charge carrier populates a cell into which the injection cell can still inject. This creates a scenario FRM is not meant to consider, i.e., to deal with (i) a close proximity of two charge carriers (the previously placed charge carrier and the one for which the next injection rate will be calculated) and (ii) an injection event blocked due to a charge carrier residing next to the interface. Rather, FRM is justified when low charge carrier densities are present. In this case, a previously injected charge carrier likely moved away from the contact prior the next injection from the same injection cell. This situation is much better accounted for, when the injection rates are reevaluated prior to the injection event. So in FRM we are, like in a bulk simulation, updating only the hopping rate of the charge carrier that hopped. For an injection process, however, we first recalculate the injection time of the injection cell, then place the charge carrier and finally calculate the hopping rate of the new charge carrier. For recombinations and escapes, no times have to be recalculated at all.

Hopping Region:
A spherical region with bulk hopping radius $r_{hop,cc}$ is assumed; this shape is consistent with the spherical-symmetrical spatial decay term in the rate expressions (1.99) and (1.100). Hops to all injection cells within the hopping radius are allowed, i.e. recombination is considered perpendicular and diagonal to the metal surface. Also for injection a spherical injection region is assumed, here with an injection radius $r_{hop,inj}$. However, as we are always starting from the Fermi level, rates for injections from different injection cells to the same bulk cell can be lumped into one rate. This single rate possesses a spatial decay factor consisting of the sum over all spatial factors $\sum_{i\in\text{injection region}} e^{-2\alpha r_{ij}}$ due to the individual injection cell i in the injection region. In essence, this can be interpreted as rate associated to an injection hop perpendicular to the metal surface with a modified spatial contribution.

Assessment of Detailed Balance:
It can be shown (for details see below) that the implementation of the simulation introduced above, i.e., in terms of injection, recombination, escape, hopping radii, etc., violates the conditions of detailed balance. Our simulation does not sample the Boltzmann distribution. For the purpose of comparing update mechanisms, being the focus of the present master thesis, sampling from an approximate Boltzmann distribution will not affect the main conclusions. However, the physical interpretation of injection following this simulation scheme is affected if not even inhibited.

To overcome this issue, we first turn towards the origin of the problem. The state space of an injection simulation only considers the locations of charge carriers that are in the organic semiconductors, i.e. the escape layers and the metal contact are not considered. A configuration within this state space is determined by the positions of the charge carriers. States associated to different numbers of charge carriers are linked via injection, recombination and escape. For injection and recombination, this linking is unproblematic if all cells allowed to inject are permitted to contribute to recombination. Strictly speaking, if the bulk hopping radius $r_{hop,cc}$ is identical with the injection radius $r_{hop,inj}$. Even though this is not always done in the following, it can be enforced and, thus, ensures detailed balance around the contact. However, the problems are more severe at the opposite side of the sample volume. An arbitrary state x with a charge carrier placed at a cell i near the escape layers is linked to a state x' in which this charge carrier has been removed via an escape process. If we call the rate for this escape process R, the q-matrix entry for the corresponding state is $q_{x,x'} = R$ (see chapter 1.1.3 for details about states, the q-matrix, detailed balance,...). However, there is no direct process that could link x' and x and no associated rate, i.e., $q_{x',x} = 0$. Now it is obvious that the detailed balance equation $\pi_x q_{x,x'} = \pi_{x'} q_{x',x}$ is not fulfilled considering that the stationary distribution π is the Boltzmann distribution. In fact, the situation is even worse, as the implementation of escape also leads to the fact that global balance cannot be fulfilled for the

Boltzmann distribution. This is because all processes except the escape process are in detailed balance (when $r_{hop,cc} = r_{hop,inj}$), so there is no counterpart that could compensate the escape process. To sidestep this problem, an inverse process to the escape process could be considered to guarantee detailed balance. This is done for diode simulations [29] due to the second electrode.

2.2 Convergence Considerations

In this chapter we are having a look at selected parameters of our kinetic Monte Carlo simulations and why we have chosen them in the way we did. For the remaining parameters, values commonly used in literature were taken. The lattice constant, as an example, strongly influences the simulations. We have taken a value of 1 nm for our simulations as this seemed to be a value commonly used in literature. When simulating a certain material, this parameter could be adapted and also the lattice structure could be chosen other than simple cubic, but we have not investigated the effects of a varying lattice constant or lattice structure at all. The impact of the hopping prefactor ν_0 was already discussed previously and we saw that is can be easily adapted after the simulation. So the value of this factor will not affect our comparison of different update methods. A clearly non-physical parameter is the hopping radius $r_{hop,cc}$ and $r_{hop,inj}$. The strength of the effect of changing the hopping radius is governed by the physically motivated parameter α (delocalisation constant, see (1.99) and (1.100)), the lower α the stronger the effect. If α is taken high, then changing $r_{hop,cc}$ or $r_{hop,inj}$ has only little effect. A rigorous investigation of the effect of the update radius would have gone far beyond the scope of this master thesis, so we restricted the simulations to nearest neighbour hopping. Some more parameters of the simulation will be discussed in the following sections.

2.2.1 Autocorrelation Times

A very important question for all Markov Chain Monte Carlo methods is: How long should a simulation run? As we already learned in chapter 1.1.4, the autocorrelation function can give us the answer to this question. To obtain the interplay between autocorrelation times and measured quantities, we computed the discrete empirical autocorrelation function (1.54). Note that this expression may not contain all the information provided in the accurate continuous empirical autocorrelation function (1.48).

The investigated system was a $51 \times 51 \times 51$ cells bulk simulation with periodic boundary conditions in all directions with only one charge carrier. A single charge carrier simulation was taken, because it is a very fast simulation and the autocorrelation times for this system should, in principle, be rather high compared to simulations with more charge carriers, as the effects of deep traps are stronger when less charge carriers are in the simulation. The hopping radius was restricting hops to nearest neighbour cells only and Miller-Abrahams rate (1.100) was taken. The rest of the parameters for this simulation are found in tab. 2.5.

Table 2.5: Used parameters for measuring the autocorrelation function.

ν_0	$2.25 \cdot 10^{13}$ s^{-1}
ε_r	4.0
F	$1.0 \cdot 10^{8}$ Vm^{-1}
T	300 K
σ	87.4 meV
$N_1 \times N_2 \times N_3$	$51 \times 51 \times 51$
l	1.0 nm
α	$\frac{5}{l} = 5 \cdot 10^{9}$ m^{-1}

For one run, 200 million hops plus 20 million hops thermalisation were performed and multiple energetic landscapes were investigated. The absolute value of the empirical autocorrelation function of the retention times Δt of the hops $\left|\rho^E_{\Delta t}\right|$, the displacements Δs during one hop in field direction $\left|\rho^E_{\Delta s}\right|$, and the

average mobility μ of one hop $\left|\rho_{\mu}^{E}\right|$ (see (1.131) and (1.133)) were measured. Results of some energetic landscapes are shown in fig. 2.5 for $\left|\rho_{\Delta t}^{E}\right|$, fig. 2.6 for $\left|\rho_{\Delta s}^{E}\right|$ and fig. 2.7 for $\left|\rho_{\mu}^{E}\right|$.

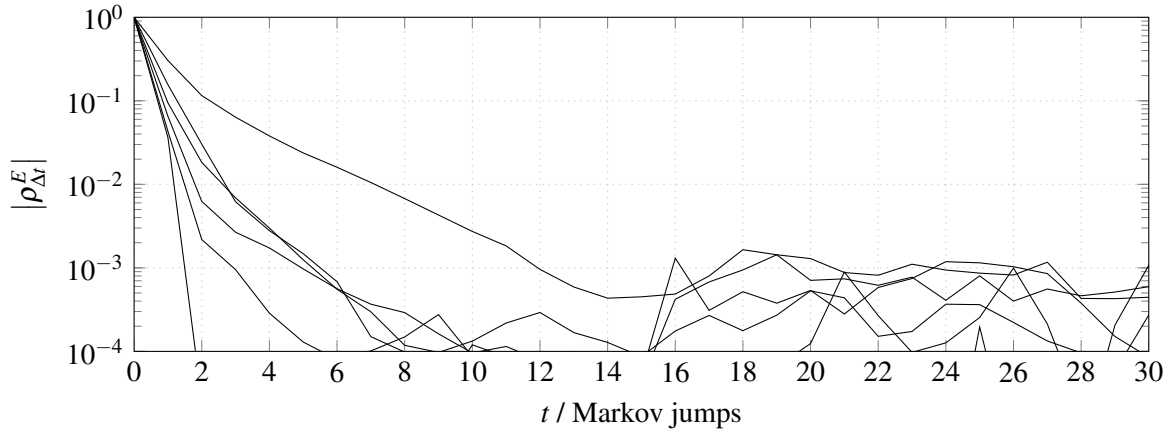

Figure 2.5: Absolute value of the empirical autocorrelation function $\left|\rho_{\Delta t}^{E}\right|$ of the hopping times Δt calculated with (1.54) for a single charge carrier bulk simulation with energetic disorder $\sigma = 87.4$ meV. The different curves show 6 different randomly chosen energetic landscapes.

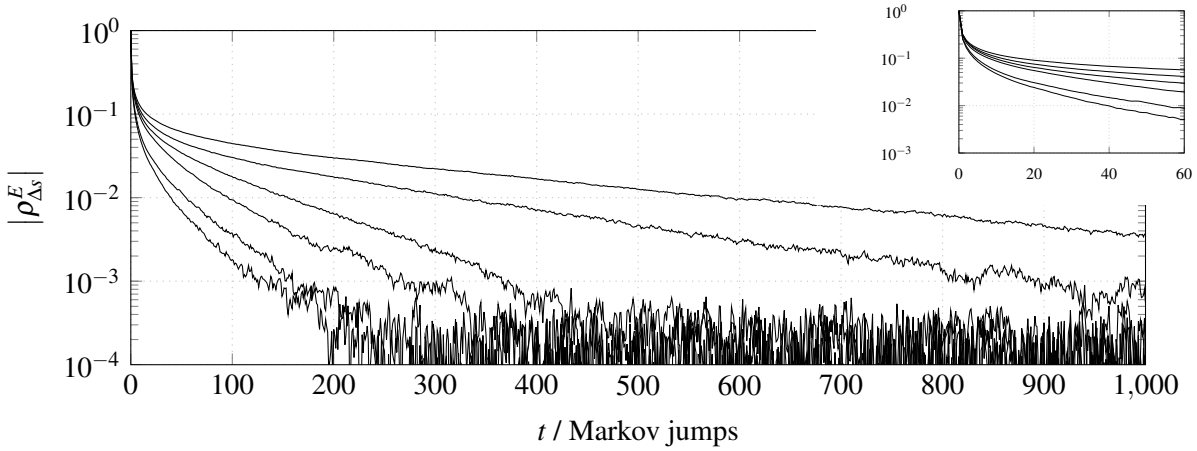

Figure 2.6: Absolute value of the empirical autocorrelation function $\left|\rho_{\Delta s}^{E}\right|$ of the hopping distances in field direction Δs calculated with (1.54) for a single charge carrier bulk simulation with energetic disorder $\sigma = 87.4$ meV. The different curves show 6 different randomly chosen energetic landscapes. The insert is a zoom for low times up to 60 Markov jumps.

When looking at the autocorrelation functions, the first eye-catcher is that the different energetic landscapes lead to completely different autocorrelation functions (the six curves in fig. 2.5 to 2.7 belong to six different energetic landscapes). Moreover, the autocorrelation function for the retention times Δt (see fig. 2.5) drops below noise much faster than the other two. The autocorrelation function for the hopping distances in field direction Δs (see fig. 2.6) and the mobility μ (see fig. 2.7) are quite similar apart from the slight difference that the straight line appearing for times $t > 100$ are shifted to lower values of the autocorrelation function for the mobility. Those facts can be interpreted consistently when we recall that the autocorrelation function for an observable A is given by

$$\rho_A(t) = \sum_{i=1}^{N_{ac}} c_i(A) e^{-\frac{t}{\tau_i}} \tag{2.12}$$

with N_{ac} autocorrelation times τ_i only depending on the transition function of the Markov chain and

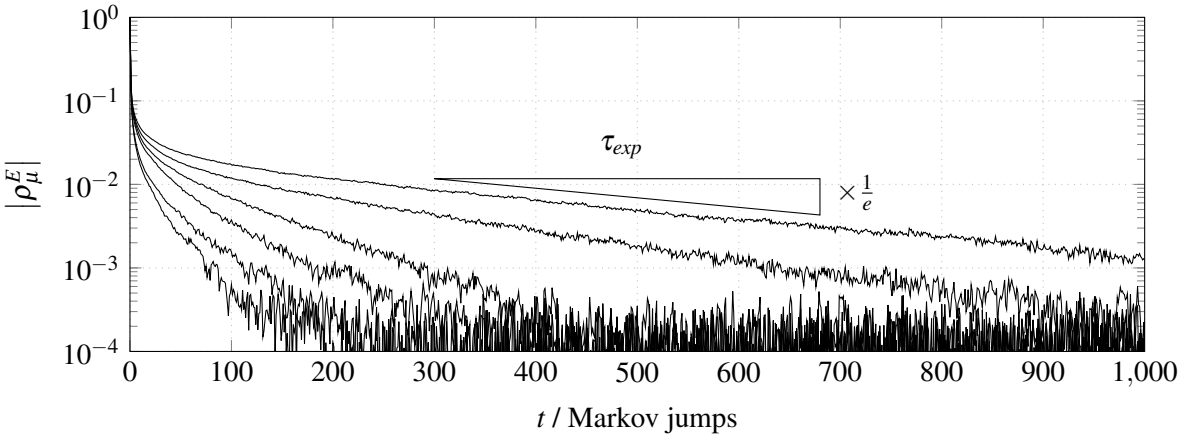

Figure 2.7: Absolute value of the empirical autocorrelation function $\left|\rho_\mu^E\right|$ of the mobility μ calculated with (1.54) for a single charge carrier bulk simulation with energetic disorder $\sigma = 87.4$ meV. The different curves show 6 different randomly chosen energetic landscapes. From the asymptotic behaviour of the autocorrelation function, the asymptotic autocorrelation time τ_{exp}, where $\left|\rho_\mu^E\right|$ drops by a factor of $\frac{1}{e}$, can be evaluated (see triangle above the curves).

coefficients, $c_i(A)$, of those exponentials only depending on the observable A. The autocorrelation times are determined by the transition function which, in turn, is given by the rates that are associated to the energy scales present in the simulation. This means that all three observables Δt, Δs, and μ should have the same autocorrelation times but with different coefficients. An example for evaluating an autocorrelation time from the autocorrelation function is shown in fig. 2.7 for the topmost curve. There the slope of the straight line for high Markov times $t > 200$ in the logarithmic plot provides the asymptotic autocorrelation time τ_{exp}. Looking at the autocorrelation function for the retention times Δt (fig. 2.5), we see very short autocorrelation times (all below 4 Markov jumps) before the six curves for the different energetic landscapes drop below noise. As they drop very fast, it cannot be said clearly if this is one specific autocorrelation time or many different but quite similar ones. In the insert in fig. 2.6, it is seen that there are many correlation times forming a continuously curved graph that finally ends in a straight line. This tells us that there is an interplay of many energy scales as it can be expected for a randomly disordered material. The straight line for long times seen in the autocorrelation function of Δs (fig. 2.6) and μ (fig. 2.7) is associated to the deepest trap in the system and, hence, can change a lot from one energetic landscape to another. The kink after which the behaviour of the autocorrelation function is dominated by the asymptotic autocorrelation time τ_{exp} is always approximately at the same value of the autocorrelation function (≈ 0.07 for Δs and ≈ 0.03 for μ) for different energetic landscapes. Those observations all reflect the fact that the autocorrelation times τ_i are governed by the energetic landscape and the coefficients $c_i(A)$ are determined by the observable. In the autocorrelation function for the retention times the asymptotic autocorrelation time should appear as well, but obviously the coefficient for this is below 10^{-3}.

After all this information about autocorrelation functions, what is the take home message of this section? For single charge carrier bulk simulation with a disorder of $\sigma = 87.4$ meV, we observed asymptotic autocorrelation times τ_{exp} in the range of 30 to 380 Markov jumps that strongly depend on the choice of the energetic landscape. The integrated autocorrelation time is much lower due to the multitude of autocorrelation times and a rapid drop of the autocorrelation function below at least 10^{-1} depending on the observable that was investigated. For our further simulations, we will not investigate autocorrelation functions and rather, calculate errors with Jackknife (see the passage about Jackknife on page 14). To check if thermalisation worked and to assure that no correlation times in the order of the simulation time appeared, the time series of the measured observable is inspected by eye. As long as thermalisation was successful, the estimators of errors calculated by Jackknife should give trustworthy results.

2.2.2 System Size

The system size is the number of cells $N_{cells} = N_1 \times N_2 \times N_3$ that we are simulating. For injection simulations the injection cells are not within this system size but the escape layers are. So N_{cells} is the number of cells of organic semiconducting material. As the simulation results depend on the choice of those values, we will give thought to those values in the following.

For bulk simulations with periodic boundary conditions, the choice of N_{cells} does not matter that much. Taking a smaller system mainly increases the variance between different energetic landscapes. Of course, there are further finite size effects that could be eliminated with finite size scaling, but this is not influencing our aim to compare different update mechanisms. Hence, we do not bother about finite size effects. In literature (e.g. [28]), typical values of about $50 \times 50 \times 50$ cells are thought to be 'large enough' to get measures that are showing the trend of the thermodynamic limit.

When injection is considered, the choice of the value for the system size parallel to the contact $N_1 \times N_2$ is related to the choice for bulk simulations as periodic boundary conditions are used here as well. So a value of about 50×50 injection cells is reasonable for our purposes. However, the number of layers perpendicular to the contact N_3 that are needed until the effect of the contact has decayed and the charge carriers behave like in a bulk simulation, is more challenging to estimate. Although we are giving an estimation in the following, the convergence to the bulk properties should be always checked as well. The convergence is seen best in the layer average of the charge carrier density parallel to the contact, which should reach a plateau far away from the contact. Of course there can be layers that have much higher charge carrier densities than others due to deep trap energy levels in those layers, but the trend of the layer averaged charge carrier density should not change much for an increasing distance to the contact.

Our estimation assumes, that the influence of the contact is negligible as soon as the probability that a charge carrier returns to the contact is negligible. To return to the contact, an energy difference ΔE has to be overcome by the charge carrier which is either determined by the Fermi energy E_F of the contact or by the maximum of the transport level in the organic semiconductor depending on which energy is the higher. To be able to easily calculate an estimation, we consider that one single charge carrier is in the simulation. This is reasonable, because other charge carriers would rather push our charge carrier away than help him to get back to the contact. For one charge carrier, the potential is formed by the external electric field F (1.103) and the image charge potential (1.105) for an isoenergetic case ($\sigma = 0$ meV). The maximum of the transport level is, in this case, found at a distance d to the contact that is given by

$$d = \sqrt{\frac{e}{16\pi\varepsilon_0\varepsilon_r F}}. \tag{2.13}$$

As we only have discrete distances, the maximum is either in layer $j = \left\lfloor \frac{d}{l} \right\rfloor$ or in layer $j+1 = \left\lceil \frac{d}{l} \right\rceil$[1] with the lattice constant l. We can calculate the maximum of the transport level with $E_{t,max} = \max\left(E(j), E(j+1)\right)$ where $E(j)$ is the image charge potential plus the potential from the electric field F for layer j. With non-vanishing energetic disorder $\sigma \neq 0$ meV, the effective maximum of the transport level is assumed to be reduced by σ which leads to $E_{t,max} = \max\left(E(j), E(j+1)\right) - \sigma$.

Regardless of which energy is the highest, the energy $\Delta E_{tot}(i)$ that has to be overcome by the charge carrier positioned in layer i is

$$\Delta E_{tot}(i) = \max\left(E_F, E_{t,max}\right) - \left(-\frac{e^2}{16\pi\varepsilon_0\varepsilon_r l \cdot i} - eFl \cdot i - \sigma\right) \tag{2.14}$$

To overcome this energy barrier, m layers have to be crossed, where m is the number of layers betweeen the current layer i of the charge carrier and, (i), the metal contact if the Fermi energy is the maximum, or, (ii), the layer containing the maximum of the transport level if $E_{t,max} > E_F$. As hopping through a disordered energetic landscape only decreases the probability that a charge carrier finds back to the

[1] $\lfloor x \rfloor$ and $\lceil x \rceil$ is floor and ceiling function of x

contact, an isoenergetic landscape is assumed in the following calculations, so that the energy difference between two layers is simply given by $\Delta E = \frac{\Delta E_{tot}(i)}{m}$. Note that this energy difference is depending on the layer i that is investigated. Furthermore we only consider nearest neighbour hops as long range hops will not significantly influence the results while increasing the complexity of the calculations tremendously.

So now we have reduced the problem to an isoenergetic case with energy difference ΔE between two layers and we want to know the probability that a charge carrier goes m steps upwards in energy at any time $P_{\Delta E}(m)$. The rates R_+ and R_- for hops upwards and downwards in energy and the rate R_0 for hops perpendicular to the contact can be calculated with Marcus rate (1.99) or Miller-Abrahams rate (1.100). The corresponding probabilities are

$$\tilde{P}_+ = \frac{R_+}{R_+ + R_- + 4R_0} \quad \tilde{P}_- = \frac{R_-}{R_+ + R_- + 4R_0} \quad \text{and} \quad \tilde{P}_0 = \frac{4R_0}{R_+ + R_- + 4R_0} \tag{2.15}$$

with four nearest neighbours perpendicular to the contact. Now we have further reduced the problem to that of a one dimensional random walk with probabilities $\tilde{P}_+$,$\tilde{P}_-$ and $\tilde{P}_0$ to go left (towards the contact or upwards in energy), right (away from the contact or downwards in energy) or stay at the current position. All four hops to the nearest neighbours perpendicular to the contact can be merged into one rate, $4R_0$, as they are not changing the distance to the contact and we do not care about at which lateral position the charge carrier overcomes the barrier or reaches the contact. The probability that the walker goes left m times $P_{\Delta E}(m)$ can be given iteratively. In the first step the walker can go left, right or stay. From this new position, the walker has to go left $(m-1)$, $(m+1)$ or m times, respectively, to totally move m steps to the left. So the iteration writes as follows:

$$P_{\Delta E}(m) = \tilde{P}_0 P_{\Delta E}(m) + \tilde{P}_- P_{\Delta E}(m+1) + \tilde{P}_+ P_{\Delta E}(m-1) \tag{2.16}$$

By moving the first term on the right to the left and dividing by $(1-\tilde{P}_0)$ we get

$$P_{\Delta E}(m) = P_- P_{\Delta E}(m+1) + P_+ P_{\Delta E}(m-1) \tag{2.17}$$

with

$$P_- = \frac{R_-}{R_+ + R_-} \quad \text{and} \quad P_+ = \frac{R_+}{R_+ + R_-} \tag{2.18}$$

which is a one dimensional random walk with probabilities P_+ and P_- to move to the left and right. The probability that the walker ever moves m steps to the left can also be given in a different way: The probability $P_{\Delta E}(m)$ that a walker at any time moves m steps to the left is given by the combined probability that it at any time moves $(m-1)$ steps to the left, $P_{\Delta E}(m-1)$, multiplied with the probability that, from there, it at any time moves one step to the left $P_{\Delta E}(1)$.

$$P_{\Delta E}(m) = P_{\Delta E}(m-1) P_{\Delta E}(1) \tag{2.19}$$

This relation is fulfilled by an exponential expression $P_{\Delta E}(m) = P_{\Delta E}(1)^m$. Plugging this into the iterative equation (2.17) leads to a quadratic equation for $P_{\Delta E}(1)$

$$\begin{aligned} P_{\Delta E}(m) &= P_- P_{\Delta E}(m+1) + P_+ P_{\Delta E}(m-1) && (2.20)\\ P_{\Delta E}(1)^m &= P_- P_{\Delta E}(1)^{m+1} + P_+ P_{\Delta E}(1)^{m-1} && (2.21)\\ 0 &= P_- P_{\Delta E}(1)^2 - P_{\Delta E}(1) + P_+ && (2.22) \end{aligned}$$

with the solution

$$\begin{aligned} P_{\Delta E}(1) = \frac{1 \pm \sqrt{1-4P_-P_+}}{2P_-} = \frac{1 \pm \sqrt{1-4P_-(1-P_-)}}{2P_-} = \frac{1 \pm \sqrt{(1-2P_-)^2}}{2P_-} = \\ = \frac{1 \pm (1-2P_-)}{2P_-} = \frac{1+(1-2P_-)}{2P_-} = \frac{1-P_-}{P_-} = \frac{P_+}{P_-} \end{aligned} \tag{2.23}$$

The relation $P_+ = (1 - P_-)$ was used and the positive sign of the root is taken for $P_- \geq \frac{1}{2}$ to get the lower of both probabilities. This is satisfied as P_- is the probability for hopping downwards in energy. With this we get our final result

$$P_{\Delta E}(m) = \left(\frac{P_+}{P_-}\right)^m = \left(\frac{R_+}{R_-}\right)^m \tag{2.24}$$

To calculate the number of layers for the simulation, a limit $P_{\text{max ret}}$ for the return probability $P_{\Delta E}(m)$ and a minimum number of layers i_{start} has to be given. For this minimum number of layers i_{start}, the energy difference $\Delta E_{tot}(i_{start})$ is calculated with (2.14) and the layer of the energy maximum j is evaluated. The distance between the last layer in simulation and the energy maximum $m = i - j$ is used to get the energy difference between two layers $\Delta E = \frac{\Delta E_{tot}(i_{start})}{m}$, with which the rates R_+ and R_- are evaluated, and the return probability $P_{\Delta E}(m)$ is calculated. If the return probability is already below our chosen limit $P_{\text{max ret}}$, i_{start} is the number of layers that is used for our simulation. Else the number of layers is increased and the procedure is repeated until the probability $P_{\Delta E}(m)$ undercuts $P_{\text{max ret}}$. In conclusion, we end up with a number of layers i that are needed for our simulation to achieve the desired accuracy. Finally, the number of escape layers has to be added to the calculated number of layers to get the total number of layers for our system size.

2.2.3 Interaction Range

The Coulomb potential of a charge carrier is very long ranging. So it is obvious that, with the cut-off radius introduced for the calculation of interactions, a methodological error is introduced that can be very significant for certain measures. For some measures, the effect of the cut-off radius might be negligible, but for others it is clearly not. Measures that are very sensitive to changes in the cut-off radius are, e.g., spatially resolved charge carrier densities in injection simulations (see [26] page 52). In the implementation we were introducing above, the interaction potential can be easily precalculated. That means that it is no significant difference in the performance of a simulation of the Markov chain whether the potential is calculated, (i), with only one term or, (ii), by summing up multiple terms and taking long range interactions into account. To get the exact potential of a system with periodic boundary conditions, the so-called Ewald summation method [30] is the method of choice. In this method the total energy of an interacting system with periodic boundary conditions can be calculated under the assumption that the system is charge neutral. With the considerations presented below, it can also be used to calculate the Coulomb potential of a single charge carrier in a 3D or 2D periodic system. The starting point for our calculations is the Ewald summation method described in [31]. There we find the calculation of the potential field generated by one single charge carrier at position $\vec{r}_i$ with charge q_i and all its periodic images

$$\varphi_i(\vec{r}) = \frac{q_i}{\varepsilon_0 \varepsilon_r} \frac{1}{4\pi} \sum_{\vec{T}} \frac{1}{\left|\vec{r} - \vec{r}_i - \vec{T}\right|} \tag{2.25}$$

where $\vec{T}$ is a translation vector of the system. In the case of periodicity in three dimensions and a system size of $N_{cells} = N_1 \times N_2 \times N_3$ cubic cells with lattice constant l, the translation vector can take values $\vec{T} = (i \cdot N_1 l, j \cdot N_2 l, k \cdot N_3 l)^\top$ with $i, j, k \in \mathbb{Z}$. The sum in (2.25) collects all those possibilities. Note that this sum is diverging, i.e., by subtracting another diverging sum we could produce any number that we want, depending on the summation sequence. This, of course, needs to be avoided. Thus we rewrite the sum with the so called Ewald method and isolate the singularity of the summation. The key idea of the Ewald summation method is to split the sum into a fast converging real space sum and a fast converging reciprocal space sum. In the following, the prefactor $\frac{q_i}{\varepsilon_0 \varepsilon_r}$ is set to 1 for notational convenience and will be, in the end, reintroduced.

3D-Ewald Summation

In a three dimensional system with periodicity in all three directions, the Fourier transform of the potential is given by

$$\tilde{\varphi}_i(\vec{G}) = \int_V d^3r \frac{1}{4\pi} \sum_{\vec{T}} \frac{1}{\left|\vec{r}-\vec{r}_i-\vec{T}\right|} e^{-i\vec{G}\vec{r}} e^{-\varepsilon\left|\vec{r}-\vec{r}_i-\vec{T}\right|} \tag{2.26}$$

where V is the volume of the system and the factor $e^{-\varepsilon\left|\vec{r}-\vec{r}_i-\vec{T}\right|}$ for $\varepsilon > 0$ assures the convergence of the integral and is afterwards taken out by letting ε go to zero. The reciprocal lattice vector $\vec{G}$ for a cubic system of $N \times N \times N$ cubic cells with lattice constant l is given by $\vec{G} = \frac{2\pi}{Nl}(i,j,k)^\top$ with $i,j,k \in \mathbb{Z}$. The sum over $\vec{T}$ is equivalent to integrating over $\mathbb{R}^3$ and with some integral transformations we get

$$\begin{aligned}
\tilde{\varphi}_i(\vec{G}) &= \int_{\mathbb{R}^3} d^3r \frac{1}{4\pi} \frac{1}{|\vec{r}-\vec{r}_i|} e^{-i\vec{G}\vec{r}} e^{-\varepsilon|\vec{r}-\vec{r}_i|} = \int_{\mathbb{R}^3} d^3r \frac{1}{4\pi} \frac{1}{r} e^{-i\vec{G}(\vec{r}+\vec{r}_i)} e^{-\varepsilon r} && (2.27)\\
&= \frac{1}{2} e^{-i\vec{G}\vec{r}_i} \int_0^\infty dr r^2 \int_{-1}^1 d\cos\Theta \frac{1}{r} e^{-iGr\cos\Theta - \varepsilon r} && (2.28)\\
&= \frac{1}{2} e^{-i\vec{G}\vec{r}_i} \int_0^\infty dr \frac{1}{-iG} \left(e^{-iGr} - e^{iGr}\right) e^{-\varepsilon r} && (2.29)\\
&= \frac{1}{2} e^{-i\vec{G}\vec{r}_i} \frac{1}{-iG} \left(\frac{1}{-iG-\varepsilon} e^{-iGr-\varepsilon r} - \frac{1}{iG-\varepsilon} e^{iGr-\varepsilon r} \right)\Bigg|_0^\infty && (2.30)\\
&= \frac{1}{2} e^{-i\vec{G}\vec{r}_i} \frac{1}{-iG} \left(\frac{1}{iG+\varepsilon} + \frac{1}{iG-\varepsilon} \right) = \frac{1}{G^2} e^{-i\vec{G}\vec{r}_i} && (2.31)
\end{aligned}$$

To reconstruct the real space potential, an inverse Fourier transformation has to be performed:

$$\varphi_i(\vec{r}) = \frac{1}{V} \sum_{\vec{G}} \tilde{\varphi}_i(\vec{G}) e^{i\vec{G}\vec{r}} = \frac{1}{V} \sum_{\vec{G}} \frac{1}{G^2} e^{-i\vec{G}\vec{r}_i} e^{i\vec{G}\vec{r}}. \tag{2.32}$$

Although we are now summing up a function that behaves like $\frac{1}{G^2}$ for large G instead of $\frac{1}{\left|\vec{T}\right|}$ for large $\left|\vec{T}\right|$, the sum over the absolute value of the terms still diverges as the number of equivalent terms with same G is increasing like G^2 in three dimensions. Additionally, the $G = 0$ term is ill-defined.

The Ewald method splits the potential into a short range $\tilde{\varphi}_i^S(\vec{G})$ and a long range $\tilde{\varphi}_i^L(\vec{G})$ part by exploiting the mathematical identity $\int_0^\infty e^{-G^2 t} dt = \frac{1}{G^2}$ and splitting the integral into two at $t = \frac{1}{4\eta^2}$.

$$\tilde{\varphi}_i(\vec{G}) = \tilde{\varphi}_i^S(\vec{G}) + \tilde{\varphi}_i^L(\vec{G}) \tag{2.33}$$

with

$$\tilde{\varphi}_i^S(\vec{G}) = e^{-i\vec{G}\vec{r}_i} \int_0^{\frac{1}{4\eta^2}} e^{-G^2 t} dt \quad \text{and} \quad \tilde{\varphi}_i^L(\vec{G}) = e^{-i\vec{G}\vec{r}_i} \int_{\frac{1}{4\eta^2}}^\infty e^{-G^2 t} dt \tag{2.34}$$

The long range part $\tilde{\varphi}_i^L(\vec{G})$ is integrated again and Fourier transformed to real space. This yields

$$\varphi_i^L(\vec{r}) = \frac{1}{V} \sum_{\vec{G}} \frac{1}{G^2} e^{-\frac{G^2}{4\eta^2}} e^{i\vec{G}(\vec{r}-\vec{r}_i)} \tag{2.35}$$

where a very fast convergence of the sum is guaranteed by the factor $e^{-\frac{G^2}{4\eta^2}}$. Attention has to be drawn to the term for $G = 0$ which is still ill-defined. The short range part is first Fourier transformed and then integration is carried out:

$$\varphi_i^S(\vec{r}) = \sum_{\vec{T}} \frac{1}{4\pi \left|\vec{r}-\vec{r}_i-\vec{T}\right|} \operatorname{erfc}\left(\eta \left|\vec{r}-\vec{r}_i-\vec{T}\right|\right) \tag{2.36}$$

with the complementary error function

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt = 1 - \operatorname{erf}(x). \tag{2.37}$$

The behaviour of the complementary error function for large $x \to \infty$ is like a Gaussian $\operatorname{erfc}(x) \sim e^{-x^2}$ which also assures a fast convergence. Overall, we end up with the potential at position $\vec{r}$ that is created by a charge carrier with charge q_i positioned at cell $\vec{r}_i$ and all its 3D periodic replica given by

$$\varphi_i(\vec{r}) = \sum_{\vec{T}} \frac{q_i}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}-\vec{r}_i-\vec{T}\right|} \operatorname{erfc}\left(\eta \left|\vec{r}-\vec{r}_i-\vec{T}\right|\right) + \frac{q_i}{V\varepsilon_0\varepsilon_r} \sum_{\vec{G}} \frac{1}{G^2} e^{-\frac{G^2}{4\eta^2}} e^{i\vec{G}(\vec{r}-\vec{r}_i)} \tag{2.38}$$

where both sums rapidly converge and the divergence of the initial summation is packed into the ill-defined term for $G = 0$. To get rid of this term, charge neutrality has to be established. This, however, is difficult if we are just interested in the potential of one charge carrier. We impose charge neutrality by positioning the charge carrier (for which we want to know the potential) with charge q at $\vec{r}_i = \vec{r}_1$ (including all its periodic images) and a second charge carrier with charge $-q$ at $\vec{r}_i = \vec{r}_2$ but leaving out the $\vec{T} = 0$ term in the sum. The potential of both charge carriers is now given by

$$\begin{aligned}
\varphi(\vec{r}) &= \sum_{\vec{T}} \frac{q}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}-\vec{r}_1-\vec{T}\right|} - \sum_{\vec{T}} \frac{q}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}-\vec{r}_2-\vec{T}\right|} + \frac{q}{4\pi\varepsilon_0\varepsilon_r |\vec{r}-\vec{r}_2|} && (2.39)\\
&= \sum_{\vec{T}} \frac{q}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}-\vec{r}_1-\vec{T}\right|} \operatorname{erfc}\left(\eta \left|\vec{r}-\vec{r}_1-\vec{T}\right|\right) - \sum_{\vec{T}} \frac{q}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}-\vec{r}_2-\vec{T}\right|} \operatorname{erfc}\left(\eta \left|\vec{r}-\vec{r}_2-\vec{T}\right|\right) \\
&\quad + \frac{q}{V\varepsilon_0\varepsilon_r} \sum_{\vec{G}\neq 0} \frac{1}{G^2} e^{-\frac{G^2}{4\eta^2}} e^{i\vec{G}(\vec{r}-\vec{r}_1)} - \frac{q}{V\varepsilon_0\varepsilon_r} \sum_{\vec{G}\neq 0} \frac{1}{G^2} e^{-\frac{G^2}{4\eta^2}} e^{i\vec{G}(\vec{r}-\vec{r}_2)} + \frac{q}{4\pi\varepsilon_0\varepsilon_r |\vec{r}-\vec{r}_2|}. && (2.40)
\end{aligned}$$

In (2.40) the $G = 0$ term already vanishes during the first Fourier transformation due to charge neutrality. The $\left|\vec{T}\right| = 0$ term can be taken out of the sums, so that

$$\begin{aligned}
\varphi(\vec{r}) = \sum_{\vec{T}\neq 0} q \frac{\operatorname{erfc}\left(\eta \left|\vec{r}-\vec{r}_1-\vec{T}\right|\right)}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}-\vec{r}_1-\vec{T}\right|} - \sum_{\vec{T}\neq 0} q \frac{\operatorname{erfc}\left(\eta \left|\vec{r}-\vec{r}_2-\vec{T}\right|\right)}{4\pi\varepsilon_0\varepsilon_r \left|\vec{r}-\vec{r}_2-\vec{T}\right|} + q \frac{\operatorname{erfc}(\eta |\vec{r}-\vec{r}_1|)}{4\pi\varepsilon_0\varepsilon_r |\vec{r}-\vec{r}_1|} \\
- q \frac{\operatorname{erf}(\eta |\vec{r}-\vec{r}_2|)}{4\pi\varepsilon_0\varepsilon_r |\vec{r}-\vec{r}_2|} + \frac{q}{V\varepsilon_0\varepsilon_r} \sum_{\vec{G}\neq 0} \frac{1}{G^2} e^{-\frac{G^2}{4\eta^2}} \left(e^{i\vec{G}(\vec{r}-\vec{r}_1)} - e^{i\vec{G}(\vec{r}-\vec{r}_2)}\right) \qquad (2.41)
\end{aligned}$$

where $1 - \operatorname{erfc}(x) = \operatorname{erf}(x)$ was used. The idea to get the potential of one charge carrier always possessing the same but finite potential offset is to set the position of the artificially introduced charge carrier $\vec{r}_2$ to the position for which we want to know the potential, i.e., $\vec{r} \to \vec{r}_2$. Except the fourth term, the limit can be evaluated straight-forwardly.

$$\lim_{\vec{r}\to\vec{r}_2} q \frac{\operatorname{erf}(\eta |\vec{r}-\vec{r}_2|)}{4\pi\varepsilon_0\varepsilon_r |\vec{r}-\vec{r}_2|} = \frac{q}{4\pi\varepsilon_0\varepsilon_r} \lim_{x\to 0} \frac{\operatorname{erf}(\eta x)}{x} = \frac{q}{4\pi\varepsilon_0\varepsilon_r} \lim_{x\to 0} \frac{\frac{2}{\sqrt{\pi}} e^{-(\eta x)^2} \eta}{1} = \frac{q\eta}{2\pi^{\frac{3}{2}}\varepsilon_0\varepsilon_r} \tag{2.42}$$

From the second to the third term, L'Hôpital's rule was used. By defining the difference, $\Delta\vec{r} = \vec{r}_1 - \vec{r}_2$, between the point where the charge carrier is sitting and the position where the potential is evaluated, we get

$$\begin{aligned}
\varphi(\Delta\vec{r}) = \frac{q}{4\pi\varepsilon_0\varepsilon_r} \Bigg[\frac{\operatorname{erfc}(\eta\Delta r)}{\Delta r} - \frac{2\eta}{\sqrt{\pi}} + \sum_{\vec{T}\neq 0} \left(\frac{\operatorname{erfc}\left(\eta \left|\Delta\vec{r}+\vec{T}\right|\right)}{\left|\Delta\vec{r}+\vec{T}\right|} - \frac{\operatorname{erfc}\left(\eta \left|\vec{T}\right|\right)}{\left|\vec{T}\right|} \right) \\
+ \frac{4\pi}{V} \sum_{\vec{G}\neq 0} \frac{e^{-\frac{G^2}{4\eta^2}}}{G^2} \left(\cos\left(\Delta\vec{r}\vec{G}\right) - 1\right) \Bigg] \qquad (2.43)
\end{aligned}$$

where the symmetric summation over $\vec{G}$ was used to eliminate the imaginary part of the complex exponential. Both sums in this equation can be easily calculated and rapidly converge. The only remaining question is how to choose the cutting point η to separate the short-range from the long-range region. If we are choosing η to high, then the sum over $\vec{T}$ converges fast but the sum over $\vec{G}$ lasts very long. If we choose η to low it is the other way round. Introducing a comparable summation variable $u = \sqrt{i^2+j^2+k^2}$ for $\vec{T} = l \cdot N(i,j,k)^\top$ and $\vec{G} = \frac{2\pi}{l \cdot N}(i,j,k)^\top$ with $i,j,k \in \mathbf{N}$, the two sums converge like $e^{-(\alpha u)^2}$ with $\alpha_T = \eta l \cdot N$ and $\alpha_G = \frac{\pi}{\eta l \cdot N}$. The fastest convergence is given if both sums converge in the same way, i.e., $\alpha_T = \alpha_G$, which leads to $\eta = \frac{\sqrt{\pi}}{l \cdot N}$. With this condition, our Ewald summation for a system with three dimensional periodicity is complete.

2D-Ewald Summation

In a system with periodicity only in two directions the derivation in principle works in the same way but with some more subtle considerations about the $G = 0$ term which is described in detail in [31]. The basic idea how to get charge neutrality is the same and the limit $\vec{r} \to \vec{r}_2$ can be performed as shown above. The final expression is a bit longer

$$\begin{aligned}\varphi(\Delta\vec{r}) = \frac{q}{4\pi\varepsilon_0\varepsilon_r}\Bigg\{ & \frac{\operatorname{erfc}(\eta\Delta r)}{\Delta r} + \frac{2\eta}{\sqrt{\pi}} + \frac{2\sqrt{\pi}}{A\eta}\left(1 - e^{-\eta^2|\Delta x_3|^2}\right) - \frac{2\pi|\Delta x_3|}{A}\operatorname{erf}(\eta|\Delta x_3|) \\ & + \sum_{\vec{T}\neq 0}\left(\frac{\operatorname{erfc}\left(\eta\left|\Delta\vec{r}+\vec{T}\right|\right)}{\left|\Delta\vec{r}+\vec{T}\right|} - \frac{\operatorname{erfc}\left(\eta\left|\vec{T}\right|\right)}{\left|\vec{T}\right|}\right) \\ & + \frac{\pi}{A}\sum_{\vec{G}\neq 0}\Bigg[\frac{\cos\left(\vec{G}\Delta\vec{r}\right)}{G}\left(e^{G|\Delta x_3|}\operatorname{erfc}\left(\frac{G}{2\eta}+\eta|\Delta x_3|\right) + e^{-G|\Delta x_3|}\operatorname{erfc}\left(\frac{G}{2\eta}-\eta|\Delta x_3|\right)\right) \\ & - \frac{2\operatorname{erfc}\left(\frac{G}{2\eta}\right)}{G}\Bigg]\Bigg\} \end{aligned} \tag{2.44}$$

with the distance between the charge carrier and the position where the potential of the charge carrier is calculated $|\Delta x_3|$ perpendicular to the 2D periodic plane and the area $A = (l \cdot N)^2$ of the periodic plane. The lattice constant is given by l and the system size by $N \times N \times N_3$ where N_3 is chosen regarding to chapter 2.2.2. The convergence of those sums is for large $\vec{T}$ and $\vec{G}$ again $e^{-(\alpha u)^2}$ and the cutting point η is, like in the 3D case, again $\eta = \frac{\sqrt{\pi}}{l \cdot N}$. Especially for large $|\Delta x_3|$, the convergence can be numerically instable, so a careful numerical treatment of the terms and sums is recommended. Taking the same potential with a different sign of the charge for the interaction between the charge carriers and their image charges guarantees the correct boundary condition $\varphi = 0$ at the metal interface.

2.3 Troubleshooting

This chapter is not thought to give a recipe for debugging a code, it discusses some probably surprising effects of numerical method errors. Most of them are associated with the creation and application of random numbers.

2.3.1 Time Cumulation

When the simulation is performed in FRM or with the new update mechanism, differences of times are calculated very often. At a certain Markov time t_i after i Markov jumps we have retention times Δt_i^j for all possible processes j. The fastest process k with the shortest retention time Δt_i^k is performed. Its retention time and probably some more retention times of the processes j_{new} are recalculated and the new retention times are given by $\Delta t_{i+1}^{j_{new}}$. The retention times of the processes j_{old} that are not recalculated can be retrieved by subtracting the retention time of the performed process $\Delta t_{i+1}^{j_{old}} = \Delta t_i^{j_{old}} - \Delta t_i^k$. The total

Markov time is evolved by $t_{i+1} = t_i + \Delta t_i^k$.

An alternative and, at first sight, faster way is to use total times rather than time differences. This means that for Markov jump i at Markov time t_i we do not have retention times but total Markov times t_i^j when process j would happen in total Markov time. Again the process k with the lowest time t_i^k is performed and the Markov time is evolved to this time $t_{i+1} = t_i^k$. New retention times $\Delta t_{i+1}^{j_{new}}$ are evaluated for the processes j_{new} and the Markov time at which those processes would occur is calculated $t_{i+1}^{j_{new}} = t_i^k + \Delta t_{i+1}^{j_{new}}$. The times for processes j_{old}, that are not recalculated, can be used unchanged $t_{i+1}^{j_{old}} = t_i^{j_{old}}$. Especially in FRM, where only one process is recalculated, some computational time can be saved with this method. However, as this manipulation of the retention times is of linear order $\mathcal{O}(N_{cc})$ for the charge carriers N_{cc}, the difference in computational cost is not really significant. Considering this, it is remarkable, as we will show below, that the mathematically identical method produces numerically different results.

We performed bulk simulations with the same sequence of random numbers, once with retention times Δt_i^j and once with cumulated times t_i^j. The energetic landscape and the starting configuration was the same. Two charge carriers were put into bulk and 20 billion hops (+10% thermalisation) were performed. The used parameters are shown in tab. 2.6. For this methodological test, the interactions were turned off.

Table 2.6: Used parameters to demonstrate the error due to time cumulation.

ν_0	$2.25 \cdot 10^{13}$ s^{-1}
ε_r	4.0
F	$1.0 \cdot 10^8$ Vm^{-1}
T	300 K
σ	100 meV
$N_1 \times N_2 \times N_3$	$51 \times 51 \times 51$
l	1.0 nm
α	$\frac{5}{l} = 5 \cdot 10^9$ m^{-1}
r_c	0.0 nm

The results of the two methods are shown in fig. 2.8. On the x-axis the progress of the simulation $\frac{i}{N_M}$ is shown, which is the current Markov jump i divided by the total amount of Markov jumps simulated N_M. The y-axis shows the mobility μ that is calculated by averaging over the time series up to the current Markov jump.

At the very beginning, i.e., up to a progress of about 0.03, the two methods produce the same mobilities. Note that a progress of 0.03 already refers to a number of 600 million Markov jumps which is already quite a lot. Having a very detailed look at those mobilities at the beginning, they are differing but to a negligible extent. After those 600 million Markov jumps, the two curves are diverging continuously. The method of the retention times Δt_i^j is described more or less by a straight line (as it should be) and the method of the cumulated times t_i^j bends down. At a progress of about 0.42, a distinctive kink appears for the curve of the cumulated time method and after this kink, the mobility for this method is no longer trustworthy at all. This figure illustrates that it is worth to invest a bit more computer time and work with retention times rather than with cumulated ones.

To interpret the difference between the two methods, numerical rounding has to be taken into account. Our random numbers are floating point accuracy which means that the difference between two uniformly distributed random numbers between 0 and 1 is in the order of 10^{-8}. The total Markov time is calculated with double precision which means that rounding errors of about 10^{-16} can appear. After $6 \cdot 10^8$ Markov jumps, the fastest times out of our exponentially distributed random numbers are not distinguishable any longer, as the rounding connected to the time cumulation cuts off the short time differences of the hops. Accordingly, the hopping times for the two charge carriers, which are in our simulation, will sometimes be equal. In this case, the code always selects the first charge carrier and, due to this artificial preference, we are sampling out of a wrong probability distribution. The onset of the wrong probability distribution is continuously moving to higher and higher hopping times until it reaches a certain threshold (at a

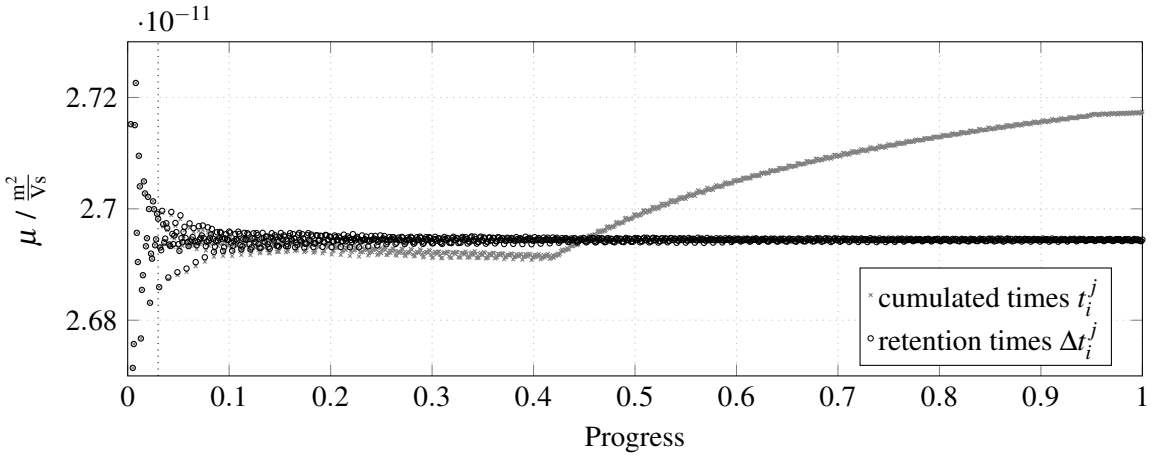

Figure 2.8: Comparison between the usage of a retention time Δt_i^j and a cumulated time t_i^j for Markov jump i and charge carrier j. For this comparison the averaged mobility μ up to a certain Markov time i with progress $\frac{i}{N_M}$ is plotted, N_M is the total number of Markov jumps. Above a progress of about 0.03 (visualised by a dotted line), the cumulation of the retention times to an absolute Markov time t_i^j has significant numerical problems (detailed discussion see text).

progress of about 0.42) where the sample-path behaviour deviates from the original one so strongly that a different stationary distribution is approached by the Markov chain.

2.3.2 Random Number Generator

Special attention has to be drawn to the random number generator. In Fortran there are two different ones, `rand()` and `random_number()`. `rand()` is a fast random number generator which gains its pseudo-random numbers from a simple modulo generator. The drawback of this simple method is that the random numbers can be correlated and the periodicity of such a random number generator is low. The periodicity is given by the number of random numbers that are created until the sequence of random numbers is repeated. A bulk simulation with two charge carriers, nearest neighbour hopping with Miller-Abrahams rates and in total 4 billion hops (+10% thermalisation) was simulated. The parameters of the simulation are the same as in tab. 2.6 but with a different sequence of random numbers and hence a different energetic landscape. So it is not surprising, that the mobility in this simulation converges to a slightly different value than before. The convergence of the mobility μ over the Markov jumps is shown in fig. 2.9.

The graph shows a pronounced periodic behaviour, but such a periodicity could also belong to a very long autocorrelation time. So what makes us so sure that this is caused by the random number generator `rand()`? In this simulation, the hopping time to each neighbour assigned an exponentially distributed random time and all times are recalculated after a hop. This means that we need 12 random numbers for each of the 4 billion Markov jumps yielding a total of 48 billion random numbers in the simulation. The difference between the two very significant peaks at progress (0.399 ± 0.001) and (0.444 ± 0.001) belongs to one period. In this period $(2,160 \pm 96)$ million random numbers were used which belongs to a periodicity of $2^{(31.01 \pm 0.07)}$. The random number generator `rand()` is supposed to have a period of 2^{31} which perfectly matches with our observed one. This is already a very strong indicator that the periodicity is not caused by an enormously large autocorrelation time. An even stronger indicator is, that this periodic behaviour disappears as soon as we take the random number generator `random_number()` with a guaranteed periodicity higher than 2^{123} (not shown).

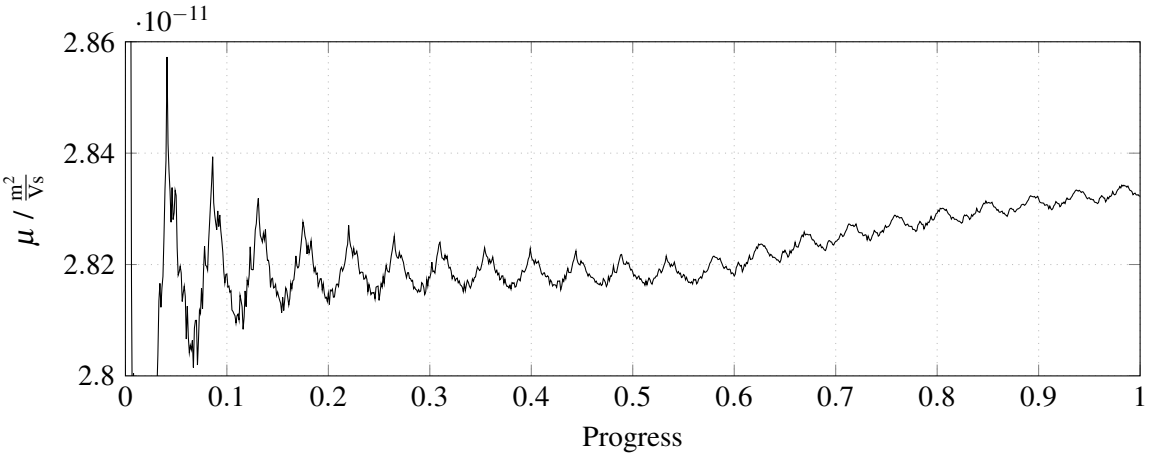

Figure 2.9: Convergence of the mobility μ over the progress of the bulk simulation. The periodic behaviour is caused by the periodicity of the Fortran random number generator `rand()`. The time cumulation discussed in chapter 2.3.1 induces the kink at a progress of about 0.6.

2.3.3 Upper and Lower Limits of the Random Numbers

If we think of the continuous probability distribution of our exponentially distributed random number with rate parameter R, it is clear that the limits 0 and ∞ have to be excluded from the probability density. The lower limit 0 has to be excluded as we are not allowing simultaneous events for our Poisson process and the upper limit ∞ cannot be reached $\lim_{C\to\infty}\int_C^\infty Re^{-Rt}dt = 0$. From a random number generator we usually get uniformly distributed random numbers $\xi = \texttt{random\_number}()$ in the interval $[0,1)$ which means that the lower limit $\xi = 0$ is included and would lead to $t = -\frac{\log(\xi)}{R} = \infty$. This would cause problems as the time $t = \infty$ should not be reached. Taking a random number $\xi = 1 - \texttt{random\_number}()$ leads to a random number out of the interval $(0,1]$ and a time $t = 0$ can appear for $\xi = 1$. This choice can also cause severe problems as shown in the scenario below. As a consequence, both limits have to be excluded.

The system, in which severe problems were observed when $t = 0$ is not excluded from the random number generation, was an injection simulation with the parameters shown in tab. 2.7. For bulk hopping, only nearest neighbour hopping was allowed, and for injections, a hopping radius of 2 nm was used. Interactions were considered correctly with Ewald summation and the system size was chosen to give a return probability of about 10^{-6}.

Table 2.7: Used parameters to show the error due to the wrong limits of the random numbers.

ν_0	$2.25 \cdot 10^{13}$ s^{-1}
ε_r	4.0
F	$3.78 \cdot 10^{7}$ Vm^{-1}
T	300 K
σ	100 meV
Δ	0.8 eV
$N_1 \times N_2 \times N_3$	$51 \times 51 \times 13$
l	1.0 nm
α	$\frac{5}{l} = 5 \cdot 10^{9}$ m^{-1}

The simulation was performed for different injection update radii $r_{up,inj}$ and DMC. The escape current density j_{esc} was measured by counting the number of escape events over time (see chapter 1.1.4). In fig. 2.10, the obtained current densities over the update radius are shown. The straight line corresponds to the DMC results. All simulations are done with the same energetic landscape and the errors are

calculated with Jackknife.

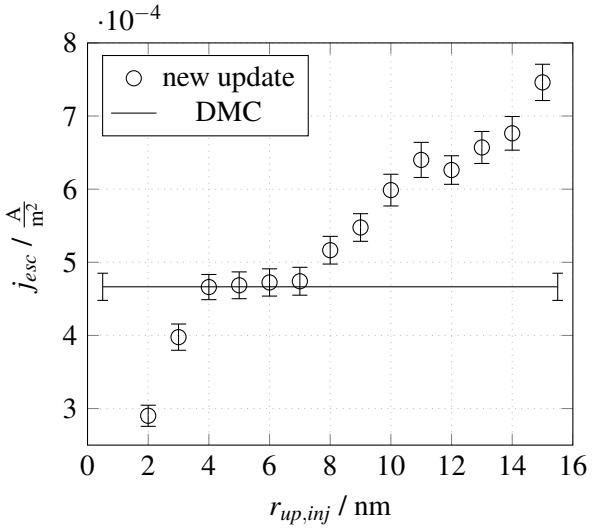

Figure 2.10: Divergence of the new update method compared to a DMC simulation due to the wrong choice of the limits of the generated random numbers. The escape current density j_{esc} of the injection simulation is plotted over the update radius $r_{up,inj}$ and the straight line shows the results of the proper DMC simulation. For details see text.

To understand the data presented in fig. 2.10, we have to go into the matter very deeply. Injection simulations are mainly governed by the interplay between the zero-field-energy barrier Δ, the own image charge of the charge carrier, and the externally applied electric field. Depending on those quantities, the charge carrier densities can vary from very high to extremely low values. In this case, we have a high zero-field-energy barrier $\Delta = 0.8$ eV and a rather low electric field $F = 3.78 \cdot 10^7$ Vm^{-1} and, thus, a very low charge carrier density. Additionally, the disorder $\sigma = 100$ meV is very high, so most injections are performed to low lying energy levels in the first layer. From there, all hops are upwards in energy except the recombination with the contact. So the recombination is the most likely event and only a very small part of the injected charge carriers can escape over the remaining barrier. If we now think of a random number $\xi = 1$ that creates a retention time $\Delta t = 0$, independent of the energy difference between the two cells, this random number chooses one process that will be performed immediately. If this was an injection, then the injected charge carrier will populate a cell with a rather high energy level compared to normal injections and approximately half of the cells surrounding the charge carrier will be lower in energy. This means that it is very likely that the charge carrier starts its journey towards the escape layers. Consequently, the randomly chosen injection with retention time $\Delta t = 0$ in this special scenario strongly biases the escape current density towards higher values. With this insight, we can again look at fig. 2.10. For a DMC calculation we need two random numbers per Markov jump to get, (i), the randomly chosen process out of all processes and, (ii), the retention time. This is the lowest possible amount of required random numbers that is also needed in FRM. There we also need one random number to pick the neighbour of the recalculated process and a second one for the retention time. With our new update mechanism we need two random numbers for each process that is recalculated. This leads to the fact that the higher the update radius $r_{up,inj}$, the more injections need to be recalculated in each Markov jump and the more random numbers are required. Hence also the random number $\xi = 1$ is produced more often and the escape current density is biased more strongly with increasing update radius $r_{up,inj}$. This is exactly the behaviour that can be seen in fig. 2.10. For a very low update radius, our new method quickly converges to the accurate DMC results. For $r_{up,inj} = 4$ nm to $r_{up,inj} = 7$ nm, the measured mobilities are the same within their error bars. At a certain threshold of the update radius of about $r_{up,inj} = 8$ nm, the escape current density created by the randomly chosen injections starts to dominate the correct escape current density. The results for our new update mechanism rise according to the increasing amount of random

numbers needed for one Markov jump. When $\Delta t = 0$ is excluded from the simulation, this behaviour is not observed (not shown).

To summarise all those problems, we can say that special attention has to be drawn to the application of random numbers. It is recommended to generate as little random numbers as possible from the correct discretised probability density and process them as direct as possible. Doing so prevents many unnecessary problems in advance.

3 Results

With a working program in our hands, it is time to test the new update mechanism that we have developed so far. To get a feeling for kinetic Monte Carlo simulations, the method is tested for bulk simulations. With the knowledge gained in the bulk simulations, an efficient systematic check of the correctness of our new method for injection simulations can be developed.

3.1 Bulk Simulations

Our new update mechanism should be able to cope with high charge carrier densities, so it is quite obvious that the most important parameter to be investigated with our new update mechanism is the charge carrier density. This quantity can be easily altered in bulk simulations by putting a certain initial number of charge carriers into the simulation. As the number of charge carriers does not change due to the periodic boundary conditions, the charge carrier density stays at this initial value. Another important parameter is the energetic disorder σ which can dramatically change the behaviour of our charge carriers in the simulation. Besides those important parameters, there are many other parameters that have effects on the simulation, but those effects are not as substantial as the effects of the two parameters charge carrier density and energetic disorder. As an example, the externally applied electric field F influences the results. However, especially for Miller-Abrahams rates (which we will be using exclusively in chapter 3), it mainly changes the depth of traps. The stronger the fields the higher the energy difference between two layers perpendicular to the field and the lower the traps get. Some parameters probably significantly change the behaviour such as the lattice constant l or the relative permittivity ε_r, but they are more or less physically determined for the simulation of organic semiconductors. All those parameters that are kept constant for the bulk simulation are listed in tab. 3.1.

Table 3.1: Used parameters for the bulk simulations to compare the new update mechanism and FRM to the accurate DMC.

ν_0	$2.25 \cdot 10^{13}$ s^{-1}
ε_r	4.0
F	$1.0 \cdot 10^{8}$ Vm^{-1}
T	300 K
$N_1 \times N_2 \times N_3$	$51 \times 51 \times 51$
l	1.0 nm
α	$\frac{5}{l} = 5 \cdot 10^{9}$ m^{-1}

Only nearest neighbour hopping is allowed and Miller-Abrahams rates (1.100) are used to calculate the hopping rates. Interactions between all charge carriers including all their periodic images are considered with Ewald summation method (see chapter 2.2.3). The randomly chosen energetic landscape was always calculated with the same seed, which means that the same sequence of random numbers was used and, hence, the same energetic landscape is simulated. As we are only comparing the different update mechanisms we do not need to average over multiple grids. The comparability between the different update mechanisms is guaranteed much better, when a specific energetic landscape being representative for the chosen energetic disorder is taken to measure the same quantities on the same system. In each simulation 10 to 20 million hops (+10% thermalisation) are performed. The quantity that is measured is the bulk mobility μ and errors are calculated with Jackknife (see chapter 1.1.4).

A typical result for a simulation can be seen in fig. 3.1 in which the bulk mobility μ is plotted with respect to the update radius $r_{up,cc}$. At $r_{up,cc} = 0$ nm, the results of the FRM are plotted. The straight line with error bars at the beginning and the end indicates the mobility of the DMC method to which the mobilities of the update radius method should converge for an increasing update radius $r_{up,cc}$. This simulation was performed with an intermediate overall charge carrier density of $n_{cc}^{cell} = 0.02$ ccpc (charge carriers per cell) and no energetic disorder $\sigma = 0$ meV. For this case, FRM still produces quite good results

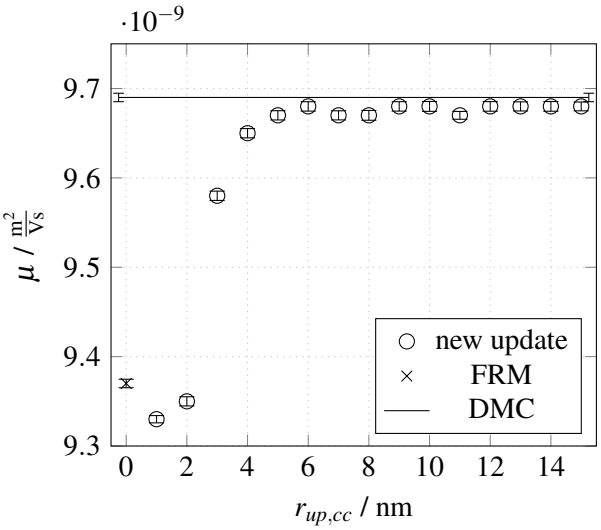

Figure 3.1: Mobility μ over the update radius $r_{up,cc}$ for an overall charge carrier density of $n_{cc}^{cell} = 0.02$ ccpc and an isoenergetic landscape $\sigma = 0$ meV. The new update mechanism converges with increasing update radius $r_{up,cc}$ to the accurate DMC results.

with an error of the mobility below 4% compared to the DMC mobility. Our new update mechanism reproduces the FRM mobilities for a low update radius $r_{up,cc} \leq 2$ nm and then shows a quick transition to a DMC-like behaviour. For an update radius above $r_{up,cc} \geq 5$ nm the results are within their error bars approximately the same. This result is, in principle, what we expected, i.e., for $r_{up,cc} \to 0$ we get FRM and for $r_{up,cc} \to \infty$ we get DMC with our new update mechanism. Furthermore, the graph reveals that the DMC limit is reached already with $r_{up,cc} \approx 5$ nm.

An extrapolation from this very special case to a general one without any further data is not possible. To gain more insight we have to think about the effects that are leading to the convergence of our new update method. The two effects that we make responsible for the convergence are already discussed in [28] in a slightly different context. In general, the two effects are mixing and the convergence is due to an intricate interplay of those two effects. For very high charge carrier densities, the two effects influence the mobility nearly additive, so we have a look at overall charge carrier densities of $n_{cc}^{cell} = 0.1$ ccpc. Furthermore one of those two effects is only observed for a disordered system, i.e., $\sigma > 0$ meV.

3.1.1 Blocking by Charge Carrier Density Modulation

We refer to the effect that is always present, independent of the energetic disorder σ, as blocking by charge carrier density modulation. The reason for this effect is illustrated in fig. 3.2.

If we now take a look at fig. 3.3, we can explain the convergence of the mobility for an increasing update radius $r_{up,cc}$ by exactly this effect (the same argumentation holds for the convergence of the mobility shown in fig. 3.1). For a charge carrier density of $n_{cc}^{cell} = 0.1$ ccpc, one charge carrier has an average share of 10 nm^3. This corresponds to a sphere with radius 1.34 nm and an approximate average distance between two charge carriers of 2.67 nm. If a charge carrier now hops 1 nm into a certain direction, the next charge carrier in forward direction is still 1.67 nm away and in the backward direction the distance is in average 3.67 nm. With an update radius $r_{up,cc}$ of 1 nm this implies that we will, on average, not include any other charge carrier in our update sphere but the currently hopped one. So it is no surprise that changing the update mechanism from FRM to $r_{up,cc} = 1$ nm does not change the simulated mobility. Increasing the update radius to $r_{up,cc} = 2$ nm already includes the charge carrier in forward direction and by further increasing the update radius to $r_{up,cc} = 4$ nm, the rates for nearly all charge carriers surrounding the charge carrier density modulation are recalculated and the modulation is balanced nearly correct. As we are talking about average distances, it is clear that the exact convergence needs a higher update radius.

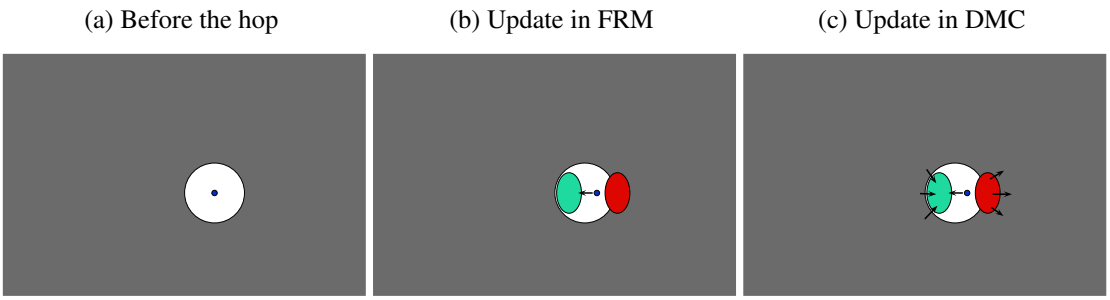

Figure 3.2: Blocking of the motion of a charge carrier by the modulation of the charge carrier density after a hop. In the left picture the homogeneous charge cloud (grey) symmetrically surrounds the blue charge carrier in its white sphere where no other charge carriers are in. If it hops, a reduced charge carrier density is created behind it (turquoise) and an increased charge carrier density in front of it (red). Due to this modulation of the charge carrier density and the Coulomb repulsion between the charge carriers, the charge carrier is forced back to the position where it previously hopped away from. In FRM (middle picture) the previously hopped charge carrier is the only one that feels this force as all other rates are unchanged. For a DMC simulation (right picture) the charge carrier density modulation can be balanced by all surrounding charge carriers and the probability that the previously hopped charge carrier hops back straight is reduced. This leads to a reduction of the blocking. As a consequence, FRM underestimates the mobility μ.

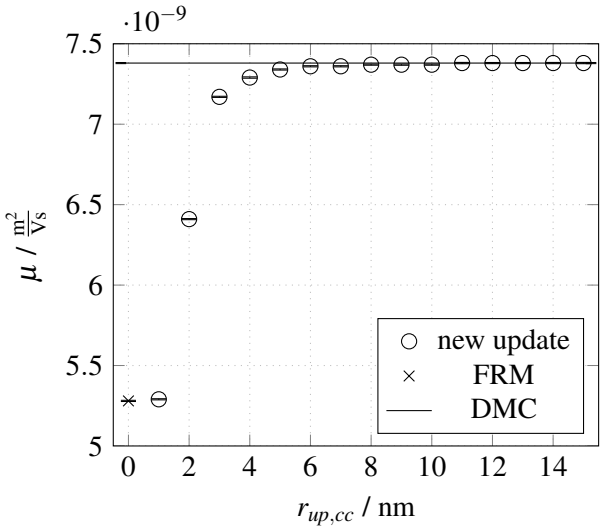

Figure 3.3: Mobility μ over the update radius $r_{up,cc}$ (including error bars) for an overall charge carrier density of $n_{cc}^{cell} = 0.1$ ccpc and an isoenergetic landscape $\sigma = 0$ meV. The convergence of the new update mechanism is governed by the effect of blocking due to charge carrier density modulations (see text).

Nevertheless the major part of the convergence is completed within a low update radius.

3.1.2 Detrapping

The second effect mentioned above, that is associated with a certain energetic disorder $\sigma > 0$ meV, will be called detrapping here. It is the effect that charge carriers can help other charge carriers out of traps when they come close enough to each other, discussed in fig. 3.4.

In fig. 3.5, the mobilities for an energetic disorder of $\sigma = 50$ meV (left panel) and $\sigma = 100$ meV (right panel) is shown. The charge carrier density is $n_{cc}^{cell} = 0.1$ ccpc like above. This means that fig. 3.3 completes the series with $\sigma = 0$ meV.

When we take a look at those graphs (fig. 3.3 and fig. 3.5), it can be seen that qualitatively the convergence

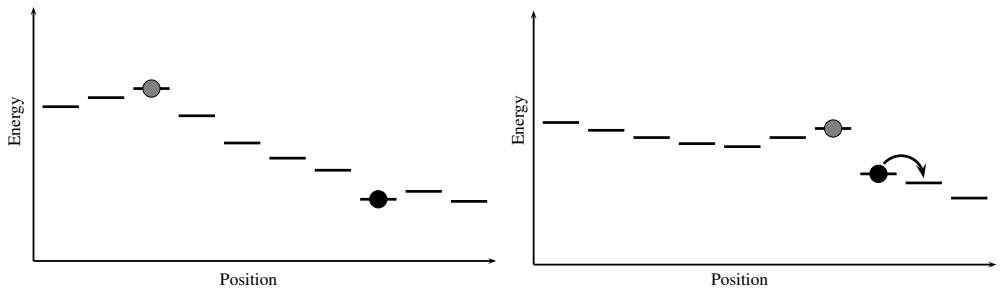

Figure 3.4: Illustration of the effect of detrapping induced by approaching charge carriers. In both panels the energy levels E_i of the cells at position i seen by the black charge carrier are indicated by different heights of the bars. The energy is including the energetic disorder (the black charge carrier is sitting in a trap), an externally applied field (the linearly decreasing energy from left to right) and the interaction with the grey charge carrier (see also (1.109)). Left panel: The black charge carrier is sitting in a trap due to a sufficiently large distance between the two charge carriers. Right panel: When the grey charge carrier is following the electric field and comes closer and closer to the black charge carrier, the trap site does not act as trap any longer. So the black charge carrier is easily released.

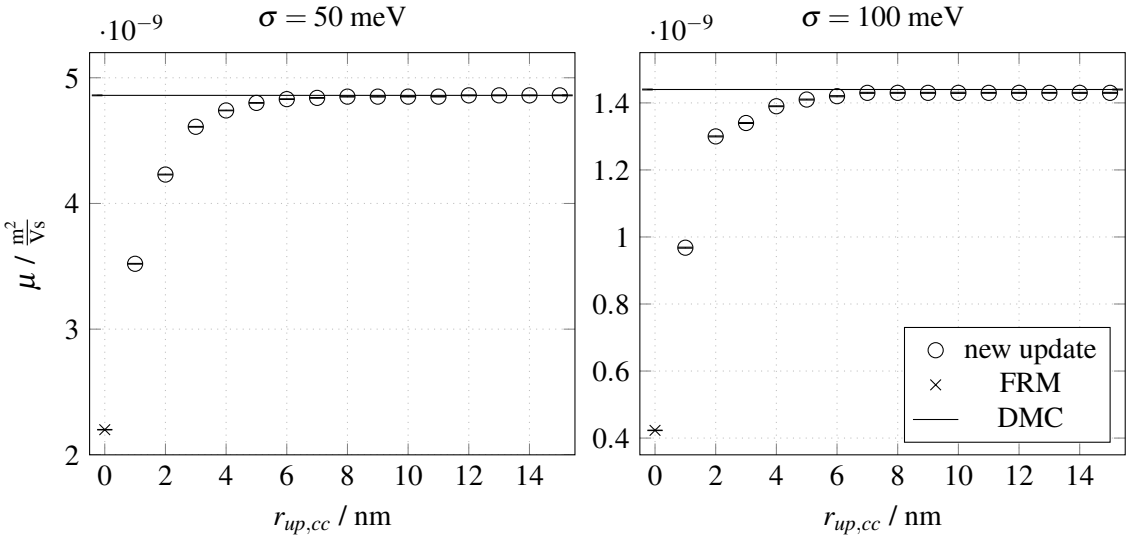

Figure 3.5: Mobility μ over the update radius $r_{up,cc}$ (including error bars) for an overall charge carrier density of $n_{cc}^{cell} = 0.1$ ccpc and different energetic disorders σ. The convergence of the new update mechanism is governed by both the effect of blocking due to charge carrier density modulations and detrapping due to approaching charge carriers (see text).

of the new update mechanism for $r_{up,cc} \geq 3$ nm is approximately the same for all energetic disorders σ. This convergence is mainly dominated by the blocking effect. Due to the reduction of homogeneity for increasing disorder, the convergence is almost but not exactly the same. Especially for $r_{up,cc} = 3$ nm the inhomogeneity induced by the energetic disorder influences the mobility. The most prominent change due to an increasing disorder is the apparent jump in mobility from FRM to $r_{up,cc} = 1$ nm. In the isoenergetic case (see fig. 3.3), FRM and $r_{up,cc} = 1$ nm are more or less identical. At non-zero disorder $\sigma > 0$ meV (see fig. 3.5), however, there is a significant change in the mobility. This apparent jump is nearly exclusively generated by the energetic disorder. It is intuitive that the effect of detrapping is most pronounced for a close proximity of the charge carriers, as then the Coulomb interaction is strongest and detrapping is most efficient.

3.1.3 Error of the New Update Mechanism

As we now understand the effects that mainly govern the convergence of our new update mechanism, we can take a look at the error of the update mechanisms compared to the accurate DMC method. This error $\Delta\mu$ is related to the measured mobility μ_{method} of the method compared with the proper results of a DMC simulation μ_{DMC}:

$$\Delta\mu = \left| \frac{\mu_{DMC} - \mu_{method}}{\mu_{DMC}} \right|. \tag{3.1}$$

The values of $\Delta\mu$ are shown for FRM and $r_{up,cc} = 1$ nm to $r_{up,cc} = 5$ nm in fig. 3.6, and for $r_{up,cc} = 6$ nm to $r_{up,cc} = 11$ nm in fig. 3.7.

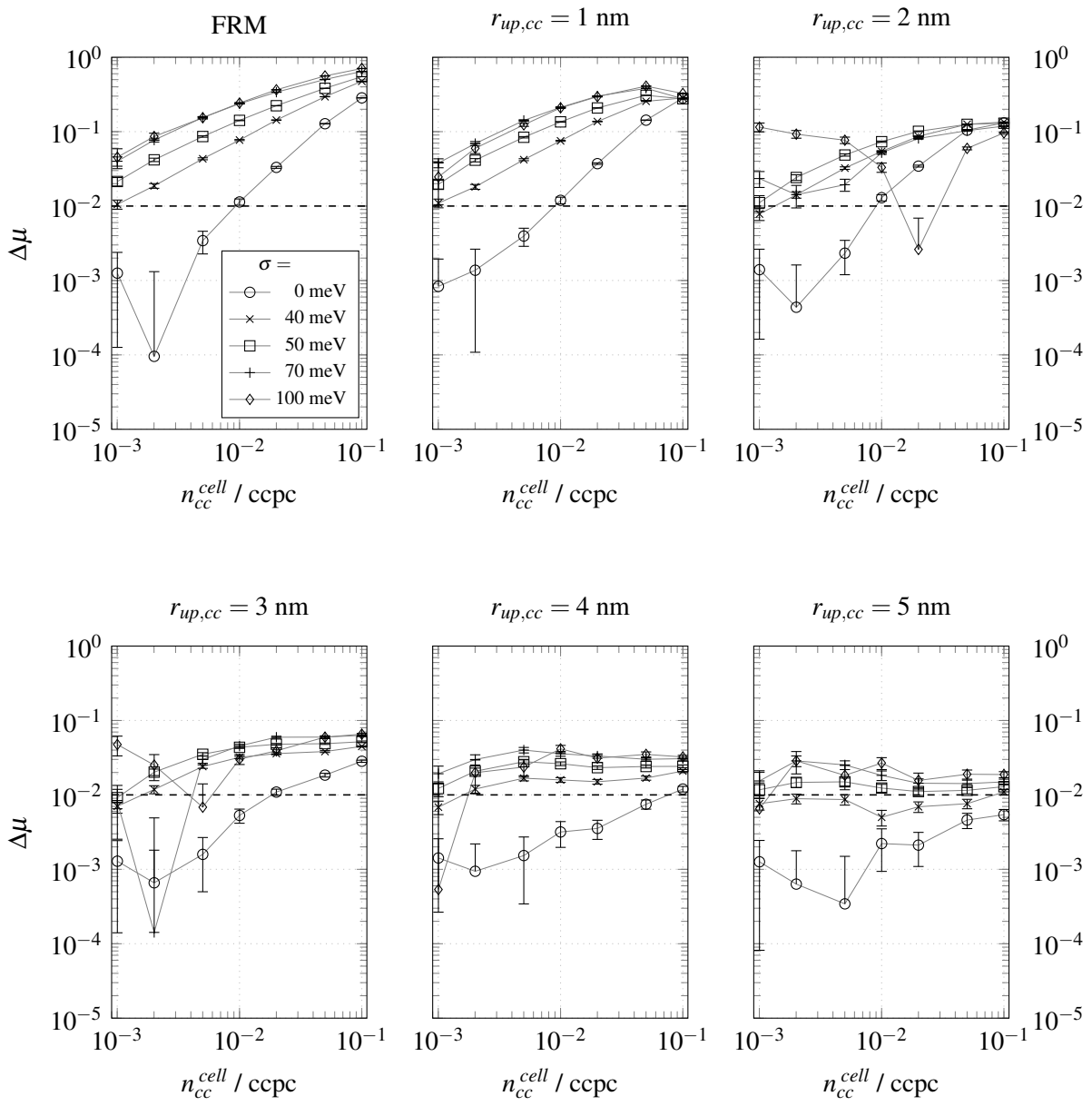

Figure 3.6: Error of the mobility $\Delta\mu$ achieved by the update mechanism compared to the accurate results received by a DMC simulation as a function of the charge carrier density n_{cc}^{cell}. The investigated update mechanisms are FRM and our new update mechanism in an update radius range of $r_{up,cc} = 1$ nm to $r_{up,cc} = 5$ nm. The data for higher update radii is found in fig. 3.7. The error is reduced for an increasing update radius (discussion see text). The dashed line at an error of 1% is do guide the eye.

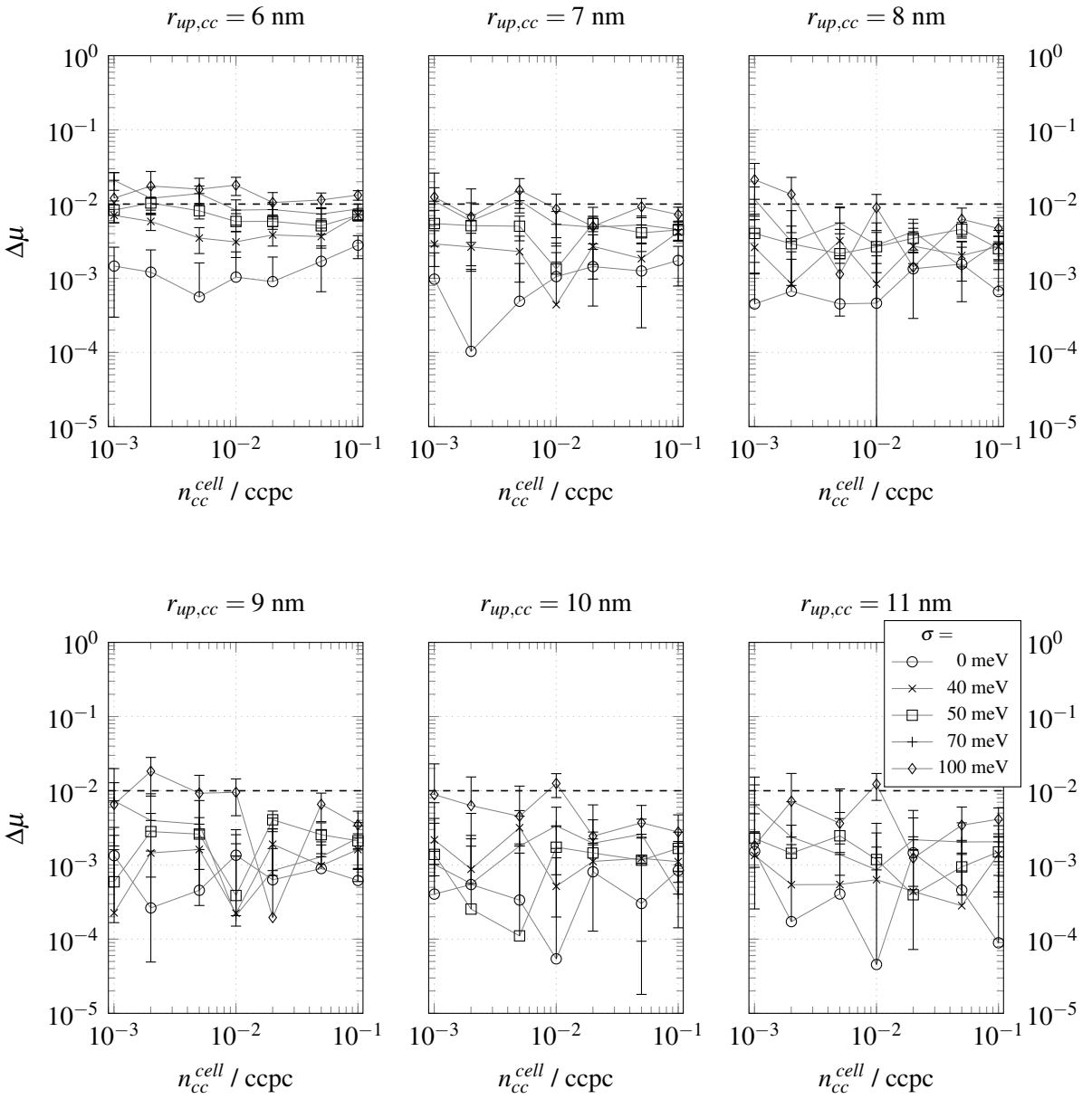

Figure 3.7: Error of the mobility $\Delta\mu$ achieved by the update mechanism compared to the accurate results received by a DMC simulation as a function of the charge carrier density n_{cc}^{cell}. The new update mechanism was tested for update radii from $r_{up,cc} = 6$ nm to $r_{up,cc} = 11$ nm. The evaluation of the FRM and lower update radii is shown in fig. 3.6. The reduction of the error seems to stagnate with increasing update radius $r_{up,cc}$ (discussion see text). The dashed line at an error of 1% is do guide the eye.

Simulations for even higher update radii were performed but did not show significant changes compared to the $r_{up,cc} = 11$ nm case. When we look at the top row of fig. 3.6, the improvement of our new update mechanism compared to FRM can be readily seen. For high charge carrier densities, the FRM totally fails with errors in the order of 70% (which is a factor of 3). If we go from FRM to $r_{up,cc} = 1$ nm, we see the behaviour discussed above, i.e., that the effect of detrapping is already partially considered which leads to reduced errors compared to FRM for high energetic disorders. The reduction is seen for all charge carrier densities but most pronounced for the highest ones. A further increment of the update radius $r_{up,cc}$ leads to a mixing of the two effects discussed above. If we have a look at the plot for $r_{up,cc} = 2$ nm, we see that for low charge carrier densities $n_{cc}^{cell} = 10^{-3}$ ccpc and high energetic disorder $\sigma = 100$ meV the error of our new update mechanism is even higher than for FRM. If we have a look at the mobility

for this point we see that the mobility of our simulation is higher than the one of the DMC calculation. This behaviour might be described by the fact that charge carriers can detrap other charge carriers as they are coming close enough to each other, but for such a low update radius the distance between the two charge carriers is already so small that the trapped charge carrier is catapulted out of the trap to fast. With increasing update radius the charge carriers are detrapped more smoothly and the results of the simulation are getting better and better. For an update radius of $r_{up,cc} = 4$ nm the charge carrier density dependency has already flattened out and with further increasing update radius (see also fig. 3.7) the error further reduces. The last significant drop of the errors is observed for the change from $r_{up,cc} = 7$ nm to $r_{up,cc} = 8$ nm where nearly all errors including their error bars are below 1%, which means that the lower limit of the error bar of all data points lies below 1%. As an error below 1% is sufficient for most KMC simulations, an update radius of $r_{up,cc} = 8$ nm for bulk hopping will be sufficient for our requirements. Especially for high energetic disorders, the variations between different energetic landscapes is much higher than 1%. So the methodological error made with an update radius of $r_{up,cc} = 8$ nm should be negligible compared to the Monte Carlo error and the error created by averaging over multiple energetic landscapes.

3.2 Injection Simulations

In contrast to bulk simulations, it is more difficult to systematically check the correctness of our new update mechanism as we can no longer choose the charge carrier density n_{cc}^{cell}. Instead, the charge carrier density is determined by the interplay between the zero-field-energy barrier Δ, the externally applied electric field strength F, and the image charge potential. So the charge carrier density, being an important parameter for bulk simulations, is replaced by Δ and F as the two important parameters for injection simulations.

For all simulations in this chapter, Miller-Abrahams rates (1.100) were used. For charge carriers in the organic semiconductor, only nearest neighbour hops are allowed $r_{hop,cc} = 1$ nm and also recombination with the contact is restricted to nearest neighbour hopping. Injection cells can inject up to a radius of $r_{hop,inj} = 2$ nm where the efficiency of the calculation of injection rates was increased by the usage of the combined spatial decay term $\sum_{i \in \text{injection region}} e^{-2\alpha r_{ij}}$ discussed in chapter 2.1.4. All interactions with all periodic replica and all image charges and their periodic replica are considered in exact fashion with the help of two dimensional Ewald summation (see chapter 2.2.3). In the same way as in chapter 3.1, (i), only one specific energetic landscape was simulated for which the different update mechanisms were compared to the accurate DMC results and, (ii), the errors were evaluated with Jackknife (see chapter 1.1.4). The measured quantity was the escape current density j_{esc} which is determined from the number of charge carriers that escape over a certain time. The number of layers N_3 parallel to the contact were chosen to give a return probability below 10^{-6} (see chapter 2.2.2) with at least 9 layers (including the escape layer but excluding the contact). Unfortunately, a bug was found in the code for the calculation of the layer convergence, so the value of 10^{-6} is not always strictly right. However, the magnitude is correct, the return probability is sometimes found to be slightly higher than 10^{-6} and sometimes it is significantly lower than that. For the new update mechanism, all hopping times for all charge carriers are recalculated after each Markov jump, because here we are investigating the effect of only partially updating the injection processes. The methodological error due to only partially updating the hopping processes was already investigated in the last chapter. Parameters that were the same for all simulations are found in tab. 3.2.

3.2.1 The Three Regimes

Depending on the different values for the zero-field-energy barrier Δ and the externally applied electric field strength F, we were able to identify three injection regimes. Those three regimes result in differently shaped, spatially resolved charge carrier densities $n_{cc}^{cell}(i)$. All of them pose specific challenges for the update mechanisms. To understand the behaviour of our system, we have to take a look at the energetic landscape of the three regimes (see fig. 3.8).

Table 3.2: Parameters used to compare the new update mechanism and FRM to the accurate DMC for the injection simulations.

ν_0	$2.25 \cdot 10^{13}$ s^{-1}
ε_r	4.0
T	300 K
$N_1 \times N_2$	51×51
l	1.0 nm
α	$\frac{5}{l} = 5 \cdot 10^9$ m^{-1}

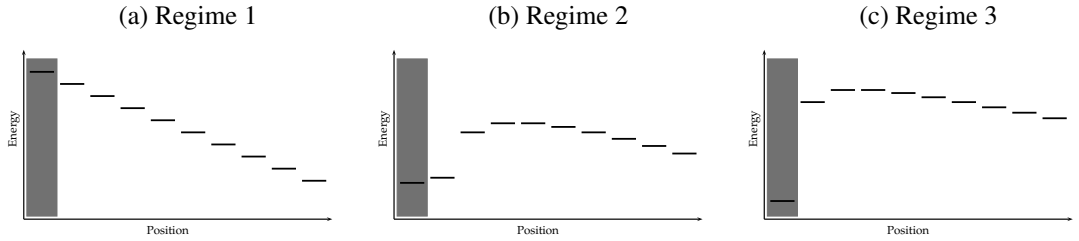

Figure 3.8: The average energy level for each layer parallel to the contact is shown for three different injection regimes. The grey region is the contact and the energy level in the grey region indicates the Fermi energy E_F. An externally applied electric field F bends the energies down for increasing distance from the contact while the image charge potential lowers the energy levels near the contact. The left picture shows regime 1 where a strong electric field F dominates the energetic landscape and the energy levels nearly drop linear. The behaviour of this system is bulk-like. Regime 2, depicted in the middle, refers to a rather weak electric field F and a low zero-field-energy barrier Δ. As the Fermi energy is very close to the energy of the first layer or even higher than it, lots of charge carrier are injected but only a few can escape over the barrier. On the right side a low electric field F and a high zero-field-energy barrier Δ is determining the energetic landscape of regime 3. In this case hardly any charge carriers are injected and out of those few charge carriers only a very small fraction can pass the barrier. For a more detailed discussion, see text.

Regime 1

The first regime (fig. 3.8a) is characterised by a high electric field F. The term 'high' in this context is determined by the requirement that, on average, the energy levels for each layer always drop when increasing the distance to the contact. So the Fermi energy is higher than the average energy level of the first layer and the average energy level of the first layer is higher than the average energy level of the second layer and so on. This energetic landscape is very similar to that of a bulk simulation (where the layer average energy levels linearly decrease due to the external electric field) with the only difference that the contact layer can inject as many charge carriers as the bulk can digest. With this knowledge, we expect a rather homogeneous distribution of the spatially resolved charge carrier density $n_{cc}^{cell}(i)$ and, depending on the interplay between the externally applied electric field strength F and the strength of the Coulomb interaction between the charge carriers, a rather high overall charge carrier density. Exactly this behaviour can be seen in fig. 3.9.

In both cases the electric field is strong enough to lead to continuously decreasing energy levels, so we are in regime 1. For a distance d to the contact of $d = 9$ nm we are in the escape layer which is the reason for a vanishing charge carrier density in this layer $n_{cc}^{cell}(d = 9 \text{ nm}) = 0$ ccpc. In the middle of the simulation volume (around $d = 5$ nm) we see a plateau in the charge carrier density. This plateau tells us that, (i), the system size is big enough to simulate the contact region and ,(i), the bulk-like transport of the charge carriers separated far enough from the contact is already observed for a distance $d > 3$ nm. Following this plateau to the escape layer, a slight reduction of the charge carrier density close to the escape layers at $d = 8$ nm is observed. This reduction can be seen clearly in the right panel of fig. 3.9. The effect is much lower but also present in the left panel of fig. 3.9. The reason for this reduction is

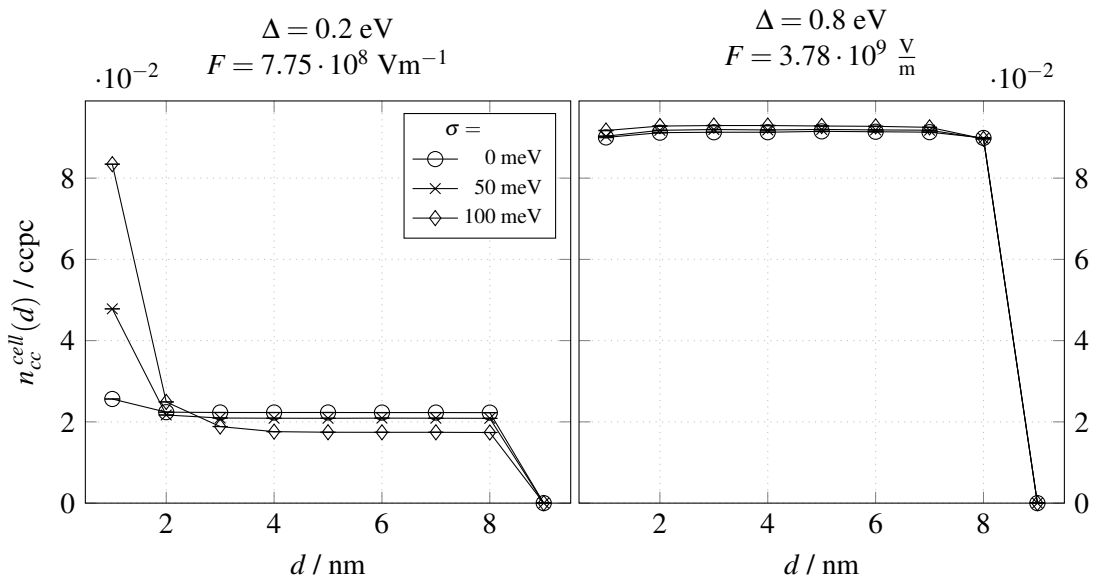

Figure 3.9: Charge carrier density of a layer $n_{cc}^{cell}(d)$ parallel to the contact in a distance d to the contact for different energetic disorders σ. The zero-field-energy barrier Δ and the externally applied electric field F is given above the plots. The simulations are performed with DMC, for which 30 million Markov jumps (+10% thermalisation) were evaluated. Only one energetic landscape was sampled and the errors were calculated with Jackknife. Both energetic landscapes are examples for the first regime governed by a strong electric field F (see text).

the fact, that the movement of the charge carriers next to the escape layer is not blocked by other charge carriers.

Near the contact ($d \leq 3$ nm), a different behaviour of the two examples in fig. 3.9 can be seen. This is due to the different total field strengths and the alignment of the Fermi energy to the layer average energies in the organic semiconductor. In the left panel of fig. 3.9, the electric field is not strong enough to digest all the charge carriers that are flooded into the bulk. This results in a high charge carrier density in the first layer compared to the other layers. For an increasing energetic disorder, the layer average of the energy levels of the occupied cell is reduced by a value in the order of the energetic disorder. So more charge carriers are injected and injection is increasingly dominating over bulk transport. Hence, the charge carrier accumulation in the first layers is increased with increasing disorder. When we look precisely at the plateaus of the charge carrier densities, we see that the lower the energetic disorder, the higher the charge carrier density at the plateau. This is expectable as a lower disorder leads to a higher bulk mobility. Hence, a higher amount of charge carriers can be guided through the bulk. The escape current density also reflects this argumentation as it is slightly higher for lower energetic disorder σ (not shown).

In contrast to this, we see a slightly decreasing charge carrier density near the contact in the right panel of fig. 3.9. This means that all injected charge carriers are efficiently transported away from the contact. So the bottleneck for this simulation is rather the injection than the bulk transport which means that the field could guide more charge carriers through the bulk than what the contact provides. Having a very precise look at the plateau, it can be seen that the charge carrier density is higher for higher energetic disorders σ, which is understandable due to the fact that disorder reduces the average occupied energy level of a layer and, hence, the injection can take place a bit easier. The bulk mobility is still higher for lower energetic disorders and as the current density is determined by the charge carrier density and the average velocity, which is proportional to the bulk mobility, those two different trends for the charge carrier density and the bulk mobility lead to a nearly equal escape current density for different energetic disorders σ. In a strict sense the bulk mobility dominates marginally and the escape current density insignificantly decreases with increasing energetic disorder (not shown).

Regime 2

The second regime is specified by a rather efficient injection followed by a barrier that has to be overcome to escape from the contact (see fig. 3.8b). In this case, a lot of charge carriers will be injected into the first layer but only a small part of those charge carriers will make it to the escape layer. If the Fermi energy is even above the average occupied energy level of the first layer, the charge carriers will move around in this layer before they decide whether to recombine or to cross the barrier. The average charge carrier density $n_{cc}^{cell}(d)$ in the layer with a distance d to the contact will be very inhomogeneous in this case.

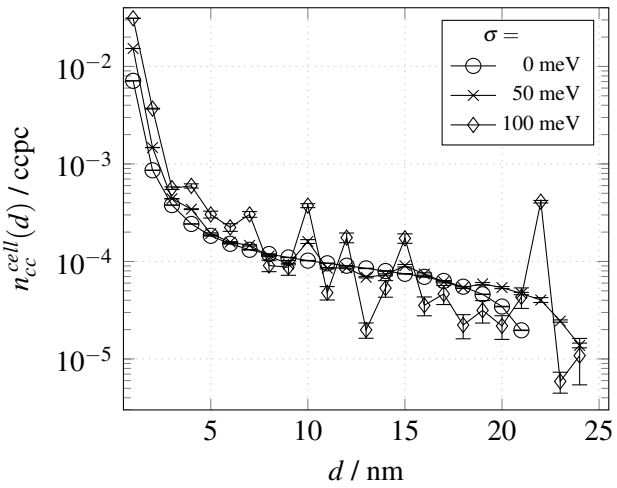

Figure 3.10: Charge carrier density of a layer $n_{cc}^{cell}(d)$ parallel to the contact in a distance d to the contact for different energetic disorders σ. The zero-field-energy barrier is $\Delta = 0.2$ eV and the externally applied electric field is $F = 7.75 \cdot 10^6$ Vm^{-1}. The simulations are performed with DMC where 100 million Markov jumps (+10% thermalisation) were evaluated. Only one energetic landscape was sampled and the errors were calculated with Jackknife. For $\sigma = 0$ meV 22 layers were simulated and for the other two cases 25 layers. The energetic landscape represents the second regime of easy injection followed by a barrier that has to be overcome to escape from the contact. Injection is getting easier with higher energetic disorder σ (see text).

The most obvious feature in fig. 3.10 is that the missing averaging over multiple energetic landscapes is causing strongly oscillating curves for the cases $\sigma = 50$ meV ($\times$) and $\sigma = 100$ meV ($\diamondsuit$). The Fermi energy is located at $E_F = -0.2$ eV and for the isoenergetic case ($\sigma = 0$ meV) the energy level of the first layer is $E_{\text{layer 1}} = -0.098$ eV due to the electric field and the own image charge potential. An energetic disorder $\sigma > 0$ meV reduces the average energy level of occupied cells by a value that approximately corresponds to the value of the energetic disorder. Thus, $E_{\text{layer 1}} - \sigma$ approximates the layer averaged occupied energy level and attains the values -0.148 eV and -0.198 eV for $\sigma = 50$ meV and $\sigma = 100$ meV, respectively. The hop upwards in energy for injections reduces from 0.102 eV for $\sigma = 0$ meV to 0.052 eV for $\sigma = 50$ meV, and to 0.002 eV for $\sigma = 100$ meV. This explains the behaviour of the curves for a low distance to the contact d where the highest disorder σ has the highest charge carrier density $n_{cc}^{cell}(d = 1)$ due to the most efficient injection. For a distance $d \approx 2$ nm to $d \approx 3$ nm the charge carrier density drops due to the barrier. As the barrier shape after the first layer is about the same for different energetic disorders, the drop in the charge carrier density observed when going from $d = 1$ nm to $d = 3$ nm has approximately the same magnitude for all three energetic disorders. At higher distances the oscillations caused by the energetic disorder prevent further interpretations. The escape current density for such low charge carrier densities in the bulk is already strongly determined by the bulk mobility whose value changes over several magnitudes with varying σ (see fig. 2.4). The latter is the reason due to which the escape current density tremendously drops for increasing energetic disorder (not shown). We additionally see that the influence of the contact is reaching to higher distances d compared to the results for regime 1 (see fig. 3.9) and the influence of the escape layers is also stronger and reaches further

into the bulk. There is not really a plateau and hence the system size is probably not large enough to guarantee that the transition from the contact region to the bulk region is considered correctly.

Regime 3

In regime 3, a very high zero-field-energy barrier Δ and a low electric field F gives rise to a large energy offset between the contact and the first layer followed by an additional barrier due to the electric field and the own image charge (see fig. 3.8c). A very low charger carrier density in the first layer can be anticipated, as nearly no charge carriers will overcome the huge injection barrier. For the other layers, an even lower charge carrier density can be expected. Additionally, oscillations due to the energetic disorder can be foreseen, as a higher charger carrier density averages out the effects of disorder to a certain amount, while very low charge carrier densities react much more sensitive to changes in the energetic landscape.

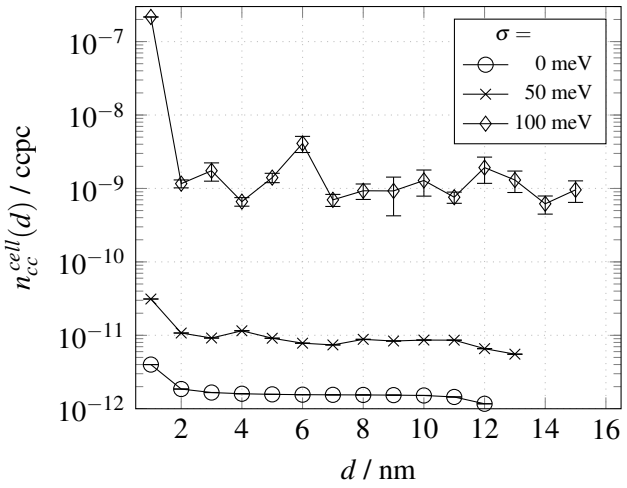

Figure 3.11: Charge carrier density for regime 3 (see fig. 3.8c) of a layer $n_{cc}^{cell}(d)$ parallel to the contact in a distance d to the contact for different energetic disorders σ. The zero-field-energy barrier is $\Delta = 0.8$ eV and the externally applied electric field is $F = 3.78 \cdot 10^7$ Vm^{-1}. The simulations are performed with DMC where 100 million Markov jumps (+10% thermalisation) were evaluated. Only one energetic landscape was sampled and the errors were calculated with Jackknife. For $\sigma = 0$ meV 13 layers were simulated, for $\sigma = 50$ meV 14 layers and for $\sigma = 100$ meV 16 layers. Hardly any charge carriers are injected and if they make it to the first layer, a barrier has to be overcome to escape from the contact (see also text).

The charge carrier density shown in fig. 3.11 reflect these expectations. The $\sigma = 100$ meV curve shows fluctuations over one magnitude for the bulk-dominated plateau and indicates a charge carrier density below $3 \cdot 10^{-7}$ ccpc in the first layer. I.e. less than 0.0008 charge carriers are on average in this layer. From the first to the second layer, the charge carrier density drops due to the barrier that has to be overcome. However, once the charge carrier made it to the third or fourth layer, it usually escapes. This argumentation is supported by the fact that a plateau can be observed for the charge carrier density in the middle layers. The influence of the escape layers is stronger than in regime 1 but much less compared to regime 2. This indicates that the system size is sufficiently large for this scenario. For higher energetic disorders, $\sigma \geq 50$ meV, the lowest energy levels of the first layer are reduced by much more than $1 \times \sigma$, it is rather in the order of $2 \times \sigma$ or even $3 \times \sigma$. Hence, energetic disorder strongly reduces the energy difference for injection. As this energy difference enters the rates exponentially, the charge carrier density in the first layer is dramatically increased, i.e., by several orders of magnitude, compared to the isoenergetic case. From such a low energy level in the first layer, the remaining barrier, that prevents the escape from the contact, is much higher compared to the isoenergetic case. This is the reason why the charge carrier density for $\sigma = 100$ meV drops by two orders of magnitude when we go from the first to the second layer and for $\sigma = 0$ meV it just decreases approximately by a factor of 2.

Nevertheless, the charge carrier density in bulk increases by approximately three orders of magnitude when we go from $\sigma = 0$ meV to $\sigma = 100$ meV. For the escape current density, the huge differences of the charge carrier densities are competing with the huge differences of the bulk mobilities for different energetic disorders. In fact, they are nearly cancelling each other. With only a factor of about 2.6 between the different escape current densities, it is increasing with increasing energetic disorder σ (not shown).

In Between the Three Regimes

With the three regimes discussed so far, we cover, in principle, all challenging scenarios for our new update mechanism. Once it can digest each of them, it can be assumed that also a superposition of those regimes can be digested. The charge carrier density for an exemplary mix of the regimes is shown in fig. 3.12.

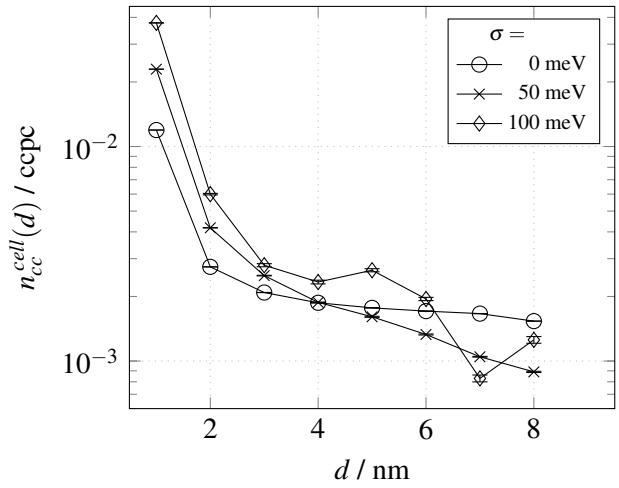

Figure 3.12: Charge carrier density of a layer $n_{cc}^{cell}(d)$ parallel to the contact in a distance d to the contact for different energetic disorders σ. The zero-field-energy barrier is $\Delta = 0.2$ eV and the externally applied electric field is $F = 7.75 \cdot 10^7$ Vm^{-1}. The simulations are performed with DMC where 50 million Markov jumps (+10% thermalisation) were evaluated. Only one energetic landscape was sampled and the errors were calculated with Jackknife. For all energetic disorders 9 layers were simulated. This scenario depicts an intermediate case of regime 1 and regime 2 discussed so far (see text and fig. 3.8).

Here, the electric field is high enough to cause decreasing energy levels with increasing distance from the contact. However, the Fermi energy is below the energy level of the first layer in the isoenergetic case. For a disorder of $\sigma = 50$ meV, the average occupied energy level in the first layer is about the same as the Fermi energy and for $\sigma = 100$ meV it is clearly below E_F. The situation in the first layers is reminiscent to regime 2 in which, (i), injection is getting easier with increasing energetic disorder and, (ii), the relative reduction in charge carrier density from the first to the second layer is approximately the same for all three energetic disorders (compare fig. 3.10). The rest of the system is behaving more like regime 1 in which many charge carriers are available near the contact but cannot be transported due to a low (and thus limiting) bulk mobility. Corresponding to the decreasing bulk mobilities for increasing energetic disorder, the charge carrier densities far away from the contact ought to decrease for increasing energetic disorder. However, this cannot be clearly seen in fig. 3.12. The formation of a bulk plateau is prevented by an insufficient size of the system (for $\sigma = 0$ meV the plateau is nearly reached). Nevertheless, a reordering of the charge carrier densities can already be notified. This is, in principle, the same reordering that can be seen in the left plot of fig. 3.9 for a distance d between 2 nm and 3 nm. As the electric field in our mixed case is much lower, the reordering occurs at much larger distances from the contact. Nevertheless, for a sufficiently large system, plateaus such as in the left plot of fig. 3.9 determined by the bulk mobility should appear. The escape current density (not shown) for this scenario is strongly decreasing for increasing energetic disorder, in accord with our argumentation. For the rather

low charge carrier densities in the order of 10^{-3} ccpc, the mobility already shows a huge dependence on the energetic disorder that also leads to a bigger difference between the charge carrier densities for the plateaus. This decreasing charge carrier density in combination with a decreasing bulk mobility leads to a strongly decreasing escape current density for increasing energetic disorder.

3.2.2 Error of the New Update Mechanism

All of the five energetic landscapes introduced above were used to test the performance of the new update mechanism and FRM compared to DMC. The chosen measure to compare the simulation results was the escape current density j_{esc} as this quantity is the one that we are interested in. The error Δj_{esc} for the method that was compared to DMC was calculated by

$$\Delta j_{esc} = \left| \frac{j_{esc,DMC} - j_{esc,method}}{j_{esc,DMC}} \right| \tag{3.2}$$

Results for the error of the escape current density plotted with respect to the injection update radius $r_{up,inj}$ can be seen in fig. 3.13 for the five scenarios that were introduced above. The two very right panels in the top and bottom row in fig. 3.13 correspond to regime 1 with charge carrier densities depicted in fig. 3.9 left and right panel respectively. The top left panel shows regime 2 (charge carrier density found in fig. 3.10), the bottom left panel represents regime 3 (charge carrier density found in fig. 3.11) and the top middle panel is an example for a superposition of regime 1 and regime 2 (charge carrier density found in fig. 3.12)).

The results for our implementation of the FRM for injection simulations are plotted at $r_{up,inj} = 0$ nm. Those simulations are orders of magnitude off the DMC results. Only for the bottom left scenario, which represents the third regime (high $\Delta = 0.8$ eV and low $F = 7.75 \cdot 10^6$ Vm^{-1}), this method works acceptable. As this regime is characterised by a very low charge carrier density, it is not really surprising that FRM works. In all other cases, the charge accumulation near the contact prevents FRM from being a proper way to simulate the scenario. When we have a very precise look at where FRM is supposed to work (regime 3, bottom left panel in fig. 3.11), we see that FRM performs worse for higher energetic disorders. For such a scenario we already found out that the charge carriers are injected to cells with the lowest energy level from which they have to overcome another high energy barrier to escape from the contact. Nearly all the current is moving over not even a handful of cells in the first layer. When a charge carrier in one of those cells decides to move over the second barrier, it takes some time to do so. But in FRM, the cell is assumed to be empty for injections. So it can happen that a new charge carrier is trying to hop to this cell when the cell is still occupied. In such a case, it will not be able to move there and the injection time for the injection cell is recalculated with this charge carrier occupying one cell of the injection region of the injection cell. This recalculated injection time is much higher than the one without the charge carrier in the vicinity of the contact and the charge carrier will be far away until the next injection is performed by this injection cell. This results in a massive blocking of the injection cell as usually the injection time would be recalculated when the charge carrier leaves the vicinity of the contact. In the simulation for $\sigma = 100$ meV, this blocking manifests impressively, as the number of charge carriers that are injected within a certain time is lowered by two orders of magnitude compared to DMC calculations (not shown). Surprisingly, the escape current density is not affected by this totally wrong injection current density. Presumably this is due to the fact that the current is mainly limited by the second energy barrier. So there is still enough charge carrier density in the first layer, even for this tremendously reduced injection current density, and the bulk can only digest a small fraction of it; this fraction is comparable for FRM and DMC. In conclusion, FRM only produces acceptable results for very low charge carrier densities and low energetic disorders, so that the formation of current channels is suppressed.

Looking at the performance of the new update mechanism once more, it shows a very fast convergence for the bulk-like regime 1 which is represented by the top right and bottom right panel in fig. 3.13. The inhomogeneous charge carrier density created by regime 2 (top left panel in fig. 3.13) also shows a fast

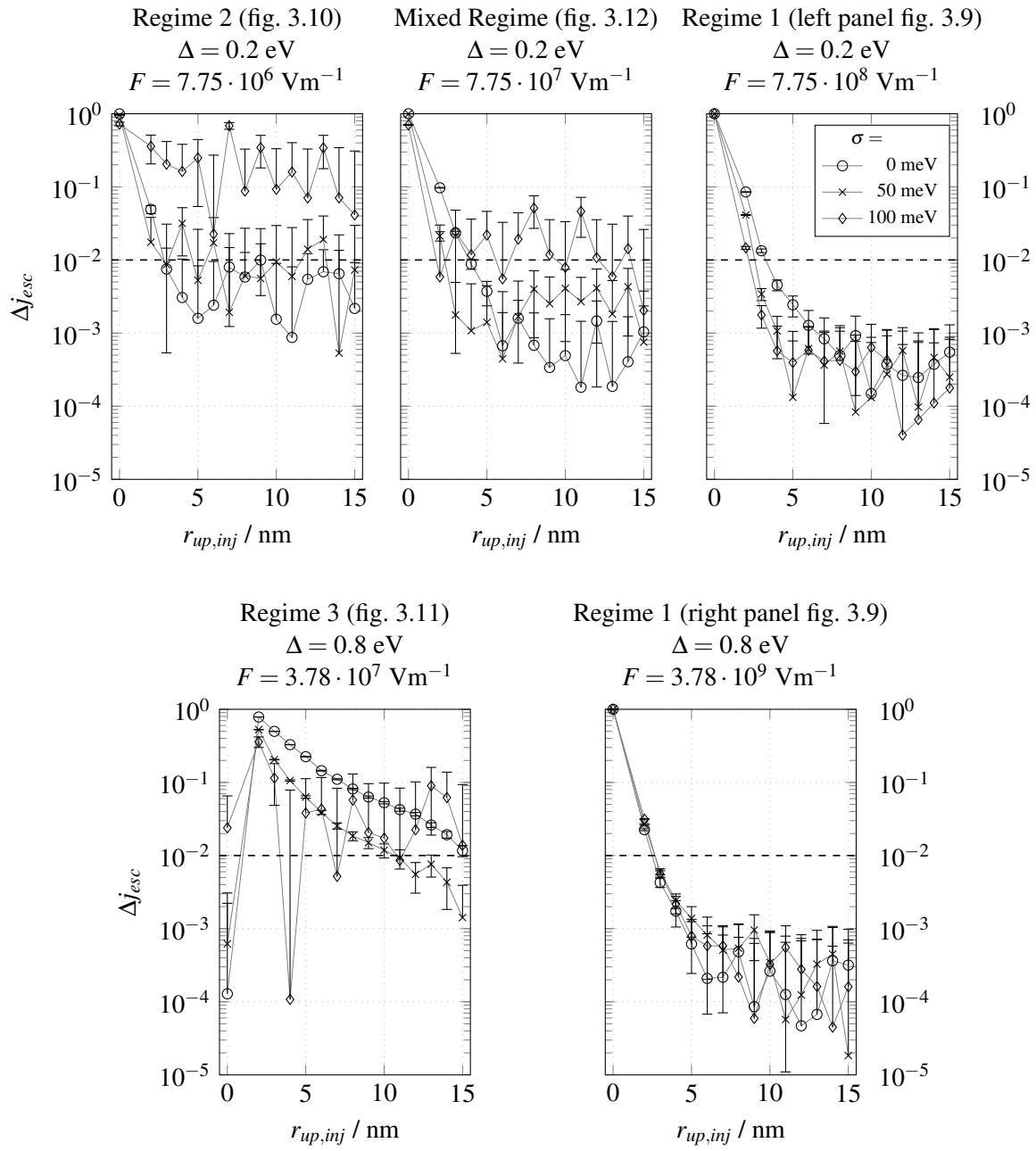

Figure 3.13: Error of the escape current density Δj_{esc} with respect to DMC achieved by FRM plotted at $r_{up,inj} = 0$ nm and the new update mechanism for injection update radii $r_{up,inj} = 2$ nm to $r_{up,inj} = 15$ nm for the five scenarios introduced above. The convergence of the new update mechanism with increasing update radius is, in general, already completed at a very low update radius $r_{up,inj}$ (discussion see text). The dashed line at an error of 1% is do guide the eye.

convergence. The measured results quickly drop below the errors of the simulation. Note that the high errors for the high energetic disorder $\sigma = 100$ meV is not due to the method but, rather, due to the bad statistics of the simulation, already for $r_{up,inj} = 3$ nm our new update mechanism and DMC are the same within their (huge) error bars. The statistics could be improved by simulating for a longer time but as this simulation is the most time-consuming of all, the knowledge gained by such a simulation is not worth the effort. Additionally when we have a look at the other graphs, the simulations for higher energetic disorders usually show better convergence as long as the data is reliable. This might be due to the fact that the change of the energy levels due to the Coulomb interactions affects the injection process less

for strongly disordered energetic landscapes. The mixed regime in the top middle picture also shows an error decreasing faster with increasing update radius $r_{up,inj}$ for higher energetic disorder. Within at least $r_{up,inj} = 4$ nm, the error of our new update mechanism dropped below 1% or it is within the accuracy of the simulation. Finally there is only one plot left, the bottom left one which shows the results for regime 3. Here the convergence is very slow, especially for low energetic disorders. The higher the energetic disorder the better the convergence, so we have to mainly consider the isoenergetic case for our convergence. As we already discussed, there are very little charge carriers in the simulation and for high energetic disorder, current channels are transporting the current through the barrier. In the isoenergetic case no current channels are present and the transport is spread all-over the bulk. A discussion of the poor convergence for this case is stated in the following.

3.2.3 The Convergence Problem of Regime 3

Our new update mechanism has the biggest problems with the isoenergetic case $\sigma = 0$ meV of regime 3 (compare fig. 3.13 bottom left plot), in which the energy barrier for the injection from the Fermi level to the first layer is much higher than the second energy barrier from the first layer to the second layer. This results in a very low charge carrier density and a rather high injection efficiency. The injection efficiency in this case can be determined by the ratio of the current density flowing through the bulk and the injection current density. The reason for the bad convergence of our new update mechanism can be understood by taking a look at the influence of the charge carriers leaving the update sphere of the injection cell. We are recalculating the energy levels of the cells involved in our injection process each time a charge carrier is hopping in the update radius of the injection cell $r_{up,inj}$. If we think of a configuration with only one charge carrier in bulk, this means that a charge carrier leaving the update sphere of the injection cell seems to be still at its previous position for the injection cell as the injection rates will no longer be updated. As a consequence, the next injection process will be wrongly blocked by an amount determined by the Coulomb interaction energy of a charge carrier sitting at a cell $r_{up,inj}$ apart from the injection cell. In general, there is not only one injection cell blocked by one charge carrier that leaves the contact region. To determine the amount of blocked injection cells, we start with no charge carriers in bulk, i.e., all injection times for all injection cells are correct. Injecting one charge carrier causes the recalculation of injection rates within an area of $r_{up,inj}^2\pi$ of the contact. On its way to the escape layers, this charge carrier leaves the update radius of an injection cell, then the one of the neighbouring injection cell, etc. So one escape from the contact region blocks an area of $r_{up,inj}^2\pi$ injection cells. Starting from a situation where all rates of all injections are calculated correctly, we build up a virtual layer of charge carriers in a distance $r_{up,inj}$ to the contact that is falsely seen by the injection cells. The competing process that reduces the charge carrier density of this virtual layer is recombination with the contact. This process leaves an area of $r_{up,inj}^2\pi$ corrected rates for the associated injection processes. Those two competing processes create a layer of virtual charge carriers in a distance $r_{up,inj}$ with a certain 'coverage' c, defined as the ratio of the number of injection processes with false rates and the number of total injection processes. If we take the number of injections N_{inj} in a long enough time interval and the number of recombinations N_{rec} in the same time interval, the coverage c should converge to $c^* = \frac{N_{inj}-N_{rec}}{N_{inj}}$. Starting from a correctly calculated injection process, the coverage would be $c = 0$ and the first escape from the contact causes a coverage of $c = \frac{r_{up,inj}^2\pi}{N_1N_2l^2}$ with the lattice constant l and the number of cells $N_1 \times N_2$ in a layer parallel to the contact. For the functional dependence of the coverage $c(i_{esc})$ on the number i_{esc} of escapes from the contact region, this implies that we require a function starting at $c(0) = 0$ with a derivative of $c'(0) = \frac{r_{up,inj}^2\pi}{N_1N_2l^2}$ and for $i_{esc} \to \infty$ we claim $c(\infty) = c^*$. The easiest function that can be interpolated between those data points is

$$c(i_{esc}) = c^* \left(1 - e^{-\frac{r_{up,inj}^2\pi}{c^*N_1N_2l^2} i_{esc}}\right) \tag{3.3}$$

If we allow more than one charge carrier in the simulation, we can expect that the charge carrier nearest to the contact influences the contact most. Assuming a charge carrier density $n_{cc,1}^{cell}$ in the first layer, the

rates are correctly recalculated in the vicinity of those charge carriers in the first layer. Each of those charge carriers influences a number of $\frac{r_{up,inj}^2\pi}{l^2}$ injection cells. This means that a fraction of $n_{cc,1}^{cell}\frac{r_{up,inj}^2\pi}{l^2}$ is calculated correctly while, for the rest, the coverage $c(i_{esc})$ holds and we get a total coverage of

$$c(i_{esc}) = \begin{cases} c_0^* \left(1 - e^{-\frac{r_{up,inj}^2\pi}{c^* N_1 N_2 l^2} i_{esc}}\right) & \text{for } c_0^* > 0 \\ 0 & \text{for } c_0^* \leq 0 \end{cases}$$
$$\text{with} \quad c_0^* = \left(1 - n_{cc,1}^{cell}\frac{r_{up,inj}^2\pi}{l^2}\right) c^* \quad \text{and} \quad c^* = \frac{N_{inj} - N_{rec}}{N_{inj}} \tag{3.4}$$

The energy shift of a virtual charge carrier can be approximated by the Coulomb interaction energy caused by a charge carrier at a distance $r_{up,inj}$ to the contact on a cell in the first layer and its virtual image charge Coulomb interaction

$$\Delta E_v = \frac{e^2}{4\pi\varepsilon_0\varepsilon_r}\left(\frac{1}{r_{up,inj} - l} - \frac{1}{r_{up,inj} + l}\right) \tag{3.5}$$

The energy difference for the correct calculation of the injection rates is determined by the Fermi energy E_F, the electric field strength F and the own image charge (compare (1.109)). This method is optimised for the isoenergetic case, nevertheless energetic disorder σ can be approximately considered by shifting down the energy levels in bulk by σ.

$$\Delta E_{corr} = E_{\text{layer 1}} - E_F = \Delta - qlF - \frac{e^2}{16\pi\varepsilon_0\varepsilon_r l} - \sigma \tag{3.6}$$

This correct energy difference is increased by ΔE_v only for the cells that are blocked. The blocked cells are a fraction given by the coverage c, so we shift the energy levels, on average, only by $c\Delta E_v$. For high energy barriers, the injection process is determining the properties of the simulation. So the error of the injection rate is a good approximation for the total error ERR_{tot} of the simulation. This error is obtained when we calculate the rate R_{corr} for the correct injection with energy difference ΔE_{corr} and the rate R_v for the case when the virtual charge carriers are blocking the injection with energy difference $\Delta E_{corr} + c\Delta E_v$

$$ERR_{tot} = \left|\frac{R_{corr} - R_v}{R_{corr}}\right| \tag{3.7}$$

The rates can be calculated by Miller-Abrahams rate (1.100) or Marcus rate (1.99), here we used Miller-Abrahams rates.

If the argumentation above is correct, we should be able to compare the error calculated by our assumptions to the ones received from the calculations (see fig. 3.13 bottom left plot). This comparison is visualised in fig. 3.14.

The estimator of the error was calculated by evaluating the coverage (3.4) with the measured injection and escape current densities and the charge carrier density in the first layer of the corresponding simulation. With this measured coverage, the rates and the error were calculated. We see that, for the isoenergetic case, the estimator fits very well to the simulated error except for very high update radii, i.e., when the update radius is in the range of the system size and the layer of virtual charge carriers is no longer in our system. So the reduction of the error is a boundary effect and does not represent the actual convergence of our new update mechanism to DMC results. For an energetic disorder of $\sigma = 50$ meV the estimated error slightly undervalues the simulated one for low update radii (always less than a factor of 2) but the trend is represented very well.

This gives us the confidence to improve our simulations by decreasing the coverage via updating all injection processes at certain Markov times. To minimise the chance of biasing the simulation, the Markov time elapsed between two total injection updates is kept constant. Note that we would strongly

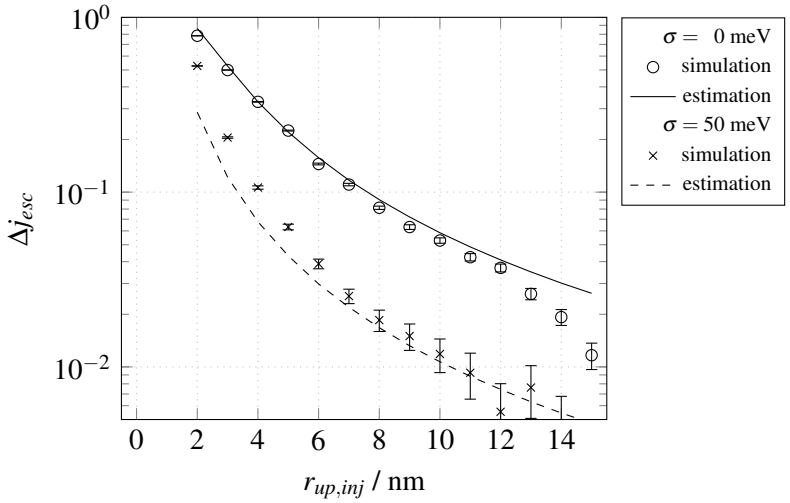

Figure 3.14: Error of the escape current density j_{esc} for the new update mechanism over the update radius $r_{up,inj}$ for a zero-field-energy barrier of $\Delta = 0.8$ eV and an electric field strength of $F = 3.78 \cdot 10^7$ Vm^{-1}. The error estimated with (3.4) to (3.7) is depicted with lines, the data points from our simulations are taken from fig. 3.13 bottom left panel (detail see text).

bias the results if we would update after a fixed number of Markov jumps. As injection in this regime takes much more time than bulk hopping, there would be, on average, to many charge carriers in the simulation during the total injection updates and hence the injection process would be blocked to much. The implementation of updating after a certain Markov time is more complex than updating after a certain number of Markov jumps. We have solved this by using a variable `time_next_update` which holds the Markov time when the next update has to be performed. Ahead of each Markov jump this variable is compared with the current Markov time t_i evolved by the retention time Δt_i of the process that would be performed next. If `time_next_update` is lower than $t_i + \Delta t_i$, then all rates of all injection processes are recalculated, the hopping times of the charge carriers are reduced by `time_next_update` $- t_i$, and the current Markov time t_i is set to `time_next_update`. This approach is understandable as recalculating at a certain time for exponentially distributed random numbers also implies advancing the Markov time to this point, although nothing happened in the time till the update. When the update is performed, the time when the next update happens is evolved by the time difference between two total injection updates. Special attention has to be drawn to the case when the current Markov time is larger than the time at which the next update would be performed, which can happen if the last total injection update was in the last Markov jump and the lowest retention time of the updated processes was larger than the time difference between two total updates. In this case, the processes were up to date anyway, so we do not have to update the injection processes, but the variable `time_next_update` has to be increased by the time difference between two total updates until it is larger than the current Markov time. With this corrected `time_next_update` we can have a look whether an update is required before the next process as described above. To evaluate the constant time difference between two injection updates, we fix a value for the desired accuracy of our simulation (e.g. 1%) and calculate the coverage c_{acc} that is needed to get this accuracy by solving the implicit equation (3.7) for the coverage numerically. To get estimators for the long time convergence c^* and the charge carrier density in the first layer $n_{cc,1}^{cell}$ we start a simulation without total injection updates by performing a number of $N_{M,therm} = 0.1 \cdot N_M$ Markov jumps (N_M is the desired number of total Markov jumps for the actual simulation) for thermalisation and afterwards another $N_{M,cov} = 0.1 \cdot N_M$ Markov jumps for measuring purposes. After thermalisation we count the number of injected charge carriers N_{inj} and recombined charge carriers N_{rec} to get c^* and measure the charge carrier density in the first layer. Now we know our desired coverage c_{acc} and all quantities in (3.4) except the number of escapes i_{esc} that are needed for this accuracy, which is, in principle, the quantity

that we want to know.

$$i_{esc} = -\frac{c^* N_1 N_2 l^2}{r_{up,inj}^2 \pi} \log\left(1 - \frac{c_{acc}}{\left(1 - n_{cc,1}^{cell} \frac{r_{up,inj}^2 \pi}{l^2}\right) c^*}\right) \quad \text{with} \quad c^* = \frac{N_{inj} - N_{rec}}{N_{inj}} \tag{3.8}$$

Note that if the desired coverage c_{acc} is already higher than the overall long time coverage c_0^*, no update is necessary. This case can already be eliminated at the very beginning of this calculation (before the implicit equation (3.7) is solved numerically) when the error of the method (3.7) is calculated with the overall long time coverage c_0^* of the simulation (see (3.4)) and this error is already below the desired accuracy.

To get the Markov time between two total injection updates $\Delta t_{\text{inj update}}$ we have to multiply the number of escapes i_{esc} (which is a non integer value) with the average Markov time between two escapes

$$\Delta t_{\text{inj update}} = i_{esc} \frac{\Delta t_{cov}}{N_{esc} - N_{rec}} \tag{3.9}$$

where Δt_{cov} is the elapsed Markov time during the $N_{M,cov}$ Markov jumps after thermalisation.

This implementation can be used for all regimes, as there will be no updates necessary for regime 1 and regime 2 due to the high charge carrier densities in the first layer. The performance of our final program can be seen in fig. 3.15 where the method error is plotted over the zero-field-energy barrier Δ for a selected set of update mechanisms compared to DMC results. Energetic disorders of $\sigma = 0$ meV and $\sigma = 50$ meV were simulated with electric field strength $F = 3.78 \cdot 10^7$ Vm^{-1} and a system size of $51 \times 51 \times 14$ cells.

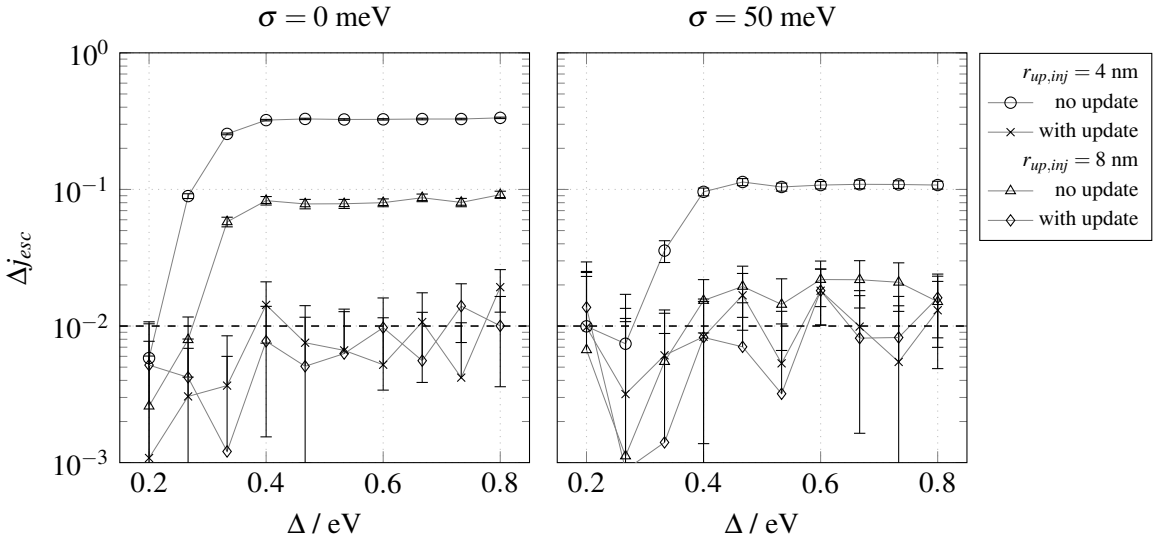

Figure 3.15: Error of the escape current density j_{esc} for the new update mechanism over the zero-field-energy barrier Δ for different update mechanisms compared to DMC results. The electric field strength is $F = 3.78 \cdot 10^7$ Vm^{-1} and the energetic disorder is written above the plots. A number of 10 million Markov jumps (+10% for thermalisation and eventually +10% for coverage evaluations) were simulated. The improved update mechanism, which is updating all injection processes after a certain Markov time, yields to an error of around 1%, which is exactly the chosen error for the method (detail see text). The dashed line at an error of 1% is do guide the eye.

The two plots in fig. 3.15 quite impressively confirm the predictions for the error of the escape current density. The range of simulated zero-field-energy barriers Δ was chosen to include the transition from regime 3 to regime 2. For $\Delta = 0.8$ eV hardly any charge carriers are injected and for $\Delta = 0.2$ eV the

charge carrier density in the first layer is around 0.01 ccpc. In the isoenergetic case, $\sigma = 0$ meV, depicted on the left side, the old method without a total injection update leads to method errors above 30% for an injection update radius of $r_{up,inj} = 4$ nm and still errors of about 10% for $r_{up,inj} = 8$ nm. With decreasing Δ this method error is changing when the transition from regime 3 to regime 2 is happening below $\Delta = 0.4$ eV and for $\Delta = 0.2$ eV all errors are below 1%. If we now take a look at the impressive results of our improved method where the injection rates are recalculated after a certain Markov time, we see that the errors of this method are 1% within their statistical fluctuation, which is exactly what we predicted. The same holds for an energetic disorder of $\sigma = 50$ meV where the improved method again produces errors around 1% whereas, in contrast, not performing total injection updates for $r_{up,inj} = 4$ nm leads to 10% and for $r_{up,inj} = 8$ nm to 2%. Another remarkable point is that the error of 1% is not depending on the chosen update radius in regime 3.

If we now have a look at tab. 3.3, where the number of total injection updates for our simulation with 10 million Markov jumps is shown, we see that the lower update radius $r_{up,inj} = 4$ nm does not really need more total injection updates than the higher update radius $r_{up,inj} = 8$ nm to reach this accuracy. Only for low zero-field-energy barriers Δ the number of recalculations decreases more slowly for the lower update radius.

Table 3.3: Number of total injection updates needed to achieve the accuracy of 1% for the simulations described above with $N_M = 10$ million. The simulation differs a bit from the introduced implementation, so the numbers in this table are slightly higher than what they really are.

	$\sigma = 0$ meV		$\sigma = 50$ meV	
Δ / eV	$r_{up,inj} = 4$ nm	$r_{up,inj} = 8$ nm	$r_{up,inj} = 4$ nm	$r_{up,inj} = 8$ nm
2.00E-001	54475	0	0	0
2.67E-001	178039	109192	0	0
3.33E-001	212208	246704	113486	0
4.00E-001	198396	247246	161823	115438
4.67E-001	195676	241321	169915	130828
5.33E-001	197822	243082	171172	128752
6.00E-001	194125	240744	174560	135149
6.67E-001	193175	238958	176286	135007
7.33E-001	192392	240046	176259	131443
8.00E-001	195962	241874	170653	136848

As a reminder, an injection update radius $r_{up,inj} = 4$ nm is enough to produce errors below 1% for the simulation of regime 1 and regime 2 (compare fig. 3.13). With our improved method we can achieve exactly the errors that we want for regime 3 by updating all injection processes after a certain Markov time practically independent of the injection update radius $r_{up,inj}$ (as long as it is ≥ 4 nm). The number of total injection updates is not influenced much by $r_{up,inj}$ as well. So it is definitely favourable to take $r_{up,inj} = 4$ nm instead of $r_{up,inj} = 8$ nm as it produces approximately the same error and the runtime is 2-4 times lower for $r_{up,inj} = 4$ nm.

4 Conclusion and Outlook

During this master thesis we developed an improved update mechanism for kinetic Monte Carlo simulations and testing its performance. The challenge was to reduce the computational effort compared to an accurate update mechanism (where all rates of all ongoing processes are updated after every Markov jump), but keep the methodological error as low as possible, especially for very high charge carrier densities, where the well established first reaction method (a computationally very economical alternative update mechanism) fails.

We started with the development of an improved update mechanism for the simplest type of kinetic Monte Carlo simulations, where only one type of charge carriers can hop through a homogeneous bulk material. This material consists of cubic cells with a cell size of 1 nm^3 where one charge carrier can occupy one cell. The total energy of the system is composed of the energetic disorder of our organic semiconductor, an external electric field, and the Coulomb interaction between the charge carriers. In this case our new update mechanism updates all hopping processes within a certain update radius around the currently hopped charge carrier. The measured mobility for our improved update mechanism rapidly converges to the accurate results with increasing update radius, especially for high charge carrier densities. The two major effects that lead to this convergence are blocking (see chapter 3.1.1) and detrapping (see chapter 3.1.2), both induced by the Coulomb interaction between the charge carriers. The methodological error of our new update mechanism for the bulk mobility was below 1% for an update radius $r_{up,cc} \geq 8$ nm. For our system size of $51 \times 51 \times 51$ nm^3 the volume of the update sphere is approximately 60 times smaller than the total volume, which leads to a reduction of the computational effort of a factor of 60 for a clever implementation.

The extension of our new update mechanism to a more complex system was shown in a next step, where injection simulations were performed. Here a layer of contact cells is put on one side of the organic semiconductor where charge carriers can be injected by the injection cells or recombine with them. This means that a charge carrier can be created or annihilated by the injection cells. Furthermore the charge carriers are taken out of the simulation as soon as they reach a certain distance to the contact cells where it can be assumed that they are no longer influencing the contact region. The hopping process of charge carriers in the organic semiconductor is now supplemented by injection processes which also need to be updated. Additionally to the energy contributions mentioned for the bulk simulations, image charge interactions induced by the metal contact have to be considered. Introducing an update radius for injections $r_{up,inj}$ again leads to a low methodological error already for low update radii in most cases. The observable that was measured for injection scenarios was the current density of the charge carriers that were leaving the contact region. Unfortunately only increasing the update radius was not enough to ensure a proper convergence of our new update mechanism for all possible regimes of an injection simulation. The most challenging regime for our new update mechanism is surprisingly the one with very low charge carrier densities. For this regime we need to update all injection processes after a fixed Markov time, in addition to updating a few injection processes within the update radius for injections $r_{up,inj}$ after each Markov jump. The methodological error is now determined by the update radius for bulk hopping $r_{up,cc}$, the update radius for injections $r_{up,inj}$, and the Markov time after which all injection processes are updated. Choosing the update radii $r_{up,cc} = 8$ nm and $r_{up,inj} = 4$ nm in combination with updating all injection processes after a certain Markov time evaluated with the method described in chapter 3.2.3 should guarantee a methodological error below 1% for the current density of the charge carriers leaving the contact region, independent of the simulated regime.

It is expected that our new update mechanism reduces the computational effort of any kinetic Monte Carlo simulation where Coulomb interactions are present while producing a low methodological error. By assigning an update radius to each type of process and eventually updating all processes of a certain type after a fixed Markov time (not a fixed amount of Markov jumps), more complex processes like hopping of multiple types of charge carriers or exiton creation and annihilation are assumed to be digestible as well.

Due to the strongly reduced simulation time of our new update method, it is very suitable e.g. for

overview-simulations, in which a wide range of parameters is scanned. This may enable the identification of the position of a certain phase transition; the details of this transition might then be simulated with the accurate update mechanism. As we are cutting off the exact interactions at a certain distance and get some kind of mean field approximation beyond this cut-off, especially the long ranging correlations at a phase transition can be expected to produce huge methodological errors and the results will be of the grade of a mean field approximation.

In a usual kinetic Monte Carlo simulation we need to average over multiple energetic landscapes. If we are not sure if our new update mechanism is producing correct results, it is recommendable to use one of those energetic landscapes to compare the results of our new update mechanism and the accurate one. With this the methodological error can be estimated but we only need to simulate the computationally expensive accurate method once. For the other energetic landscapes our new update mechanism can be used and overall we can strongly reduce the computational effort.

5 Abbreviations and Formula Symbols

Abbreviations

ccpc	charge carriers per cell
DD	Drift Diffusion
DFT	Density Functional Theory
DMC	Dynamic Monte Carlo
FRM	First Reaction Method
HOMO	Highest Occupied Molecular Orbital
KMC	Kinetic Monte Carlo
LUMO	Lowest Unoccupied Molecular Orbital

Formula Symbols

α	charge delocalisation constant
$\beta = \frac{1}{k_B T}$	inverse temperature
d	minimum distance to the metal contact
$\delta_{x,x'}$	Kronecker delta
Δ	zero-field-energy barrier
ΔE	energy difference
Δj_{esc}	error of the escape current density of a certain update mechanism compared to DMC results
$\Delta\mu$	error of the mobility of a certain update mechanism compared to DMC results
$\Delta\vec{s}$	distance overcome by a charge carrier
Δt	retention time (Markov time for which the Markov chain stays in its current state)
e	elemental charge
E	energy
E_F	Fermi energy of the metal
E_r	reorganisation energy
$\mathrm{erf}(x)$	error function
$\mathrm{erfc}(x)$	complementary error function
η	splitting parameter for the Ewald method
ε_0	vacuum permittivity
ε_i	randomly chosen energy level of cell i
ε_r	relative permittivity of the medium
F	electric field strength
$\vec{G}$	reciprocal lattice vector of a periodic system
$\hat{H}$	Hamilton operator
i,j,k,l,m,n	indices for summation and labelling
$\vec{j}$	current density
j_A	current density through a specific area A
j_{esc}	escape current density
k_B	Boltzmann factor
l	lattice constant
μ	mobility
n_{cc}	charge carrier density
n_{cc}^{cell}	charge carrier density in charge carriers per cell (ccpc)
N	integer number of a specific quantity
$N_1 \times N_2 \times N_3$	number of cells in the simulated system in x_1, x_2 and x_3 direction, contact layer is excluded, escape layers are included in N_3
N_{ac}	number of autocorrelation times
N_{cells}	number of cells $N_{cells} = N_1 \times N_2 \times N_3$

Formula Symbols (continued)

N_{cc}	number of charge carriers in the simulation
N_{esc}	number of escape events
N_{hop}	number of neighbours where hopping is allowed
N_{inj}	number of injection events
N_M	number of Markov jumps in the main simulation (after thermalisation)
N_{rec}	number of recombination events
ν_0	hopping prefactor
$p(A\|B)$	probability density of A given B
$P(A\|B)$	probability of A given B
$p_n(x,x')$ / $p_t(x,x')$	probability density that the Markov chain performes a transition from state x to x' within a discrete Markov time n / continuous Markov time t
$\pi = (\pi_x)$	probability distribution row vector
π_x	entry corresponding to state x in probability distribution vector π
ϕ	injection efficiency
φ	potential field
$q = \pm e$	charge of a charge carrier
$Q = (q_{x,x'})$	q-matrix of the continuous time Markov chain
$\vec{r}$	real space position vector
r_c	Coulomb cut-off radius
$r_{hop,cc}$	hopping radius for bulk hopping
$r_{hop,inj}$	hopping radius for injection
r_{tc}	thermal capture radius
r_{up}	update radius of the new update mechanism
$r_{up,cc}$	update radius for bulk hopping
$r_{up,inj}$	update radius for injection cells
R	rate or rate parameter of an exponentially distributed random number
$R^{(n)} = \sum_{i=1}^{n} R_i$	summarised rate of n individual rates R_i
$\rho_A(t)$	autocorrelation function for an observable A
$\rho_A^E(t)$	empirical autocorrelation function for an observable A
σ	energetic disorder of the Gaussian distributed energy levels of the cells
t	Markov time
t_0	Markov time after thermalisation
T	temperature
$\mathbf{T} = (\mathbf{T}_{x,x'})$	transition matrix / transition function of the Markov chain
$\vec{T}$	real space translation vector of a periodic system
$\tau_i(A)$	autocorrelation time with index i for an observable A
$\tau_{exp}(A)$	asymptotic autocorrelation time for an observable A
$\tau_{int}(A)$	integrated autocorrelation time for an observable A
$\Theta(t)$	Heaviside step-function
$\vec{v}$	velocity
V	volume of the system
V_{cell}	volume of a cell
$x \in \mathcal{X}$	state x out of state space $\mathcal{X}$ of the system
$\mathcal{X}$	state space of the system
ξ	uniformly distributed random number in the interval $(0,1]$ or $(0,1)$
x_1, x_2, x_3	real space Cartesian coordinate into the corresponding direction
Z	partition function

6 Bibliography

References

[1] V. Coropceanu, J. Cornil, D. A. da Silva Filho, Y. Olivier, R. Silbey, and J.-L. Brédas: CHARGE TRANSPORT IN ORGANIC SEMICONDUCTORS, Chem. Rev. **107**, 926-952 (2007)

[2] Ioffe Physico-Technical Institute: ELECTRONIC ARCHIVE NEW SEMICONDUCTOR MATERIALS - CHARACTERISTICS AND PROPERTIES [`http://www.ioffe.ru/SVA/NSM/` accessed on February 3rd, 2016]

[3] M. V. Jacob: ORGANIC SEMICONDUCTORS: PAST, PRESENT AND FUTURE, Electronics **3**, 594-597 (2014)

[4] T. Minari, P. Darmawan, C. Liu, Y. Li, Y. Xu, and K. Tsukagoshi: HIGHLY ENHANCED CHARGE INJECTION IN THIENOACENE-BASED ORGANIC FIELD-EFFECT TRANSISTORS WITH CHEMICALLY DOPED CONTACT, Appl. Phys. Lett. **100**, 093303 (2012)

[5] Y. Olivier, V. Lemaur, J. L. Brédas, and J. Cornil: CHARGE HOPPING IN ORGANIC SEMICONDUCTORS: INFLUENCE OF MOLECULAR PARAMETERS ON MACROSCOPIC MOBILITIES IN MODEL ONE-DIMENSIONAL STACKS, J. Phys. Chem. **110**, 6356-6364 (2006)

[6] B. Lüssem, M. L. Tietze, H. Kleemann, C. Hoßbach, J. W. Bartha, A. Zakhidov, and K. Leo: DOPED ORGANIC TRANSISTORS OPERATING IN THE INVERSION AND DEPLETION REGIME, Nat. Commun. **4**, 2775 (2013)

[7] S. K. Possanner, K. Zojer, P. Pacher, E. Zojer, and F. Schürrer: THRESHOLD VOLTAGE SHIFTS IN ORGANIC THIN-FILM TRANSISTORS DUE TO SELF-ASSEMBLED MONOLAYERS AT THE DIELECTRIC SURFACE, Adv. Funct. Mater. **19**, 958-967 (2009)

[8] D. P. Kroese, T. Taimre, and Z. I. Botev: HANDBOOK OF MONTE CARLO METHODS, Whiley series in probability and statistics, United States of America (2011)

[9] S. P. Meyn, and R. L. Tweedie: MARKOV CHAINS AND STOCHASTIC STABILITY, Springer Verlag (1993)

[10] W. J. Anderson: CONTINUOUS-TIME MARKOV CHAINS: AN APPLICATIONS-ORIENTED APPROACH, Springer Series in Statistics (1991)

[11] B. A. Berg: INTRODUCTION TO MARKOV CHAIN MONTE CARLO SIMULATIONS AND THEIR STATISTICAL ANALYSIS, arXiv:cond-mat/0410490v1 (2004)

[12] H. M. Cuppen, L. J. Karssemeijer, and T. Lamberts: THE KINETIC MONTE CARLO METHOD AS A WAY TO SOLVE THE MASTER EQUATION FOR INTERSTELLAR GRAIN CHEMISTRY, Chem. Rev. **113**, 8840-8871 (2013)

[13] K. A. Fichthorn, and W. H. Weinberg: THEORETICAL FOUNDATIONS OF DYNAMICAL MONTE CARLO SIMULATIONS, J. Chem. Phys. **95**, 1090 (1991)

[14] K. Jacobs: STOCHASTIC PROCESSES FOR PHYSICISTS, Cambridge University Press, United States of America (2010)

[15] H. G. Evertz: COMPUTER SIMULATIONS, lecture notes, Technical University of Graz (2009)

[16] R. A. Marsh, C. Groves, and N. C. Greenham: A MICROSCOPIC MODEL FOR THE BEHAVIOR OF NANOSTRUCTURED ORGANIC PHOTOVOLTAIC DEVICES, J. Appl. Phys. **101**, 083509 (2007)

[17] A. B. Walker: MULTISCALE MODELING OF CHARGE AND ENERGY TRANSPORT IN ORGANIC LIGHT-EMITTING DIODES AND PHOTOVOLTAICS, Proceedings of the IEEE **97 (9)**, 1587-1596 (2009)

[18] C. Groves, R. G. E. Kimber, and A. B. Walker: SIMULATION OF LOSS MECHANISMS IN ORGANIC SOLAR CELLS: A DESCRIPTION OF THE MESOSCOPIC MONTE CARLO TECHNIQUE AND AN EVALUATION OF THE FIRST REACTION METHOD, J. Chem. Phys. **133**, 144110 (2010)

[19] C. Groves: DEVELOPING UNDERSTANDING OF ORGANIC PHOTOVOLTAIC DEVICES: KINETIC MONTE CARLO MODELS OF GEMINATE AND NON-GEMINATE RECOMBINATION, CHARGE TRANSPORT AND CHARGE EXTRACTION, Energy Environ. Sci. **6**, 3202-3217 (2013)

[20] J. M. Ziman: MODELS OF DISORDER: THE THEORETICAL PHYSICS OF HOMOGENOUSLY DISORDERED SYSTEMS, Cambridge University Press, Cambridge (1979)

[21] R. A. Marcus: ELECTRON TRANSFER REACTIONS IN CHEMISTRY. THEORY AND EXPERIMENT, Rev. Mod. Phys. **65 (3)**, 599-610 (1993)

[22] A. Miller, and E. Abrahams: IMPURITY CONDUCTION AT LOW CONCENTRATIONS, Phys. Rev. **120 (3)**, 745-755 (1960)

[23] M. M. Taddei, T. N. C. Mendes, and C. Farina: SUBTLETIES IN ENERGY CALCULATIONS IN THE IMAGE METHOD, Eur. J. Phys. **30**, 965-972 (2009)

[24] D. T. Gillespie: A GENERAL METHOD FOR NUMERICALLY SIMULATING THE STOCHASTIC TIME EVOLUTION OF COUPLED CHEMICAL REACTIONS, J. Comp. Phys. **22**, 403-434 (1976)

[25] M. C. Heiber, and A. Dhinojwala: DYNAMIC MONTE CARLO MODELING OF EXCITON DISSOCIATION IN ORGANIC DONOR-ACCEPTOR SOLAR CELLS, J. Chem. Phys. **137**, 014903 (2012)

[26] P. Breitegger: KINETIC MONTE CARLO SOLVER FOR THE CHARGE TRANSPORT IN DISORDERED SOLIDS, Master Thesis, Graz University of Technology (2015)

[27] U. Wolf, V. I. Arkhipov, and H. Bässler: CURRENT INJECTION FROM A METAL TO A DISORDERED HOPPING SYSTEM. I. MONTE CARLO SIMULATION, Phys. Rev. B **59**, 7507 (1999)

[28] J. Zhou, Y. C. Zhou, J. M. Zhao, C. Q. Wu, X. M. Ding, and X. Y. Hou: CARRIER DENSITY DEPENDENCE OF MOBILITY IN ORGANIC SOLIDS: A MONTE CARLO SIMULATION, Phys. Rev. B **75**, 153201 (2007)

[29] J. J. M. van der Holst, F. W. A. van Oost, R. Coehoorn, and P. A. Bobbert: MONTE CARLO STUDY OF CHARGE TRANSPORT IN ORGANIC SANDWICH-TYPE SINGLE-CARRIER DEVICES: EFFECTS OF COULOMB INTERACTIONS, Phys. Rev. B **83**, 085206 (2011)

[30] P. P. Ewald: DIE BERECHNUNG OPTISCHER UND ELEKTROSTATISCHER GITTERPOTENTIALE, Ann. Phys. (Leibzig) **64**, 253-287 (1921)

[31] H. Lee, W. Cai: EWALD SUMMATION FOR COULOMB INTERACTIONS IN A PERIODIC SUPERCELL, Stanford University (2009) [`http://micro.stanford.edu/mediawiki/images/4/46/Ewald_notes.pdf` accessed on April 19th, 2016]

[32] H. Sormann: NUMERISCHE METHODEN IN DER PHYSIK, Vorlesungsskript, Technische Universität Graz (Wintersemester 2009/2010)

7 Appendix

7.1 Program Documentation

The described program simulates charge transport in organic semiconductors on a microscopic scale. The method that is used to perform the simulation is kinetic Monte Carlo. The code was initially developed during a stay at Durham University with the kind help of Chris Groves and further improved at the Technical University of Graz. `Fortran` was chosen as a programming language using the standard of 2003 compiled with `gfortran` and `mpif90`. Parallelisation was implemented using openmpi. Most programs consist of 10 files, the two files `morphology_modul.f90` and `mc_modul.f90` are the most important ones where all the structures and functions for building up a simulation are found. In the following chapters the content of the files is described and in chapter 7.1.11 structograms of commonly used programs can be found.

7.1.1 `Makefile`

The `Makefile` is used for compiling the program to get an executable file. The used compiler as well as the used files and their control hierarchy can be found in it. To create an executable file, `make` has to be typed into the terminal and the `Makefile` combines all the necessary modules and functions.

7.1.2 `start.f90`

In this file the starting point of a `Fortran` program, the `program`-routine, is found. Additionally the task-ID of the simulated process is evaluated for parallel computing. If there are different simulations available, the user can decide which simulation should be executed by choosing the corresponding exercise number. This can be done either by typing e.g. `./KMC -exnr 1` into the terminal to run the first exercise of the program `KMC`, or calling the program, which should be executed, without additional information `./KMC` and then following the instructions of the program to choose the right exercise number. For parallelised simulations, the program is started, e.g. with 4 cores which are all executing example 1, with the following terminal command: `mpirun -np 4 KMC -exnr 1`

7.1.3 `test.f90`

This subroutine is used for testing parts of the program and having a detailed look at the performance of certain subroutines and functions. The tests are executed if the exercise number 0 is chosen at the beginning of the program.

7.1.4 `run_sim.f90`

Here the information about the simulation is found. The parameters like the lattice size or the hopping range are chosen and the subroutines to do the simulation are called. Some typical examples of how this subroutine can be structured are found in chapter 7.1.11. The name of this file can vary, e.g. the simulation of the bulk mobility for a single charge carrier that was compared with [16] was called `rep_marsh07.f90` where `rep` is short for reproduce. To execute this part of the program, exercise number 1 (or if more than one such files are available a higher number) has to be picked.

Additionally to the parameters, a function `randE` has to be provided, which assigns the Gaussian distributed energy levels for the bulk, the Fermi-energy for the injection cells and so on. The chosen random distribution has to be conform with the morphology assigned by `get_morphology` (see page 90). This means the function gets an integer number as input that decides which random distribution should be taken for the energy level and the output of the function is the randomly chosen energy level.

For parallelisation the corresponding MPI-routines are used to collect or distribute certain variables. For details about parallelisation I have to refer to the homepage of Open MPI [`https://www.open-mpi.org/` accessed on May 11th, 2016].

7.1.5 correlations.f90

In this module the subroutines for calculating autocorrelation functions and autocorrelation times are found. The implementation is described in detail in [15].

7.1.6 numeric_lib.f90

This is a library with useful numerical calculations. If not specified in the file, the algorithms are taken out of [32] with minor adaptions.

7.1.7 plot_lib.f90

Most of the visualisations are done with `gnuplot`. The file `plot_lib.f90` contains visualisation-routines especially for 2D plots.

7.1.8 storage_lib.f90

The file `storage_lib.f90` contains standard subroutines for writing, appending and reading data to and from files.

7.1.9 morphology_module.f90

One of the two core modules used for the simulation. This module focuses on the structures and functions needed to initialise the Monte Carlo simulation, like getting the energetic landscape or building up the neighbour network for hopping. The structures, subroutines and functions given in this file will be described on the next pages.

Before the first structure is defined, some physical constants, like the Boltzmann constant k_B or the elemental charge e, and derived variables of those constants, that are used in the code, are defined.

`hopping_parameters`
Structure containing all the parameters needed to calculate the Marcus and the Miller-Abrahams hopping rates.

`beta`:	$\beta = \frac{1}{k_B T}$	k_B:	Boltzmann constant
		T:	temperature
`two_alpha`:	$2 \cdot \alpha$	α:	charge delocalisation constant
`nu_h`:	ν_0	ν_0:	hopping prefactor
`E_r`:	E_r	E_r:	reorganization energy
`beta4Er`:	$4\beta E_r$	$4\beta E_r$:	constant needed to calculate Marcus rates
`injection_ratio`:	$\frac{\nu_{0,inj}}{\nu_{0,hop}}$	$\nu_{0,inj}$:	injection hopping prefactor
		$\nu_{0,hop}$:	bulk hopping prefactor

`hopping_info`
Structure with information about the hopping region, the neighbour network for hopping and the exponential factor needed to calculate distant hopping.

`radius`:	The hopping radius $r_{hop,cc}$ defines a sphere where hopping is allowed.
`radius_contact`:	The injection radius $r_{hop,inj}$ defines a half-sphere where injection is allowed.
`region`:	Instead of the hopping radius $r_{hop,cc}$, this variable can be used to define a hopping region. The region is a box of size $(2 \cdot$`region(i)`$+1)$ into dimension `i`.

`region_contact`:	Instead of the injection radius $r_{hop,inj}$, this variable can be used to define an injection region. The region is a box of size $(2\cdot$`region_contact(i)`$+1)$ into dimension `i` except normal to the contact where only hops to bulk cells up to a distance `region_contact(i)` are allowed.
`neighbours`:	A variable containing all the possible hopping neighbours of every cell in the bulk. `neighbours(i1,i2)` addresses the hopping neighbour into hopping direction `i2` of bulk cell number `i1`
`neigh_contact`:	A variable containing all the possible hopping neighbours of the injection cell in the simulation. `neigh_contact(i1,i2)` addresses the hopping neighbour into injection direction `i2` of injection cell number `i1`
`index_x`:	Transforms the hopping direction number used for `neighbours` into a hopping distance into the specified dimension. `index_x(i1,i2)` gives the change of cells into dimension `i1` of hopping direction `i2`. The result can of course be positive and negative.
`index_x_contact`:	Transforms the hopping direction number used for `neigh_contact` into a hopping distance into the specified dimension. `index_x_contact(i1,i2)` gives the change of cells into dimension `i1` of hopping direction `i2`. The result can of course be positive and negative.
`n`:	Number of neighbours where bulk hopping is allowed.
`n_contact`:	Number of neighbours where injection is allowed.
`exp_r`:	Factor $\nu_0 \exp(-2\alpha r)$ to consider delocalisation for the calculation of the hopping times in bulk. r is the hopping distance of the hopping direction.
`exp_r_to_contact`:	The same as `exp_r`, only the hopping distance r can be different if a hop from the contact to the bulk has a spatial offset.
`exp_r_from_contact`:	Factor $\nu_0 \cdot$ `injection_ratio` $\cdot \exp(-2\alpha r)$ to consider delocalisation for the calculation of the injection times. r is the hopping distance of the injection direction.

`interaction_info`
Structure containing the range and the strength of the interactions.

`radius`:	Value of the chosen cut-off radius r_c for interactions. For $r_c =$ `nan` Ewald summation method is used to calculate the exact potential of the charge carrier and all its periodic replica, where `nan` is the numerical data type 'not a number'.
`energy`:	The variable `energy(i1,i2,i3)` contains the interaction energy between two cells that are spatially separated by a 3D index distance of `(i1,i2,i3)` (all of them can be positive and negative).
`image`:	The variable `image(i1,i2,i3)` contains the image charge interaction energy between a cell and an image cell that are spatially separated by a 3D index distance of `(i1,i2,i3)` (`i1` and `i2` can be positive and negative and `i3` is starting from 1 for the charge carrier and the image charge carrier being in the first layer directly next to the metal contact).
`i1_min`, `i1_max`, `i2_min`, `i2_max`, `i3_min`, `i3_max`:	Lowest and highes index appearing in `energy` (indices in the range $i_1 = -i_{1,min} : i_{1,max}$, $i_2 = -i_{2,min} : i_{2,max}$ and $i_3 = -i_{3,min} : i_{3,max}$) and `image` (indices in the range $i_1 = -i_{1,min} : i_{1,max}$, $i_2 = -i_{2,min} : i_{2,max}$ and $i_3 = 1 : (2 \cdot i_{3,max} + 1)$).

`i1_mapping`, `i2_mapping`, `i3_mapping`, `i3image_mapping`:	Index difference between two cells when the position of one cell is changed to the hopping neighbours.
`i1_m_contact`, `i2_m_contact`, `i3_m_contact`, `i3image_m_contact`:	Index difference between two cells when the position of one cell is changed to the injection neighbours.

`update_info`
Structure used to define the region where the hopping rates have to be recalculated. After a hop or an injection, all charge carriers and injection cells within the specified region around the current position of the moved charge carrier are recalculated (except when the charge carrier recombines with the contact, then the previous cell of the charge carrier is taken as a centre of the update region).

`radius`:	Value of the chosen update radius $r_{up,cc}$ or $r_{up,inj}$.
`i1_min`, `i1_max`, `i2_min`, `i2_max`, `i3_min`, `i3_max`:	This variables hold the index boundaries used to define the update region, e.g. a sphere (see fig. 7.1).
`n`	Number of cells in the update region.

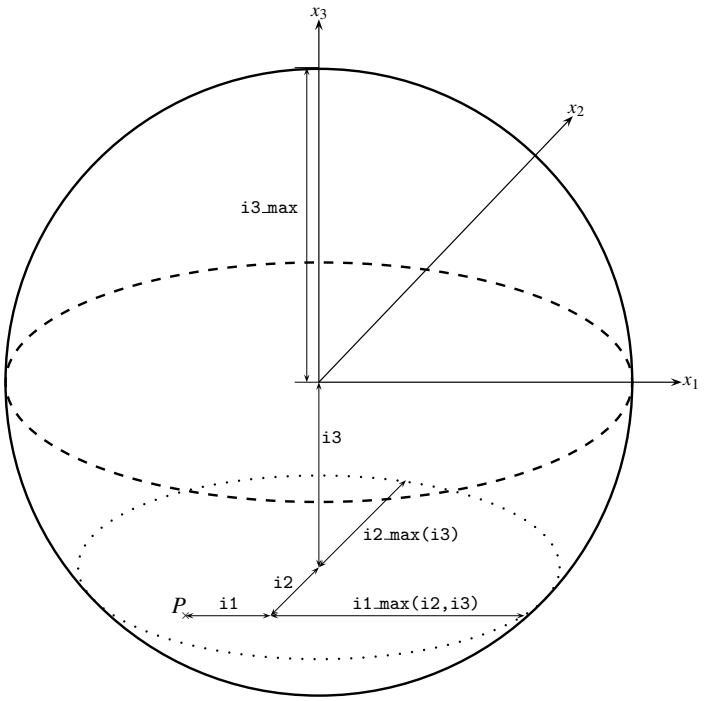

Figure 7.1: The variables `i1_min(i2,i3)`, `i1_max(i2,i3)`, `i2_min(i3)`, `i2_max(i3)`, `i3_min` and `i3_max` are defining index boundaries which can be used to address all elements in a certain shape. For this the index `i3` is running from `-i3_min` to `i3_max`, the index `i2` from `-i2_min(i3)` to `i2_max(i3)` and the index `i1` from `-i1_min(i2,i3)` to `i1_max(i2,i3)`. The example point P in the figure has the relative indices `(i1,i2,i3)` to the centre (note that all three are negative in this example).

`nearest_neighbour_info`
Structure that holds the information about the nearest neighbour network. This network is needed to perform grid searches in a certain region used e.g. for finding injection cells and charge carriers to update.

`neighbours`:	`neighbours(i1,i2)` contains the nearest neighbour of cell `i1` into direction `i2`. For a 3D cubic lattice `i2` $= 1,2$ and 3 are the positive x_1, x_2 and x_3 directions and `i2` $= 4,5$ and 6 are the negative ones.
`n`:	Number of nearest neighbours (in 3D cubic `n` = 6).

`lattice_info`
In this structure all information about the lattice is found.

`dim`:	Dimension of the lattice
`n_dim`:	Number of cells of the lattice into the corresponding dimension
`n_dim_half`:	For the calculation of the nearest periodic image of a charge carrier, half the lattice size is needed. This variable contains $\left\lfloor \frac{\texttt{n\_dim}}{2} \right\rfloor$.
`n_tot`:	Total number of cells including escape layers and contact layer
`add_dim`:	Number of cells that have to be skipped to get to the nearest neighbour in the corresponding direction. For the example lattice shown in fig. 2.1 the stored values would be $\texttt{add\_dim}(1) = 1$, $\texttt{add\_dim}(2) = 3$, $\texttt{add\_dim}(3) = 9$ and $\texttt{add\_dim}(4) = 27$. The last value is the same as n_tot, i.e. $\texttt{add\_dim}(\texttt{dim}+1) = \texttt{n\_tot}$.
`ltype`:	In this variable the type of the lattice is stored. 1: Cubic lattice with periodic boundary conditions in all directions for bulk simulations. 2: Cubic lattice with a contact and escape layers. The contact is the last layer in x_3 direction and the escape layers are the first layers in x_3 direction. Periodic boundary conditions are used into directions x_1 and x_2.
`btype`:	Holds which bulk type is used. 1: Homogeneous bulk, there is only one kind of material with one probability distribution for the energetic disorder 2: Additionally to the homogeneous bulk, there are static charges.
`ptype`:	Chosen potential type (detailed description see `get_external_potential` page 90): 1: only linear electric field 2: own image charge and linear electric field 3: additionally static charges (and if necessary their image charges) are considered
`stype`:	Simulation type regarding the update mechanism: 1: FRM 2: new update mechanism 3: DMC
`boundary_conditions`:	The boundary conditions in the desired dimension are specified here. Note that not all boundary conditions are available everywhere. Open boundary conditions result in entries that are 0 for the neighbours. Reflecting boundary conditions are recognised due to a negative sign of the neighbour (e.g. if reflecting boundary conditions into the positive direction x_3 would be established in the lattice shown in fig. 2.1, then the nearest neighbour of cell 23 into direction 3 (positive x_3 direction) would be -23). 1: Periodic boundary condition 2: Open boundary condition 3: Reflecting boundary condition in positive direction, open boundary condition in negative direction 4: Open boundary condition in positive direction, reflecting boundary condition in negative direction 5: Reflecting boundary condition in positive and negative direction
`constant`:	Real space distance between nearest neighbours (usually called lattice constant l)

bulk_end:	Cell number of the last cell in bulk before the contact starts. If no contact is present, it is equal to n_tot.
bulk_start:	Cell number of the first cell in bulk after the escape layers. If no contact is present, it is equal to 1.
cell_to_x:	Converts the cell number to a dim-dimensional index of cell. Cell number 12 in fig. 2.1 would give cell_to_x(12,1)=3, cell_to_x(12,2)=1 and cell_to_x(12,3)=2.

contact_info
Structure containing all necessary informations about the contact.

x:	Real space coordinate x_3 of the contact.
start:	Number of the first cell which belongs to the contact. The injection cells are numbered from start in the structure contact_info to n_tot in the structure lattice_info (see above).
n:	Number of cells in the contact.
layer:	Integer number of the layer of the contact, e.g. for 12 total layers, layers 0 to 10 are bulk layers and layer 11 is the contact layer.
image_x:	Integer used to calculate the layer (parallel to the contact) where the image charge carrier is found, e.g. if the layer of the charge carrier is L1 = 5 and the contact layer is layer = 9 then the image charge is in the imaginary layer L2 = 12 which is calculated by L2 = 2 · layer − 1 − L1, i.e. image_x = 2 · layer − 1 and L2 = image_x − L1.
hop_offset:	Offset of the hopping distance from the contact to the nearest neighbour cell in bulk. If hop_offset = 0.0 the distance from a injection cell to the nearest bulk cell for hopping or injection is one lattice constant, if hop_offset = 0.5 the distance from an injection cell to the nearest bulk cell for hopping or injection is half a lattice constant.
field_offset:	Offset of the position of the contact for field calculations. If field_offset = 0.0 the fields start at the centre of the injection cell, if field_offset = 0.5 the fields start at the boarder between the injection cells and the first cells in bulk. This offset is also considered for the image charges (own image charges and image charge interaction).

energy_info
Structure holding informations about the energetic landscape and the dynamical interactions.

morphology:	It holds which type a certain cell is, e.g. if it is a contact cell or a bulk cell.
static_part:	Contains all static energy contributions like energy levels of cells, own image charge potentials and potentials created by external fields.
dynamic_part:	Coulomb interaction between the charge carriers themselves and the charge carriers and other image charge carriers. This array is only updated when up_to_date is true.
up_to_date:	If true, the Coulomb interactions are continuously calculated by subtracting and adding monopole potentially to dynamic_part. If false, the Coulomb interactions are calculated by summing up all contributions of all charge carriers for every cell which energy level needs to be known.

get_lattice_and_contact_info
Subroutine for calculating certain variables in the structures lattice_info (see page 87) and contact_info (see page 88).

The calculated variables are add_dim, n_dim_half, boundary_conditions, n_tot, bulk_end, bulk_start and cell_to_x in the structure lattice_info as well as x, n, start, layer and image_x in the structure contact_info.

The subroutine is used once during the initialisation process.

get_nearest_neighbours
In this subroutine the variable neighbours in the structure nearest_neighbour_info (see page 86) is calculated.

It is called once during the initialisation process.

get_interaction_info
Subroutine that calculates the variables in the structure interaction_info (see page 85). The radius has to be handed in and all other variables in the structure interaction_info are allocated and calculated. If the interaction radius is nan, Ewald summation calc_ewald_sum (see page 90) is used to calculate the exact interaction potential including all periodic images. The results of the Ewald summation method can be stored to a file and for later simulations also read out of a file to save computer time.

The subroutine is called once during the initialisation process.

get_update_info_sphere
This subroutine allocates and calculated all variables in structure update_info (see page 86) for a given update radius.

Usually it is called once during the initialisation process. For injection simulations where the update radius for charge carriers in bulk is different to the update radius of injection cells, it is called twice.

get_hopping_info_box
This subroutine calculates the neighbours in the structure hopping_info (see page 84). For a hop, the maximum hopping distance into a direction x_i to the target cell is given by the variable region(i) in hopping_info. So this variable defines a cube of size $(2 \cdot \texttt{region}(1)+1) \times (2 \cdot \texttt{region}(2)+1) \times (2 \cdot \texttt{region}(3)+1)$ where hopping is allowed. For all cells the neighbours within that region, considering the chosen boundary conditions, are stored in neighbours.

It is called during the initialisation process if a cubic hopping region is desired.

get_hopping_info_sphere
The same as subroutine get_hopping_info_box (see above). The only difference is that the hopping region is not a cube, but a sphere with a specified radius found in the structure hopping_info (see page 84). A value has to be assigned to the hopping radius before the subroutine is evaluated.

It is called during the initialisation process if a spherical hopping region is desired.

get_coordinates_from_cell_number
Subroutine calculating the real space position of the centre of a cell addressed by its cell number. Within this subroutine get_indices_from_cell_number (see page 89) is called.

It is used in add_external_potential (see page 90) and for visualisations.

get_indices_from_cell_number
Out of the cell number this subroutine calculates the three dimensional index of the cell. Looking at fig. 2.1 e.g. cell number 6 would have the 3D index $(3,2,1)$.

The subroutine is used in subroutines get_lattice_and_contact_info (see page 88) to evaluate the variable cell_to_x in structure lattice_info (see page 87).

`get_morphology`
In this subroutine the morphology of the system is determined. If the bulk type `btype` in the structure `lattice_info` (see page 87) is 1 (homogeneous bulk) then the entries in the variable `morphology` in structure `energy_info` (see page 88) are 1 for the bulk (i.e. all the energy levels are drawn from the same probability distribution) and 2 for the contact. $\texttt{btype} = 2$ additionally introduces static charges placed uniformly distributed over the bulk. Charges with the same polarity as the charge carriers are labelled 3 in `morphology` and charges with a different polarity are labelled 4.

This subroutine is used during the initialisation process or if multiple morphologies are sampled, it is used once at the beginning of every run.

`get_energetic_landscape`
This subroutine uses the variable `morphology` in structure `energy_info` to assign each cell its energy level and store it in variable `static_part` of structure `energy_info` (see page 88). A function `randE` has to be provided by the user, which assigns the right energy levels for the different cell types. This function receives the cell type and gives back the energy level, e.g. for $\texttt{morphology(i1)} = 1$ a random number from a Gaussian distribution is drawn or for $\texttt{morphology(i1)} = 2$ the energy level is set to the Fermi energy of the metal contact.

This subroutine is called during the initialisation process or if multiple energetic landscapes are sampled, it is used once at the beginning of every run.

`add_external_potential`
Depending on the chosen `ptype`, found in the structure `lattice_info` (see page 87), the external potentials are added to the variable `static_part` of structure `energy_info` (see page 88).

If $\texttt{ptype} = 1$ only a linear potential from an electric field pointing in the x_3 direction is considered. With contact the potential starts with 0 at the contact and decreases departing from the contact. Without a contact the lowest potential is at low x_3 values and increases with increasing x_3.

$\texttt{ptype} = 2$ additionally includes the image potential of the charge carrier itself. In this case a contact has to be present as else there would be no image charge.

In case $\texttt{ptype} = 3$ static charges are included. The Coulomb interaction potential for those charges are considered in the same way as interactions between charge carriers (with a cut-off radius or Ewald summation). If the contact is present, also image charges of the static charges and the image charge of a single charge carrier are concerned. Additionally the potential of an electric field can be considered in the same way as for $\texttt{ptype} = 1$.

The subroutine is called once during the initialisation process. If sampling over multiple arrangements of the static charges is performed, then the external potential has to be recalculated at the beginning of each run.

`change_charge_neutrality_dipole_potential`
With this subroutine the potential of a homogeneous charge background all-over the system can be added or changed. The dipole is created by the consideration of the image charge background.

This subroutine would be needed if charge neutrality has to be assured far apart from the contact.

`calc_direct_sum`
Calculate the potential field created by the charge carrier formation which is used for the Ewald summation method to check if the Ewald summation method is leading to the correct values. The convergence of this direct summation is very poor compared to the Ewald summation.

`calc_ewald_sum`
Calculate the potential field created by the charge carrier formation described in chapter 2.2.3 with Ewald

summation method.

This subroutine is used in `get_interaction_info` (see page 89) if the interaction `radius` is chosen `nan`.

7.1.10 `mc_module.f90`

Together with `morphology_module.f90` (see chapter 7.1.9) this is the central module for the kinetic Monte Carlo simulation. Different to the subroutines found in `morphology_module.f90`, in which most of the subroutines are only used during the initialisation process, the subroutines in `mc_module.f90` are mainly for the Markov Chain Monte Carlo process. The focus is on bulk hopping, injection and combining those two processes with different update mechanisms.

The module can in principle be used for bulk and injection simulations. As there are some very time critical routines in this module, not all differences are covered with an if-statement. The sections that have to be commented or uncommented for changing from bulk to injection simulations are marked with the following comment above it:

```
!  change bulk-injection
```

`carrier_info`
A structure containing all information about the charge carriers. As new charge carriers can be created and annihilated due to injection, recombination with the contact and escape processes, the storing of the properties of the charge carriers is a task were a lot of time can be saved. The current number of charge carriers in the simulation is held in `n`. It would be intuitive, that the indices of the charge carriers currently in the simulation go from 1 to `n`. But as this would imply that after every creation or annihilation, all the lists of the properties of the charge carriers would have to be resorted, it is too time consuming to implement it in that way. A maybe faster way is to use `queue` and `free` to manage the indices of the charge carriers that are in the simulation and those who are available to be injected. In `queue` the indices of the charge carriers that are currently in use are stored from 1 to `n` sorted by their hopping time `tau` and in `free` the indices of the unused charge carriers are found from 1 to `i_free`.

During a creation the new charge carrier assigns the index `free(i_free)` and `i_free` is decremented by 1. Additionally the hopping time `tau` is calculated for the new charge carrier, its index is sorted into `queue` and the number of present charge carriers `n` is incremented by 1.

An annihilation works the other way round, `n` is decremented and `i_free` incremented, the index of the annihilated charge carrier is stored in `free(i_free)` and in `queue` the index is ejected and the appearing gap is closed by the slower charge carriers.

The strength of this approach is, that only `queue` has to be resorted after a creation or annihilation, but this has to be done anyway, as the changed hopping times always need to be resorted. No strength without weakness, unfortunately the charge carriers cannot be addressed directly, one always needs `queue` to find the right index.

`occupied`:	This variable, with indices going from 1 to `bulk_end` in the structure `contact_info` (see page 88), indicates if a bulk cell is occupied by a charge carrier. An occupied cell is marked with the index of the charge carrier that occupies it, all unoccupied cells are labelled by 0.
`cell`:	Holds the current cell of the charge carriers. As for all other variables, the stored value of an unused charge carrier is not specified.
`hop_to`:	Here the suggested cell where the charge carrier would hop to is found.
`tau`:	Calculated retention time of the suggested hop.
`queue`:	In `queue(1:n)` the indices of the used charge carriers are stored. The order is determined by the retention times of the next hops with the fastest first.
`rates`:	Calculated rates of the charge carrier, only needed for DMC.
`n`:	Number of charge carriers that are currently used in the simulation.

n_max:	Number of available charge carriers. If more than n_max charge carriers are tried to be injected, an error is produced and the program stops.
free:	free(1:i_free) holds the indices of the charge carriers that are currently not used in the simulation.
i_free:	Number of charge carriers that are still available for injection. free(i_free) is the charge carrier that will be injected next and escaped last.
injected:	Number of charge carriers that are injected during the simulation up to the current Markov time.
escaped:	Number of charge carriers that escaped during the simulation up to the current Markov time.
contact:	Number of charge carriers that recombined with the contact during the simulation up to the current Markov time.
update:	Indices of the charge carriers for which the hopping time should be updated.
n_update:	Number of charge carriers for which the hopping time should be updated.
direction:	Direction of the next hop corresponding to the directions defined by neighbours and index_x in the structure hopping_info (see page 84).
t_hop:	Markov time where the charge carriers was hopping for the last time. This variable is used to calculate the occupation times of the cells and hence the spatially resolved charge carrier densities.

injection_info

Like the structure carriers_info for charge carriers (see page 91), this structure contains information about the injection process. In principle it is built up in the same way as carrier_info, but less complicated due to the fact, that the number of injection cells in the contact does not change during the simulation. So all the fields have a size of n in the structure contact_info (see page 88) and all those indices are always in use.

cell:	Index number of the injection cells where injection can take place. Used to find the potential cells in bulk where the injection process can place a charge carrier.
hop_to:	Number of the cell in bulk where the charge carrier would be injected.
tau:	Retention time when the injection would be performed
queue:	Indices of the injection cells sorted by their retention time tau.
rates:	Calculated rates of the injection processes, only needed in DMC.
update:	Indices of the injection cells for which the injection process should be updated.
n_update:	Number of injection cells for which the injection process should be updated.
direction:	Direction of the next injection corresponding to the directions defined by neigh_contact and index_x_contact in the structure hopping_info (see page 84).

measure_info

This structure is used to measure quantities during the simulation that are needed for the evaluation.

time:	Current Markov time of the simulation.
frm_failed:	Number of times that a hop or injection would have been performed onto an occupied cell in FRM and a recalculation of the process was necessary.
n_up_inj_radius:	Number of times that a total injection update was performed for the new update mechanism.
injection_count:	Number of injections that were performed from a certain contact cell.
recomb_count:	Number of charge carrier that recombined with a certain contact cell.
occ_time:	Markov time that a certain cell in bulk was occupied, used to measure the spatially resolved charge carrier density.
occ_count:	Number of times that a cell was occupied.

`distance_cell`: Hopping distance that was overcome within hops from a certain bulk or contact cell, used to measure spatially resolved current densities.

`calc_miller_abrahams_rates`
The central subroutine to calculate the rates with the help of the Miller-Abrahams rate equation (1.100). The parameters for all possible neighbouring hopping cells are handed over and this subroutine calculates the individual rates to all those cells and the sum of them.

The header is exactly the same as in subroutine `calc_marcus_rates` (see page 93), so that a function handle can be used to decide externally if the Miller-Abrahams rate equation or Marcus theory should be used to calculate the rates without loosing computational performance.

The function handles (internally called `fct_calc_hopping_rates` or `fct_calc_injection_rates`) are used by the following subroutines:

`calc_single_hopping_time` (see page 93)
`calc_single_injection_time` (see page 94)
`calc_time_until_injection_needs_update` (see page 96)
`calc_new_hopping_time_dmc` (see page 99)
`calc_new_injection_dmc` (see page 99)
`calc_layer_convergence` (see page 99)

`calc_marcus_hopping_time`
The central subroutine to calculate the rates with the help of Marcus theory (1.99). The parameters for all possible neighbouring hopping cells are handed over and this subroutine calculates the individual rates to all those cells and the sum of them.

The header is exactly the same as in subroutine `calc_miller_abrahams_rates` (see page 93), so that a function handle can be used to decide externally if Marcus theory or the Miller-Abrahams rate equation should be used to calculate the rates without loosing computational performance.

The function handles (internally called `fct_calc_hopping_rates` or `fct_calc_injection_rates`) are used by the following subroutines:

`calc_single_hopping_time` (see page 93)
`calc_single_injection_time` (see page 94)
`calc_time_until_injection_needs_update` (see page 96)
`calc_new_hopping_time_dmc` (see page 99)
`calc_new_injection_dmc` (see page 99)
`calc_layer_convergence` (see page 99)

`calc_single_hopping_time`
The cell number of the charge carrier to recalculate the hopping time is handed over to this subroutine. By calling `calc_energies_with_interaction_up_to_date_hop` (see page 97) or `calc_energies_with_interaction_without_precalc_hop` (see page 96), the neighbouring hopping cells with their corresponding energies and delocalisation factors are evaluated and handed over to the function `calc_miller_abrahams_rates` (see page 93) or `calc_marcus_rates` (see page 93). The choice of the function is determined externally by handing in the function handle `func_calc_hopping_rates`. With the calculated rates a certain hop is chosen randomly and a retention time is assigned corresponding to theorem 1.6. If all potential hopping cells are occupied, the cell where the hop should go to is the current cell and an infinite retention time is handed over. The case that the cell has no neighbour at all, which is characteristic for an escape cell, has to be prevented proir to calling this subroutine.

The subroutine is called by:

`calc_multiple_hopping_times` (see page 94)

perform_single_hop_frm (see page 94)
perform_single_injection_frm (see page 95)

calc_multiple_hopping_times
This subroutine recalculates the hopping times for all charge carriers whose indices are found in update(1:n_update) from structure carrier_info (see page 91). To recalculate the hopping times it uses calc_single_hopping_time (see page 93). The charge carriers with new hopping times are sorted into queue in the structure carrier_info (see page 91).

This subroutine is used during the initialisation process in some simulations and called by the following subroutines:

perform_single_hop_radius (see page 94)
perform_single_injection_radius (see page 95)
recalc_tau_inf_carriers_frm (see page 96

perform_single_hop_frm
This subroutine performs the fastest hop and updates the hopping time of this charge carrier after the hop took place, considering recombination with the contact and the escape process. To calculate the retention time calc_single_hopping_time (see page 93) is used. The retention times of all other charge carriers and all injection processes (if present in the simulation) are reduced by the current retention time. At the end the hopping time is sorted into queue in the structure carrier_info.

The subroutine is called by perform_single_step_frm (see page 95) and for some simulations it is used in the Markov Chain Monte Carlo iteration.

perform_single_hop_radius
After the fastest hop is performed in this subroutine and recombination and escape is checked, the hopping times for all charge carriers and injection cells within the update radius $r_{up,cc}$ and $r_{up,inj}$ defined in the structure update_info (see page 86) are recalculated. The update radius $r_{up,cc}$ has to be at least as big as the hopping radius $r_{hop,cc}$ to prevent unintended double occupation and as a consequence misbehaviour of the program. The charge carriers to update are found with the subroutine find_carriers_to_update (see page 97) and recalculated with calc_multiple_hopping_times (see page 94). The injection cells to update are found with find_injection_cells_to_update_sphere (see page 98) and recalculated with calc_multiple_injection_times (see page 95). The retention times of charge carriers and injection cells that are not recalculated are reduced by the retention time of the current hop.

The subroutine is called by perform_single_step_radius (see page 95) or directly in the Markov Chain Monte Carlo iteration.

calc_single_injection_time
Similar to calc_single_hopping_time (see page 93), an injection cell is handed over to this subroutine. With this it finds the potential injection cells with the associated energy levels and the delocalisation factors with calc_energies_with_interaction_up_to_date_injection (see page 97) or calc_energies_with _interaction_without_precalc_injection (see page 97) and calculates the injection rates with calc_miller_abrahams_rates (see page 93) or calc_marcus_rates (see page 93) depending on the chosen function selected by the function handle fct_calc_injection_time. If all allowed hopping cells of the injection cell are currently occupied, the cell where the injection should go to is the injection cell itself and the assigned hopping time gets infinite.

The subroutine is called by:

calc_multiple_injection_times (see page 95)
perform_single_injection_frm (see page 95)

`calc_multiple_injection_times`
This subroutine recalculates all injection times of the injection cells whose indices are found in `update(1:n_update)` in the structure `injection_info` (see page 92) with the subroutine `calc_single_injection_time` (see page 94). Afterwards the new injection times are sorted into `queue` in the structure `injection_info` (see page 92).

This subroutine is used during the initialisation process and called by:

`perform_single_hop_radius` (see page 94)
`perform_single_injection_radius` (see page 95)
`recalc_tau_inf_injection_frm` (see page 96
`check_if_injection_needs_update` (see page 96)

`perform_single_injection_frm`
The fastest injection is performed in this subroutine. The hopping time of the created charge carrier is calculated with `calc_single_hopping_time` (see page 93) and it is included in `queue` in the structure `carrier_info` (see page 91). The escape process is also considered, but no recombination with the contact, as then the charge carrier would not be injected. The retention time of the injection process is recalculated with `calc_single_injection_time` (see page 94) and sorted into `queue` in the structure `injection_info` (see page 92). As the former charge carrier situation around the injection cell is thought to be more representative for the next injection process, the Coulomb interactions of the currently injected charge carrier are not considered for the calculation of the injection time. The retention times of all other injection cells and charge carriers are reduced by the retention time of the current injection.

The subroutine is called by `perform_single_step_frm`.

`perform_single_injection_radius`
In this subroutine the fastest injection is performed and all hopping times of the charge carriers and injection cells within the update radius $r_{up,cc}$ and $r_{up,inj}$ are updated. To find the charge carriers that need to be updated `find_carriers_to_update` (see page 97) is used and they are updated with `calc_multiple_hopping_times` (see page 94). The injection cells that need to be updated are found with `find_injection_cells_to_update_sphere` (see page 98) and updated with `calc_multiple_injection_times` (see page 95). The retention times of the charge carriers and injection cells, that are not recalculated, are reduced by the retention time of the current injection.

The subroutine is called by `perform_single_step_radius`.

`perform_single_step_frm`
This subroutine compares the fastest hopping time and the fastest injection time. It performs a hop with `perform_single_hop_frm` (see page 94) if the hopping time is lower or an injection with `perform_single_injection_frm` (see page 95) if the injection time is lower. Previously it checks if there is a charge carrier in the simulation and performs an injection if not. For an injection process an unused charge carrier has to be available, so trying to inject when `i_free` in the structure `carrier_info` (see page 91 is zero leads to an error stopping the whole simulation.

This subroutine is used during the Markov Chain Monte Carlo iteration if FRM is used.

`perform_single_step_radius`
This subroutine compares the fastest hopping time and the fastest injection time. It performs a hop with `perform_single_hop_radius` (see page 94) if the hopping time is lower or an injection with `perform_single_injection_radius` (see page 95) if the injection time is lower. Previously it checks if there is a charge carrier in the simulation and performs an injection if not. For an injection process an unused charge carrier has to be available, so trying to inject when `i_free` in the structure `carrier_info` (see page 91 is zero leads to an error stopping the whole simulation.

When updating charge carriers and injection cells within a certain update radius $r_{up,cc}$ and $r_{up,inj}$ is desired, this subroutine is used during the Markov Chain Monte Carlo iteration.

`recalc_tau_inf_carriers_frm`
In FRM simulations it is possible, that hopping times get infinite, because all the potential hopping neighbours of a charge carrier are occupied. With other update mechanism this is no problem as the times are recalculate anyway when a neighbouring cell gets unoccupied, but in FRM this would lead to a charge carrier that is fixed to its position for ever. To overcome such a scenario, this subroutine recalculates all hopping times that are infinite.

This subroutine is only used for high charge carrier densities during the Markov Chain Monte Carlo iteration. It is not performed in every iteration, but after a certain number of steps or a certain number of infinities it could be called.

`recalc_tau_inf_injection_frm`
In FRM simulations it is possible, that injection times get infinite, because all the potential hopping neighbours of an injection cell are occupied. With other update mechanism this is no problem as the times are recalculate anyway when a neighbour gets unoccupied, but in FRM this would lead to an injection cell that would not inject any more. To overcome such a scenario, this subroutine recalculates all injection times that are infinite.

This subroutine is only used for high charge carrier densities during the Markov Chain Monte Carlo iteration. It is not performed in every iteration, but after a certain number of steps or a certain number of infinities it could be called.

`calc_time_until_injection_needs_update`
This subroutine calculates the Markov time between two total injection updates as described in chapter 3.2.3.

It is called once after the extended thermalisation for injection simulations with the new update mechanism.

`check_if_injection_needs_update`
To assure the accuracy of an injection simulation with very low charge carrier densities, this subroutine checks if a total injection update is required (see chapter 3.2.3). It needs the Markov time between two total injection updates calculated with `calc_time_until_injection_needs_update` (see page 96).

The subroutine is used during the Markov Chain Monte Carlo iteration for an injection simulation with the new update mechanism if a total injection update is required to achieve a small enough methodological error for regimes with a very low charge carrier density.

`calc_energies_with_interaction_without_precalc_hop`
This subroutine is needed to prepare the information needed to calculate the hopping rates when the interactions contained in variable `dynamic_part` in structure `energy_info` (see page 88) are not up to date. A bulk cell is handed over to this subroutine and it finds the neighbours that are available for hopping and calculates its energy levels including the static potentials as well as interactions with other charge carriers and image charges. The interactions are calculated by directly summing up all Coulomb contributions of all other charge carriers and their image charges. The spatial decay term for the allowed hops is determined as well. It is also concerned that only one charge carrier can occupy a cell, occupied cells are rejected from the list of possible hopping cells.

This subroutine is called by:

`calc_single_hopping_time` (see page 93)
`calc_new_hopping_time_dmc` (see page 99)

`calc_energies_with_interaction_up_to_date_hop`
This subroutine is needed to prepare the information needed to calculate the hopping rates when the interactions contained in variable `dynamic_part` in structure `energy_info` (see page 88) are up to date. A bulk cell is handed over to this subroutine and it finds the neighbours that are available for hopping and calculates its energy levels including the static potentials as well as interactions with other charge carriers and image charges. The interactions are derived directly from `dynamic_part`. The spatial decay term for the allowed hops is determined as well. It is also concerned that only one charge carrier can occupy a cell, occupied cells are rejected from the list of possible hopping cells.

This subroutine is called by:

`calc_single_hopping_time` (see page 93)
`calc_new_hopping_time_dmc` (see page 99)

`calc_energies_with_interaction_without_precalc_injection`
This subroutine is needed to prepare the information needed to calculate the injection rates when the interactions contained in variable `dynamic_part` in structure `energy_info` (see page 88) are not up to date. An injection cell is handed over to this subroutine and it finds the neighbours that are available for hopping and calculates its energy levels including the static potentials as well as interactions with other charge carriers and image charges. The interactions are calculated by directly summing up all Coulomb contributions of all other charge carriers and their image charges. The spatial decay term for the allowed injections is determined as well. It is also concerned that only one charge carrier can occupy a cell, occupied cells are rejected from the list of possible hopping cells.

This subroutine is called by:

`calc_single_injection_time` (see page 94)
`calc_new_injection_time_dmc` (see page 99)

`calc_energies_with_interaction_up_to_date_injection`
This subroutine is needed to prepare the information needed to calculate the injection rates when the interactions contained in variable `dynamic_part` in structure `energy_info` (see page 88) are up to date. An injection cell is handed over to this subroutine and it finds the neighbours that are available for hopping and calculates its energy levels including the static potentials as well as interactions with other charge carriers and image charges. The interactions are derived directly from `dynamic_part`. The spatial decay term for the allowed injections is determined as well. It is also concerned that only one charge carrier can occupy a cell, occupied cells are rejected from the list of possible hopping cells.

This subroutine is called by:

`calc_single_injection_time` (see page 94)
`calc_new_injection_time_dmc` (see page 99)

`find_carriers_to_update`
A cell number is handed over to this subroutine and from this cell, the charge carriers that are within a distance $r_{up,cc}$ to it are marked for updating the hopping time. The cell number that is handed in is usually the new position of the currently hopped or injected charge carrier. For recombination and the escape process the cell where the charge carrier was before it hopped is taken as a centre for the update region. The information about the update region is found in the structure `update_info` (see page 86). All the marked charge carriers are held by `update(1:n_update)` in structure `carrier_info` (see page 91).

To find the charge carriers, two different opportunities are imaginable. The first opportunity is to go through all the charge carriers and find out if their distance to the considered cell is lower than the update radius $r_{up,cc}$. For low charge carrier densities and a big update radius this is the method of choice. If

the charge carrier density is high and/or the update radius $r_{up,cc}$ is small, the number of charge carriers gets higher than the number of potential update cells and hence it is preferable to have a look at if the cells in the update region house a charge carrier or not. Depending on the number of update cells `n` in structure `update_info` (see page 86) and the number of charge carriers in the simulation `n` in structure `carrier_info` (see page 91), either the update region is scanned for charge carrier or the distance to all charge carriers is checked to find out which charge carriers have to be updated.

This subroutine is called by:

`perform_single_hop_radius` (see page 94)
`perform_single_injection_radius` (see page 95)

`find_injection_cells_to_update_sphere`
With the cell number where the charge carrier is at the moment (or was previously for recombination and the escape process), all injection cells within the radius $r_{up,inj}$ are marked for updating their hopping time. This marking is done by writing their indices into `update(1:n_update)` in the structure `injection_info` (see page 92).

For finding the injection cells whose injection time needs to be recalculated, the distance from the committed bulk cell to the nearest injection cell is evaluated. Starting from this nearest injection cell, a circle is drawn with a radius determined by the distance between the committed bulk cell and the nearest injection cell. All injection cells within this circle are marked for updating.

This subroutine is called by:

`perform_single_hop_radius` (see page 94)
`perform_single_injection_radius` (see page 95)

`find_injection_cells_to_update_projection`
This subroutine in principle is the same as `find_injection_cells_to_update_sphere` (see page 98) with the only difference that we are not updating all injection cells within an update sphere but projecting a circle with radius $r_{up,inj}$ onto the contact layer and update all injection cells within this projection. So the update shape is rather a cylinder with radius $r_{up,inj}$ and the axis is oriented into x_3 direction.

The subroutine was written to check if this update is leading to better results for the convergence problem of regime 3 (see chapter 3.2.3), but it does not. So it is no longer in used.

`perform_single_step_dmc`
This subroutine compares the calculated hopping time and injection time for a DMC simulation and performs a hop with `perform_single_hop_dmc` (see page 98) if the hopping time is lower or an injection with `perform_single_injection_dmc` (see page 99) if the injection time is lower. Previously it checks if there is a charge carrier in the simulation and performs an injection if not. For an injection process an unused charge carrier has to be available, so trying to inject when `i_free` in the structure `carrier_info` (see page 91 is zero leads to an error stopping the whole simulation.

When performing a DMC injection simulation is desired, this subroutine is used during the Markov Chain Monte Carlo iteration.

`perform_single_hop_dmc`
After the chosen hop is performed in this subroutine and recombination and escape is checked, the new hopping time and injection time is recalculated with `calc_new_hopping_time_dmc` (see page 99) and `calc_new_injection_time_dmc` (see page 99).

The subroutine is called by `perform_single_step_dmc` (see page 98) or directly in the Markov Chain Monte Carlo iteration.

`perform_single_injection_dmc`
In this subroutine the chosen injection is performed and escape is checked. Afterwards the new hopping time and injection time is recalculated with `calc_new_hopping_time_dmc` (see page 99) and `calc_new_injection_time_dmc` (see page 99).

The subroutine is called by `perform_single_step_dmc` (see page 98).

`calc_new_hopping_time_dmc`
All rates for all hopping possibilities of all charge carriers are calculated and according to theorem 1.6, an exponentially distributed random number with the sum over all rates as a rate parameter is drawn for the retention time and one of the hops is randomly chosen according to its rate.

This subroutine is called by:

`perform_single_hop_dmc` (see page 98)
`perform_single_injection_dmc` (see page 99)

`calc_new_injection_time_dmc`
Like for the hopping process, this subroutine calculates all rates for all hopping possibilities of all injection cells and according to theorem 1.6, an exponentially distributed random number with the sum over all rates as a rate parameter is drawn for the retention time and one of the injections is randomly chosen according to its rate.

This subroutine is called by:

`perform_single_hop_dmc` (see page 98)
`perform_single_injection_dmc` (see page 99)

`update_energy_dynamic_part`
This subroutine is used to keep the variable `dynamic_part` in structure `energy_info` (see page 88) up to date. After every Markov jump the previous and the new cell number of the charge carrier that has just hopped is handed in and the monopole potential centred at the previous cell is subtracted from `dynamic_part` where the monopole potential centred at the new cell is added to `dynamic_part`. The update of the dynamic Coulomb interactions is only done if `up_to_date` in structure `energy_info` is true. In this case the subroutine is used by:

`perform_single_hop_frm` (see page 94)
`perform_single_injection_frm` (see page 95)
`perform_single_hop_radius` (see page 94)
`perform_single_injection_radius` (see page 95)
`perform_single_hop_dmc` (see page 98)
`perform_single_injection_dmc` (see page 99)

`initialise_energy_dynamic_part`
Starting from a blank variable `dynamic_part` in structure `energy_info` (see page 88) the Coulomb interactions of all charge carriers and their image charges are added to `dynamic_part`.

This subroutine is needed to initialise the variable `dynamic_part` in structure `energy_info` before the Markov Chain Monte Carlo iteration is started when `up_to_date` in structure `energy_info` is true.

`calc_layer_convergence`
With this subroutine the needed layers parallel to the contact to undercut a certain return probability are evaluated (see chapter 2.2.2).

For injection simulations, it is called once during the initialisation.

`allocate_and_initialise_fields`
This subroutine allocates all necessary fields in the structures

`carrier_info` (see page 91)
`injection_info` (see page 92)
`measure_info` (see page 92)
`energy_info` (see page 88)

with the correct size.

It is used during the initialisation process. Depending on which structures are handed in, only the arrays that are in those structures are allocated. So it can be called multiple times during the initialisation to allocate only some arrays one after the other.

`deallocate_fields`
This subroutine deallocates all the fields that are used during the simulation. The fields are found in the structures

`carrier_info` (see page 91)
`injection_info` (see page 92)
`measure_info` (see page 92)
`lattice_info` (see page 87)
`hopping_info` (see page 84)
`interaction_info` (see page 85)
`update_info` (see page 86)
`nearest_neighbour_info` (see page 86)
`energy_info` (see page 88)

This subroutine is used after all simulations are done at the end of the program. It can also be used to deallocate the arrays in one particular structure if only this structure is handed in. This can be useful when certain arrays need to be deallocated during the simulation.

Dependency Diagram
In module `mc_module.f90` the dependencies are much more complicated than in module `morphology_module.f90` (see chapter 7.1.9), where in principle no dependencies are present. For a better understanding the important dependencies in `mc_module.f90` are shown in fig. 7.2, fig. 7.3 and fig. 7.4.

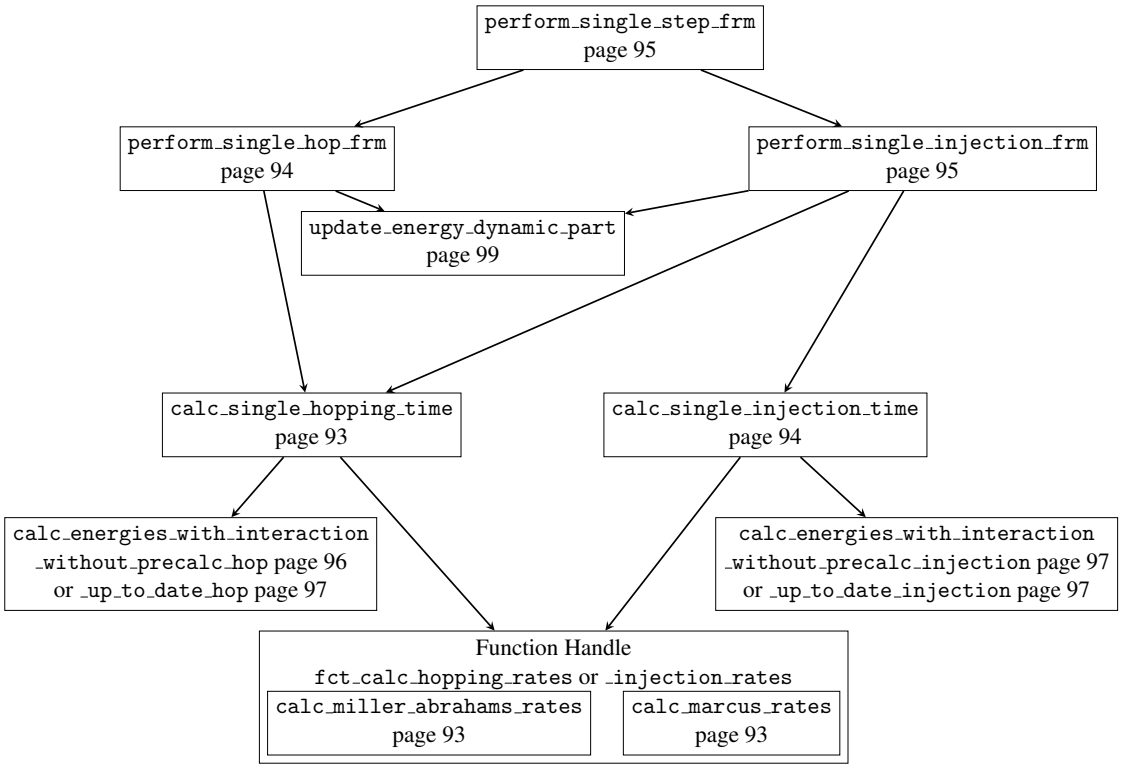

Figure 7.2: Dependency diagram of the subroutines found in the module `mc_module.f90` used for an FRM simulation.

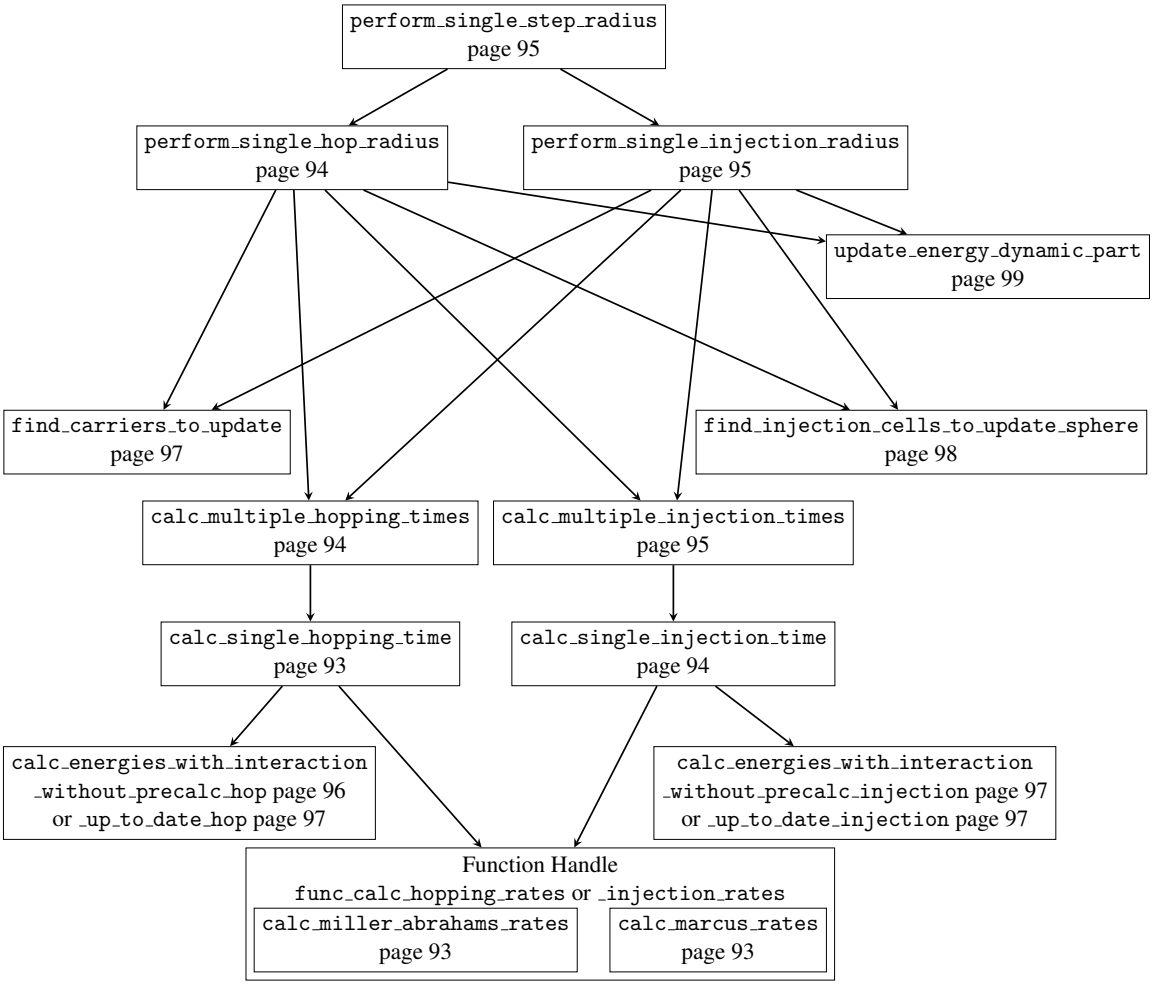

Figure 7.3: Dependency diagram of the subroutines found in the module `mc_module.f90` used for a simulation with the new update mechanism.

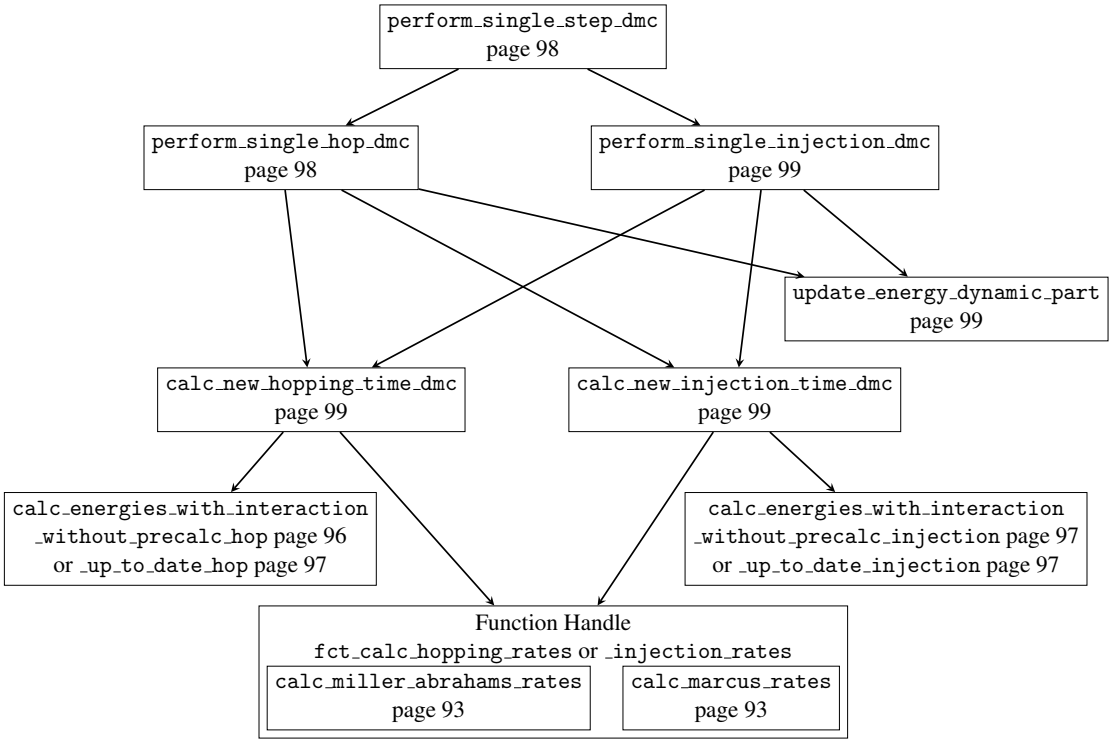

Figure 7.4: Dependency diagram of the subroutines found in the module `mc_module.f90` used for a DMC simulation.

7.1.11 Structograms

Most of the simulations are either bulk simulations or injection simulations. In the following, the basic concepts of the main program for those simulations are schemed.

Bulk Simulation
The term bulk simulation means, that a cubic lattice with periodic boundary conditions in all directions is simulated and an electric field is applied in x_3 direction. The number of charge carriers is constant during the simulation, as no recombination, no escape and no injection takes place. The global variable `qEfield_int` found in the module `mc_module.f90` is used to consider the periodic boundary conditions for the electric field correctly. The electric field must not be considered as an external potential in `add_external_potential` (see page 90). This means without additional external potentials like static charges, `add_external_potentials` does not have to be evaluated and if further external potentials are considered, `Efield` $= 0$ has to be handed over to this subroutine. The variable `qEfield_int` is the electric field strength times the elemental charge of the used charge carriers and has to be provided. The potential drop due to this electric field is directly considered in the calculation of the energy levels in `calc_energies_with_interaction_without_precalc_hop` (see page 7.1.10) and `calc_energies_with_interaction_up_to_date_hop` (see page 7.1.10). Although the contact is not used here, internally the structure `contact_info` (see page 88) is necessary. This is due to the fact that the simulation is optimised for the injection simulation. A typical structogram of a bulk simulation for our new update mechanism is found in fig. 7.5. When FRM or DMC is used as update mechanism, the structogram has to be slightly changed, especially the called functions in the Markov Chain Monte Carlo iteration have to be chosen differently. If `up_to_date` in structure `energy_info` (see page 88) is true, the subroutine `initialise_energy_dynamic_part` (see page 99) has to be called additionally at the end of the initialisation process.

Injection Simulation
The cubic lattice is limited by a contact in the positive x_3 direction and escape layers (the number of escape layers corresponds to the maximum hopping distance) in the negative x_3 direction. Periodic boundary conditions are introduced into the other directions. As the program is optimised for injection simulations, the implementation is much more straight forward compared to a bulk simulation. In fig. 7.6 the stuctogram of a typical injection simulation for our new update mechanism can be seen. Structograms for other update mechanisms (FRM or DMC) are comparable with slight changes of the used subroutines in the Markov Chain Monte Carlo iteration.

	Choose parameters for the simulation held by the structures `hopping_parameters`, `lattice_info`, `contact_info` and `energy_info`	
	Choose the cut-off radius r_c, the update radius $r_{up,cc}$ and the hopping radius $r_{hop,cc}$	
	Choose the energetic disorder σ, the relative permittivity ε_r and the electric field strength F to set `qEfield_int`	
	call `get_lattice_and_contact_info`	
	call `get_nearest_neighbours`	
	call `get_hopping_info_sphere`	
	call `get_interaction_info`	
	call `get_morphology`	
	Provide the function `randE` which assigns the randomly chosen energy levels	
	call `get_energetic_landscape`	
	call `get_update_info_sphere` for bulk hopping with $r_{up,cc}$	
	Choose parameters for the Markov Chain Monte Carlo process like the number of thermalisation steps `n_thermalise` and the number of Monte Carlo steps `n_steps` in the main simulation	
	Choose the number of charge carriers for the desired charge carrier density and parameters for measurements	
	call `allocate_and_initialise_fields`	
	Randomly place the charge carriers in the system, double occupation of cells has to be prevented	
	call `calc_multiple_hopping_times` to initialise the hopping times of all randomly placed charge carriers	
	`i = 1 to n_thermalise`	
	call `perform_single_hop_radius`	
	`i = 1 to n_steps`	
	call `perform_single_hop_radius`	
	Perform measurements	
	Evaluate and illustrate measurements	
	call `deallocate_fields`	
	Deallocate rest of the fields that are not part of structures	

Figure 7.5: Structogram of a bulk simulation with our new update mechanism

Choose parameters for the simulation held by the structures `hopping_parameters`, `lattice_info`, `contact_info` and `energy_info`

Choose the cut-off radius r_c, the update radius $r_{up,cc}$ and $r_{up,inj}$ and the hopping radius $r_{hop,cc}$ and $r_{hop,inj}$

Choose the zero-field-energy barrier Δ, the energetic disorder σ, the electric field strength F and the relative permittivity ε_r

- call `get_layer_convergence` to evaluate the system size
- call `get_lattice_and_contact_info`
- call `get_update_info_sphere` for bulk hopping with $r_{up,cc}$
- call `get_update_info_sphere` for injection with $r_{up,inj}$
- call `get_nearest_neighbours`
- call `get_hopping_info_sphere`
- call `get_interaction_info`
- call `get_morphology`

Provide the function `randE` which assigns the randomly chosen energy levels

- call `get_energetic_landscape`
- call `add_external_potential`

Choose parameters for the Markov Chain Monte Carlo process like the number of thermalisation steps `n_thermalise` and the number of Monte Carlo steps `n_steps` in the main simulation

Choose the maximum number of charge carriers and parameters for measurements

- call `allocate_and_initialise_fields`
- call `calc_multiple_injection_times` to initialise the hopping times of all injection cells

`i = 1 to n_thermalise`

- call `perform_single_step_radius`

`i = 1 to n_steps`

- call `perform_single_step_radius`
- Perform measurements

Evaluate and illustrate measurements

- call `deallocate_fields`

Deallocate rest of the fields that are not part of structures

Figure 7.6: Structogram of an injection simulation with the new update mechanism