

BACHELOR

KMC modelling of light induced halide segregation in perovskite solar cells

van Belois, Allard

Award date:
2021

[Link to publication](#)

Disclaimer

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Bachelor Final Project



KMC modelling of light induced halide segregation in perovskite solar cells

June 14, 2021

Allard van Belois

1356852

Bachelor Applied Physics and Bachelor Applied Mathematics

Supervisors:

Prof. Dr. P.A. Bobbert (Applied Physics)

Dr. S. Tao (Applied Physics)

Dr. B. Baumeier (Applied Mathematics)

Z. Chen MSc (Applied Physics)

R.H.J. Gerritsen MSc (Applied Mathematics)

Eindhoven, June 14, 2021

Abstract

Halide perovskites are currently researched for solar energy applications due to their high efficiency and tunable band gap. This material, however, faces instability issues under illumination, as the mixed halides segregate. The segregation leads to iodine rich zones, which have a lower band gap. The segregation is assumed to be powered by this lowering of the band gap as the excited charge carriers funnelled to these low band gap regions reduce their free energy. The segregation is believed to be vacancy-mediated halide diffusion that is prone to segregation. In this thesis, a current kMC model and code used to simulate this halide segregation are looked at. The current code showed issues with speed as well as scaling for the amount of unit cells. The current kMC code runs systems of $12 \times 12 \times 12$ unit cells in the order of days and the simulation time scales quadratically with number of unit cells. The amount of memory needed to run the simulations also scales quadratically and has been problematic. In this thesis a new code is developed and shown to linearise the simulation time with respect to the number of sites. Furthermore, the code is able to simulate systems of size $20 \times 20 \times 20$ within half an hour, which is a huge improvement to the weeks that the simulation with the Python code would take. These results have been achieved by restructuring the code and parallelising parts of the calculation. The code and model are shown to perform most efficiently for showing segregation in regards to simulation time for low numbers of vacancies, as the amount of vacancies is shown not to correlate with the required steps for segregation to occur. However, unexpected results have been produced, as the iodine rich zones stopped growing in this model, while the experimental observations produce opposing results. It is hypothesised this behaviour is due to the small simulation box, however, this needs to be looked into before conclusions can be drawn from the model.

Contents

1	Introduction	4
1.1	Perovskite solar cells	4
1.2	Light-induced segregation	5
1.3	Modelling	6
1.4	Scope and contents of the thesis	6
2	Theory	8
2.1	Thermodynamics	8
2.2	Kinetic Monte Carlo	9
2.3	Random Number Generator	10
3	Model Set-Up	11
3.1	General Set-Up	11
3.2	Rate	11
3.3	Energy	11
3.4	Assumption justification	12
3.4.1	Coinciding events	12
3.4.2	Halide diffusion	12
3.4.3	Carrier influence	13
3.4.4	Memoryless processes	13
3.5	Parameters	13
3.5.1	Simulation steps	13
3.5.2	Size of the model	13
3.5.3	Vacancy concentrations	14
3.5.4	Energy parameters	14
4	Code	15
4.1	Python Code	15
4.2	C++ Code	17
4.2.1	Topology and next event list	18
4.2.2	Time	19
4.2.3	Parallelisation	19
4.2.4	Information Retention and Summing	19
4.3	Performance	19
4.3.1	Direct comparison	20
4.3.2	Order of algorithms	20
4.3.3	Test case	24
4.3.4	Further speed improvements	24
4.3.5	File export reduction	25
4.3.6	Set-up linearisation	25
4.3.7	Parallelisation	26
5	Modelling Results	27
5.1	Method and Data Analysis	27
5.1.1	Segregation measure	27
5.1.2	Visualisation	27
5.2	Band gap radius	29
5.3	Carrier density	30
5.4	Halide ratio	33
5.5	Vacancy density	33
6	Conclusion and outlook	36
6.1	Outlook	36

A	Table of parameters	40
B	Manual code	41

1 Introduction

The current energy transition from fossil fuels to renewable alternatives relies on the innovation of green technologies that make renewable energy more efficient [1]. Within solar energy technology, perovskite solar cells have shown promising results with a rapid efficiency increase from 15% in 2013 to 25% in 2020 and lower costs than the conventional SiO_2 based solar cells, which has a maximal efficiency of 27.6% [2, 3]. Furthermore, perovskites do not need rare earth metals to reach such high efficiencies, which makes them more widely available for the coming decades [2]. In this thesis a problem concerning segregation of the halides in mixed halide perovskite cells is investigated. In this section, the perovskite solar cells, the problem and the investigation method are discussed.

1.1 Perovskite solar cells

In perovskite solar cells the photo-active layer is the perovskite crystal. Perovskites are a group of crystals that have the general chemical composition of ABX_3 , where A is an organic or inorganic cation, B is a metal cation and X is a halide anion. Note that each component can be a mix throughout the crystal, i.e. A can be a mixture of methylammonium and caesium. In solar applications these metal-halide perovskites are studied for their efficiency. One of the advantages in metal-halide perovskites is the tunability of the band gap by compositional alloying at X sites. This makes mixed halide perovskites useful for the applications in tandem solar cells, which are solar cells with two stacked photoactive layers with different band gaps. In this research the tunable perovskite methylammonium lead bromine iodine (MAPBI) is modelled. The closely resembling methylammonium lead iodide (MAPI) is shown in Figure 1.

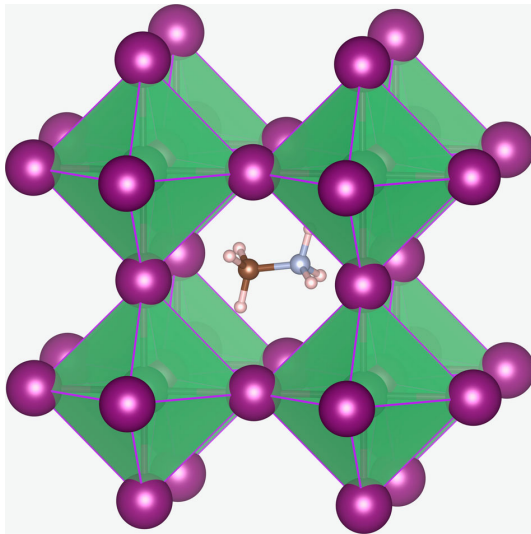


Figure 1: A schematic drawing of a methylammonium lead iodide (MAPI) crystal. In the centre the methylammonium ion (brown/grey) is depicted, surrounded by octahedrons of halide ions (purple) with lead ions (green) in the centre. For MAPBI, the halide ions are either iodine or bromine. The purple lines depict the nearest neighbour's relation between halides. Source: Eames et al. (2015) [4].

Perovskite solar cells convert sunlight into electric energy. The perovskite absorbs a photon, which creates an electron hole pair. These charge carriers are drawn to opposite sides by an anode and cathode respectively. The energy that an excited charge carrier has, depends on the band gap of the absorbent layer. The band gap also determines accordingly what wavelengths of light are absorbed. What wavelengths are absorbed is relevant for the efficiency of the cell [5]. Furthermore, highly efficient multi-junction cells are possible if the absorbing spectra of the multiple cells do not overlap starkly [6, 7]. As stated before MAPBI has a tunable band gap. The band gap of MAPBI increases continuously from 1.57 eV to 2.29 eV with an increasing concentration of bromine [8, 9, 10].

1.2 Light-induced segregation

An issue that mixed halide perovskites face is the segregation of halides under illumination. The band gap of the solar is only well-tuned if the halides are mixed homogeneously throughout the crystal. The segregation of the halides leads to domains with different band gaps, which leads to a lower efficiency and loss of the band gap tunability. When MAPBI is exposed to continuous illumination, it is observed that zones of high iodine concentration arise from the mixed phase [9, 11, 12, 13]. In Figure 2 a schematic of the light induced halide segregation in mixed halide perovskites is shown. The segregation can be observed in photo-luminescence experiments. A MAPBI crystal is illuminated and the light that is re-emitted is measured. The energy of this re-emitted light corresponds to the band gap of the regrouped photocarrier's location in the crystal. It is observed that suddenly all re-emitted light starkly red shifts, which strongly suggests that the iodine rich phases behave as carrier traps [11, 12, 13]. The photocarriers hop to the zones with lower band gaps and remain there, until they decay and emit light with energy equal to the local band gap. This so called funnelling effect causes these photocarriers to lower their free energy and make the solar cell lose efficiency. The segregation reverses when the cell is placed in the dark afterwards, which suggests that the phenomena is of a thermodynamic nature [14].

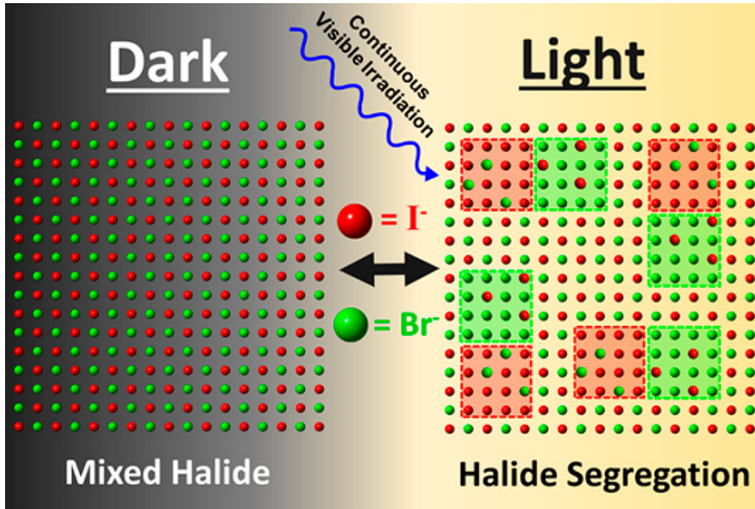


Figure 2: A schematic drawing of light induced halide segregation. The red zones in the light represent iodine rich regions where the band gap is lower, while the green zones represent bromine rich zones where the band gap is higher. The transition from the mixed phase in the dark to the segregated phase in the light is a reversible process. Source: Brennan et al. (2017) [15].

Light-induced segregation can be explained by the lowering of free energy of the charge carriers. When charge carriers get excited, they funnel to regions with lower band gaps within the crystal [11, 12]. The free energy is reduced by this funnelling behaviour. When iodine-rich region grow, the free energy of the system is further reduced, leading to phase separation. Other theories about the driving force of the growth of iodine rich zones include that illumination creates polaron-induced strain gradients [16], or that local electric fields are formed by electron-hole pairs at the surface [17] or in the film [18], or that the carrier formation has a strong gradient in the thickness [19]. These theories do not, however, account for the observed illumination threshold [12]. Therefore, in this research it is assumed that the lowering of free energy of the carriers are the driving force behind the segregation of the halides, since it does account for the illumination threshold [14].

The segregation mechanism is assumed to be vacancy-guided halide diffusion. Vacancies are empty halide sites throughout perovskite crystals that are caused by impurities [20, 21]. The neighbouring halide ions can move into this vacancy and as such, change the local concentration of bromine and iodine and band gap [20, 21]. In Figure 1, this is equivalent to a halide moving from one halide site to another site connected with a purple line. In dark conditions, the halides randomly diffuse through the crystal and a perfect mixture is obtained. In illuminated conditions, however, jumps that lower the local band gap, and as such lower the

free energy of the funnelling charge carriers, occur at a higher rate [9, 12]. The lowering of the local band gap is equivalent to increasing the local iodine concentration or lowering the local bromine concentration, hence this results in iodine and bromine segregation.

1.3 Modelling

To test whether the theory concerning vacancy-guided halide diffusion matches experiments, a kinetic Monte Carlo (kMC) model is used. The kMC model in this thesis is a continuation of previous models [16, 9, 22]. The model assumes that the segregation is driven by the lowering of free energy due to the funnelling behaviour of photocarriers. In this model it is assumed that these carriers instantaneously diffuse such that they follow Boltzmann statistics. The charge carrier density is sufficiently low for Boltzmann statistics to better describe the system than Fermi-Dirac. The lowering of free energy only occurs if the local band gap changes due to different concentrations of bromine and iodine ions. Therefore, in this model the fraction bromine of the halide ions within a sphere around each lead ion is taken and the band gap of an infinite crystal with equal concentration is assigned to the lead ion. The radius of this sphere is an arbitrary parameter, since no method of deriving it is known and matching to experiments has not been possible. The radius is assumed to be somewhere between 1.5 and 4 times the length of a unit cell. The current models simulate a cubic system of 12x12x12 unit cells with periodic boundary conditions to imitate an infinite crystal. The experimentally observed segregation takes in the order of seconds. Simulating this would currently take an enormous amount of computation time. For this reason, photocarrier density are taken at higher levels than real situations as it is hypothesised that this speeds up the segregation. The vacancy density in the model is also higher than realistic values, since otherwise no vacancies would be present in the simulation box, which are relatively small in comparison to solar cells.

The goal of this model is to match experimental observations and thermodynamic theory and make predictions about halide segregation. To achieve this goal, larger simulation boxes need to be simulated to prevent the periodic boundary conditions from influencing the results. The radius used for the total band gap is sometimes over a third of the box, which makes it unlikely or nearly impossible to have multiple iodine rich zones forming simultaneously. For this thesis a 20x20x20 unit cells simulation is set as a minimum goal. Besides periodic boundary conditions interfering, the model is random and therefore statistically sound conclusions require multiple runs. The current code, however, limits the possibilities to get rid of these interferences. The current code takes days to simulate a single 12x12x12 system. Furthermore, the time it takes to simulate, scales quadratically with the number of unit cells in the box. As an example, a single simulation of 20x20x20 unit cells would take months. Moreover, as running a single simulation of 12x12x12 takes a matter of days, producing statistically significant results by having multiple runs is computationally costly.

In this thesis the current model is reimplemented with a different approach, which reduces the computational costs of running the simulation and therefore speed up the simulation. Furthermore, the main calculations are parallelised such that a multithreaded simulation is possible. Moreover, the code is rewritten in C++ instead of Python to enhance the performance further. It is attempted to linearise the simulation time as a function of the amount of unit cells to make simulation of larger boxes possible within reasonable amounts of time.

1.4 Scope and contents of the thesis

In this thesis the kMC modelling of light induced halide segregation in MAPBI is discussed. The goal is to create a code that can simulate systems of 20x20x20 unit cells within a reasonable time, i.e. an hour, and that scales well with increasing number of unit cells. Furthermore, the behaviour of the model for different parameters, like the illumination, local band gap radius and number of vacancies, with respect to the experimental observations and thermodynamic theory is also looked into. To achieve these goals, first in Section 2 the thermodynamic theory of thermodynamics of the segregation is discussed, as well as the mathematical background of kMC modelling. Next in Section 3 the model is elaborated, along with the assumptions and all governing equations. The parameters of the model are explained as well with their respective range of values. In Section 4 the current implementation in Python is elaborated, then the C++ code that has been written for this thesis is discussed, for which the most important improvements with respect to the Python code are highlighted. The performance on speed and scaling of the two implementations is then compared and discussed. In Section 5 first the method of data analysis is discussed, as well as the

visualisation techniques. Then some results of the model are discussed to verify the viability of the model as well as a recommendation on how the model should be used. It is attempted to find a relation between the parameters of the model and segregation. Some future research suggestions are given as well. Finally in Section 6 the value and improvements of the model is summarised and final recommendations on the following research are given.

2 Theory

2.1 Thermodynamics

The behaviour of halide segregation is predicted by a unified thermodynamical theory [14]. In this theory it is shown that, in accordance to experimental results, MAPBI is stable in the dark at room temperature. The experimentally observed threshold illumination level is shown to be in accordance to the theory, where a threshold photocarrier density is shown to exist. In the unified theory the behaviour of the perovskite is seen as the joint effect of two driving mechanisms. On the one hand, there is the segregating effect caused by the free energy lowering demixing. On the other, there is the mixing effect of entropy which also lowers the free energy. By determining an equation for the free energy and using the equation for the mixing free energy, the spinodal and binodals can be determined in the concentration of bromine and temperature phase diagram. With these known, the stable, metastable and unstable regions in the phase diagram can be determined. The influence of the illumination level can clearly be seen in Figure 3. With illumination, high concentrations of bromine become metastable at room temperature. The lower limit for what concentration of bromine is stable decreases with increasing illumination. It is shown that at illumination equivalent to one sun, concentrations above roughly 50% are unstable. The theory also suggests the existence of a triple point, at which three different phases can coexist. This triple point has yet to be experimentally observed. The model described in this thesis uses the same assumptions about the mechanism of segregation. It is therefore interesting to see whether the model produces corresponding results for what phases are stable. This not done in this thesis, but could be done by using photocarrier densities equivalent to those from Figure 3 with different bromine concentrations and temperatures.

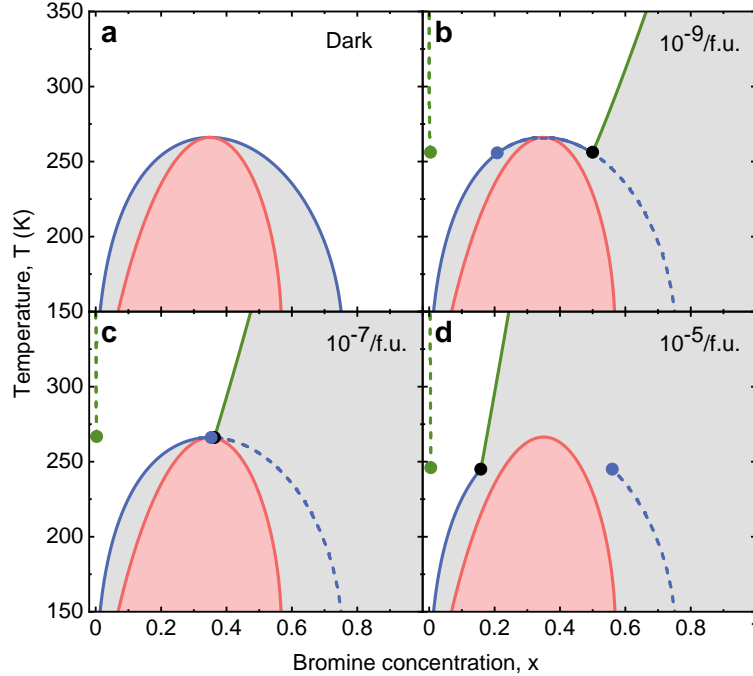


Figure 3: The temperature bromine concentration phase diagrams for MAPBI at different levels of photocarrier densities $n=0$ (dark), 10^{-9} , 10^{-7} , and 10^{-5} per formula unit (f.u.). Red lines: spinodals separating the metastable (grey) and unstable (pink) regions. Full blue and green lines: binodals separating the stable (white) and metastable regions, with the blue (green) lines indicating the compositional (light-induced) binodals. When entering the metastable region by crossing the compositional (light-induced) binodals, nucleation of a phase with a Br concentration indicated by the dashed blue (green) lines becomes favourable. The dots indicate the possible coexistence of three phases: the parent phase (black dots) and two types of nucleated phases with different Br concentration (blue and green dots). Source figure and description: Chen et al. (2021) [14].

2.2 Kinetic Monte Carlo

The system is modelled with a kinetic Monte Carlo (kMC) model, in which the transitions within the system are randomly sampled. In this halide segregation case all transitions are hops of halide ions to a vacancy. For this case it is assumed that the system can be modelled as a continuous time Markov chain. For a continuous time Markov chain it holds that

$$\mathbb{P}((X_n, t_n)|(X_{n-1}, t_{n-1}), \dots, (X_0, t_0)) = \mathbb{P}((X_n, t_n)|(X_{n-1}, t_{n-1})), \quad (1)$$

where X_n and t_n represent the state of the system at time step $n \in \mathbb{N}$. It means that the probability of each hop occurring and the time it takes for a hop to occur only depends on the state the system is currently in. Since each hop occurring by itself is memoryless, the arrival time of each hop behaves as an exponentially distributed random variable. The cumulative distribution function of an exponentially distributed random variable for some hop i is given by

$$F(t; \lambda) = \begin{cases} 1 - \exp(-\lambda t) & t \geq 0 \\ 0 & t < 0, \end{cases} \quad (2)$$

where λ is called the rate of the process and t is the time since the last hop. In a system where n independent hops could occur with rates λ_i , $i = 1, \dots, n$, the time until one hop occurs has the cumulative distribution function as shown in Equation (2) with rate $\lambda_{\text{tot}} = \sum_{i=1}^n \lambda_i$. This is because

$$\begin{aligned} \mathbb{P}(\text{A hop has occurred before time } t) &= 1 - \mathbb{P}(\text{No hop has occurred before time } t) \\ &= 1 - \prod_{i=1}^n (1 - F(t; \lambda_i)) \\ &= 1 - \prod_{i=1}^n \exp(-\lambda_i t) \\ &= 1 - \exp(-\lambda_{\text{tot}} t). \end{aligned} \quad (3)$$

So the time for a hop behaves as an exponential random variable with as rate the sum of the rates of all possible hops. Now for the probability that it was event k occurring at time t when it is known that a hop occurred at t , the density functions need to be considered. The density function of an exponential random variable with rate λ is given by

$$f(t; \lambda) = \begin{cases} \lambda \exp(-\lambda t) & t \geq 0 \\ 0 & t < 0. \end{cases} \quad (4)$$

The probability that hop k occurred at time t given that the first hop occurs at time t is, by Bayes' theorem, given by

$$\begin{aligned} \mathbb{P}(\text{hop } k \text{ occurs at time } t | \text{The first hop occurs at time } t) &= \frac{\exp(-(\lambda_{\text{tot}} - \lambda_k)t) \cdot f(t; \lambda_k)}{f(t; \lambda_{\text{tot}})} \\ &= \frac{\exp(-(\lambda_{\text{tot}} - \lambda_k)t) \cdot \lambda_k \exp(-\lambda_k t)}{\lambda_{\text{tot}} \exp(-\lambda_{\text{tot}} t)} \\ &= \frac{\lambda_k}{\lambda_{\text{tot}}}. \end{aligned} \quad (5)$$

Note that the probability that no other hop occurs before and at time t is equal to the product of one minus the cumulative function for all other processes, which is given by

$$(1 - F(t; \lambda_k)) \cdot \prod_{i=1}^N (1 - F(t; \lambda_i)) = \exp(-(\lambda_{\text{tot}} - \lambda_k) \cdot t) \quad (6)$$

So by combining the exponential distribution with rate λ_{tot} to draw a time and the probabilities per event as given by Equation (5), the model behaves the same as if all events were tracked as separate random variables. This fact is useful for coding the model, as will be explained in Section 4.

2.3 Random Number Generator

The hops in the model are probabilistic. To implement such probabilities into a functioning model a random number generator (RNG) is needed. In essence an RNG produces a sequence of numbers that cannot be predicted any better than by probability.

In practice there are two kinds of RNGs. The first kind is a true RNG that draws its random number by using random processes in nature. The decay of particles or the measuring of particles' quantum states are examples of random processes in nature that can be used to draw a random number. For these examples it would be measuring the time for the decay to take place or attaching a number to a certain state. The drawn number can then be transformed to fit the desired probability density range. For example, the decaying particle follows an exponential distribution and its density can be transformed to fit a uniform density [23].

The other kind of RNG is a pseudorandom number generator (PRNG). These generators do not use a true random process, but an algorithm to determine the numbers of the sequence. This means that the sequence is deterministic, hence the pseudorandomness. A good PRNG has two important properties. The first property is that the dependence of the next number in the sequence is minimised. The second property is that the sequence does not repeat quickly, so the period must be long. A PRNG uses a seed, usually a number, to determine the sequence. The randomness of the PRNG can be improved by using a true RNG to draw the seed number. The PRNG can be designed such that the range fits the requirements, for example a 64 bit PRNG can be used as a uniform distribution from 0 to 1 with in total 2^{64} different outcomes. This sufficiently approximates continuous number drawing for most cases.

Since high speed simulating is one of the main objectives of this thesis and the code will be run on regular servers or computers, it has been opted to use a PRNG. The Mersenne twister is the most used PRNG for kMC due to its humongous period of $2^{19937} - 1$, a Mersenne prime. Its algorithm is also implemented in C++, which makes it easily available and widely applicable. When it was first implemented into C, it was four time faster than the standard `random()` function of C [24]. However, there are some things to consider when using this PRNG. If two (P)RNGs are required, which is often the case for Monte Carlo simulations, a protocol needs to be followed in order to obtain sufficiently independent random numbers [24]. The seed that is used should also be drawn randomly if statistically sound measurements are to be obtained from the simulations. It is good practice to use a true RNG to generate such a seed [25]. Another reason a PRNG is used is for repeatability. When the same seed and parameters are used the code runs the same every time. This can be useful as a multitude of runs can be run with saving minimal data and only the runs that segregate or show interesting outcomes are repeated. During the repetition of the simulation the configuration is saved more often. The possibility to exactly repeat simulations is also useful as the results can be reviewed and verified.

3 Model Set-Up

In this section the model is set up. The underlying mechanics are elaborated upon in Section 3.1, then the assumptions that are made in the model are elaborated and justified in Section 3.4, and finally the parameters of the model are listed and some system size indications are given in Section 3.5

3.1 General Set-Up

The model describes the halide diffusion in a crystal of MAPBI. A schematic drawing of MAPBI can be seen in Figure 1. The halides are depicted by purple spheres, for which the nearest neighbours are connected by purple lines. The modelled crystal is assumed to be a cuboid of unit cells with periodic boundary conditions. These periodic boundary conditions allow for the model to behave more like a large system, such as a solar cell, while keeping the computational costs low.

In this model of $\text{MAPb}(\text{I}_{1-x}\text{Br}_x)_{3(1-y)}$, the I and Br ions move around, while the methylammonium and lead ions are assumed to be stationary. In this model y is defined as the concentration of vacancies by assuming a certain number of halide sites to be empty. The vacancies play a crucial role in transporting halide ions. The halide ions at the nearest neighbour sites can hop into the vacant halide site. This creates a new vacancy at the previous position of the moved halide, which continues the process. Such hops change the energy of the system. In Section 3.2 and Section 3.3 the rates and energy changes of the hops are elaborated.

3.2 Rate

It is assumed that the hops' arrival times behave as a Poisson point process. The rate of each hop depends on the energy difference that the hop causes, where free-energy-lowering hops are assumed to have higher rates. As such the rate for each hop between vacancy at halide site h and a halide ion at halide site j is given by

$$k_{h,j} = k_0 \exp\left(-\frac{\Delta E_{\text{tot}} + E_{\text{bound}}}{k_b T}\right), \quad (7)$$

where k_0 (s^{-1}) is a tunable prefactor rate, ΔE_{tot} (eV) is the difference in total energy, which is defined in Equation (8) and E_{bound} (eV) is the energy barrier of each halide ion which needs to be overcome for a hop to occur, which is assumed to be 0.25 eV for both iodine and bromine. In reality, the energy barrier of iodine is higher than that of bromine, but exact values are not known [26]. The (free) energy difference of the system is defined as

$$\Delta E_{\text{tot}} = E_{\text{tot, after}} - E_{\text{tot, before}}, \quad (8)$$

where $E_{\text{tot, after}}$ and $E_{\text{tot, before}}$ are the energies of the state after and before the hop respectively. Equation (10) in Section 3.3 gives the free energy of the system and is elaborated in that section.

To determine the time before a single hop occurs, the rates of all possible hops are summed. Hence, the total rate of the next hop k_{tot} is given by the sum of the rates of all possible next events:

$$k_{\text{tot}} = \sum_{(h,j)} k_{h,j}, \quad (9)$$

where (h, j) is the index pair of the vacancy halide pairs for which a swap can occur. With this total rate the time passed before the next event passes is, as determined in Section 2.2, exponentially distributed with rate k_{tot} . The expected value of the time is $\frac{1}{k_{\text{tot}}}$, so when more vacancies are present the time before the next event is expected to reduce. To model for roughly the same time, the number of steps needs to scale with the number of vacancies, as the time per event scales roughly with the reciprocal of the number of vacancies.

3.3 Energy

The rate depends on the (free) energy of the system, which is determined by the charge carriers' energy. Under illumination charge carriers form in the perovskite crystal, which is the light active layer in a perovskite based solar cell. This means that charge carriers get excited by the incoming light. These charge carriers have free energy and, as most systems in nature, the system tends to lower the free energy. The free energy of a charge carrier depends on their positions and the band gap of the system. Note that charge carriers usually behave as described by Fermi-Dirac statistics. In this model, however, it is assumed that the charge

carrier density is low enough for the charge carriers to not interact. Therefore, the system is described by Boltzmann statistics. With n_{car} the number of charge carriers in the system, the total energy is then given by

$$E_{\text{tot}} = n_{\text{car}} \sum_{i=1}^N E_{\text{gap},i} \cdot w_i, \quad (10)$$

where N is the number of lead ions (and therefore unit cells), $E_{\text{gap},i}$ is the band gap at lead site i , which will be fully defined below, and w_i is the Boltzmann weight of the local band gap at site i , which is given by

$$w_i = \frac{\exp(-E_{\text{gap},i}/(k_{\text{B}}T))}{\sum_{j=1}^N \exp(-E_{\text{gap},j}/(k_{\text{B}}T))}, \quad (11)$$

where k_{B} is the Boltzmann constant. From Equation (11), it can be seen that sites with smaller band gaps have a higher Boltzmann weight. This corresponds to a larger probability that a charge carrier is in the state corresponding to the band gap. Therefore, lower band gaps have more influence for the total energy.

The band gap of a certain position is based around the lead ion. To find the band gap for each lead site, it is assumed that a local band gap exists for which the lead ion is the centre. This local band depends on the concentrations of bromine and iodine around the lead site. The concentration is taken of a sphere with radius r_{eg} around each lead ion, the so called energy ball. The radius is in this thesis an arbitrary parameter, as it is not yet known what value best corresponds to reality. In this thesis it has been researched what influence the value of r_{eg} has on segregation with respect to the size of the iodine rich zones. Vacancies have been shown to have no effect on the local band gap [9], and as such the concentration of bromine in the sphere centred around lead ion i is defined as

$$c_i = \frac{N_{\text{Br},i}}{N_{\text{Br},i} + N_{\text{I},i}}, \quad (12)$$

where N_{Br} and N_{I} are the number of bromine and iodine ions in the sphere, respectively. It has been assumed that the band gap at the centre of the sphere is equal to the band gap of a homogeneous infinite crystal with equal concentration. The local band gap for each local band gap i is then given by [22, 27]

$$E_{\text{gap},i} = 1.57 + 0.39c_i + 0.33c_i^2, \quad (13)$$

where c_i is the concentration of Br for each local band gap i .

3.4 Assumption justification

The model makes some non-trivial assumptions. In this section these assumptions are justified and elaborated. The necessity of these assumptions is discussed to shed light on their influence and for reasonable future improvements of the model regarding these assumptions.

3.4.1 Coinciding events

In the model it is assumed that no two hops occur at the same time. Since the time in reality is continuous, the probability of two hops occurring simultaneously is zero for a finite amount of finite rate possible hops [28]. This assumption also only holds if the hop can be considered instantaneous. This is assumed in the process of halide diffusion and fits the experimental observations as elaborated below.

3.4.2 Halide diffusion

The assumed theory of how ions diffuse through the crystal is that the halide ions hop to an adjacent vacancy. This is supported by the measured barrier energies of iodine and bromine in the crystal [15]. The hopping is nearly instantaneous, opposite to interstitial hopping, which is shown to be less fitting to the measurements than vacancy hopping [4, 29]. It also means that only halides adjacent to the vacancy site can diffuse into the vacancy site. This is supported by the significantly higher bound energy for all halides at a larger distance from the vacancy than the nearest neighbour distance [30].

3.4.3 Carrier influence

The whole model is built around the assumption that the potential lowering of free energy for the charge carriers influences the diffusion of halide ions. It is known that charge carriers move to low band gap regions, which in this model corresponds to iodine rich zones [27, 11, 12]. This moving of photo carriers to low band gap zones is observed in photoluminescence experiments. It is assumed that the initial funnelling is instantaneous. It is observed that the funnelling is, sufficiently, rapid [12], which justifies this assumption. This assumption rids the model of the process of the production and funnelling of carriers, which is complex. The model also assumes that the carriers are produced homogeneously throughout the crystal, which is reasonable for a thin crystal.

3.4.4 Memoryless processes

The model is based on a point-based Poisson process. In such processes the rates are only dependent on the current situation, which means that the process is memoryless. This assumption can be justified by the fact that no information can be saved in the state of the system about previous states. Therefore, the system must be memoryless. The rate of these processes are based on the Arrhenius equation, which is given by

$$k = A \exp(-E_{\text{act}}/k_{\text{B}}T), \quad (14)$$

where k is the rate of the process, A is the base rate, E_{act} the activation energy for the process. The Arrhenius equation is an equation that is used for the rate of chemical reactions. Since the moving of ions is also a chemical reaction, the use of the Arrhenius equation is warranted.

3.5 Parameters

In this section all parameters of the model are listed and elaborated. The parameters are subdivided in simulation, size, halide and energy parameters. An overview table of the parameters can be found in Appendix A.

3.5.1 Simulation steps

The simulation specific parameters are the maximal amount of steps, N_{step} , and, optionally, the maximal amount of time simulated, t_{max} . The number of steps before segregation occurs scales with the number of sites, as more halide ions can be diffused before iodine rich zones form. The maximal time concerns the time that is modelled. It is a useful parameter if the level of clustering after a fixed time needs to be studied. Note that the prefactor k_0 in Equation (7) scales with the time passed in the model. In this model k_0 is assumed to be in the order of magnitude of phonon frequencies, which have order 10^{12} s^{-1} [31]. The maximal number of steps differs for different model sizes. For an 8x8x8 or 12x12x12 model the maximal number of steps is in the order of 10^5 steps and for a 20x20x20 model the maximal number of steps is in the order of 10^6 .

3.5.2 Size of the model

The model represents a crystal, which consists of repeating unit cells. The number of unit cells in the three spatial directions are n_x , n_y and n_z respectively. The periodic boundary conditions are such that the model behaves more like an infinite crystal, but with a small number of unit cells in a direction, side effects could occur. This is why in this research the size of the model is increased with respect to the tested 12x12x12 configuration of Jaysankar et al. (2018) [9] to 20x20x20. The total number of halide sites is equal to $3 \cdot n_x \cdot n_y \cdot n_z$, as there are three halide sites in each unit cell, and the number of lead sites is equal to $n_x \cdot n_y \cdot n_z$. Some relevant examples are given in Table 1 below.

Table 1: Some relevant sizes and the number of unit cells and sites.

Size: $n_x \times n_y \times n_z$	Unit cells and lead sites	Halides sites
4x4x4	64	192
6x6x6	216	648
8x8x8	512	1536
12x12x12	1728	5184
16x16x16	4096	12288
20x20x20	8000	24000

This model is made for MAPBI, which has cubic conventional cells. The model, however, could be used for different crystals which have different sizes. In this research the edge length of the unit cell is 6.28 Å in the x-, y- and z-direction [32]. This distance is referred to as a 'unit' for simplicity's sake. Note that the distance between two nearest neighbours for halides is equal to $\frac{1}{\sqrt{2}}$ unit, so 4.44 Å. Only halide sites at this distance, called r_{cut} , can interact for a swap in this model.

3.5.3 Vacancy concentrations

The concentrations of the halides can differ between crystals, since the band gap is tuned by altering the concentration. In this model the number of bromine and iodine ions, n_{Br} and n_I respectively, can be adjusted. The number of vacancies n_{vac} is also an important parameter, since only with vacancies present a region can alter its arrangement. The number of bromine, iodine and vacancies must add up to the number of sites in the model. In the simulations the number of vacancies in this model are taken higher than realistic conditions have as realistic vacancy densities would correspond to zero or one vacancy in an 8x8x8 box. Therefore, 0.5% up to, for some extreme case, 40% of the halide sites are taken to be a vacancy, the exact values for each simulation are indicated in the figure description.

3.5.4 Energy parameters

The rate of each swap is based on the energy difference and barrier energy. The barrier energy E_b is the energy required for an ion to jump. In this research it is assumed that both iodine and bromine have a barrier energy of 0.25 eV. It is known that bromine has a lower barrier energy than iodine, but no exact values are known [26]. Therefore, they are assumed to have the same value for simplicity's sake.

The difference of energy between and after a swap scales linearly with the number of light-generated charge carriers n_{car} . This value depends on the illumination strength, so in dark conditions it would be zero. It also scales with the volume of the box. The charge carrier density is, just like the vacancy concentration, taken significantly larger than realistic situations to make the effect of clustering stronger and reduce computational costs to see effects. The number of charge carriers per unit cell used in this model is between 10^{-3} and 10^{-1} , while realistically the number is somewhere between 10^{-9} and 10^{-4} [22].

The band gap is determined based on the local concentration of bromine. The region for which the concentration is taken is a sphere with radius r_{eg} centred on the lead ion. This value is between roughly 1.5 and 4 times the distance between nearest halide neighbours, so roughly between 6 Å and 20 Å.

The number of charge carriers in each energy level depend on temperature T , as well as the rate of each swap. The perovskite cells are meant for use in atmospheric conditions, so a temperature of 300 K is used. Lower temperatures would slow down the swaps and therefore the clustering effect.

4 Code

The model as described in Section 3 had been implemented in a Python script by combining pieces of code from different projects. The code produced the predicted results and hence it was useful as a tool in research. The code was however not optimised, which limited the practicality in simulating larger systems or performing multiple simulations of smaller systems to produce statistically significant results. The optimisation in this research is mainly focused on improving the speed of the code. The model has been rewritten in C++ along with some modifications to further speed up the model. C++ has been chosen because of its high performance when it comes to speed, especially in comparison with Python. The main difference between the two is that Python is a dynamically typed, interpreted language, while C++ is a statically typed, compiled language. Dynamically typed means that the type of variable does not need to be declared before executing, while statically typed means that the variable type has to be declared. A compiled language means that before executing the file, a binary file is made by the compiler. The binary file can directly be executed by the processors without an interpreter programme in between. With the interpreted language, however, the code is interpreted while it is run. Therefore, no efficiency algorithms can be applied, which can be done while compiling. Python is therefore an easy language to write in and good for executing simple tasks, while C++ is better suited for computationally heavy tasks, like a large kMC model, but it takes more time and effort to program. In this section, the Python code and C++ code are explained with the use of a flow chart. Next, the improvements that have been implemented in the C++ code are elaborated. Finally, the performance of the Python and C++ code are compared, along with some suggestions for future improvements.

4.1 Python Code

In this section the Python code is explained with the use of the flow chart depicted in Figure 4.

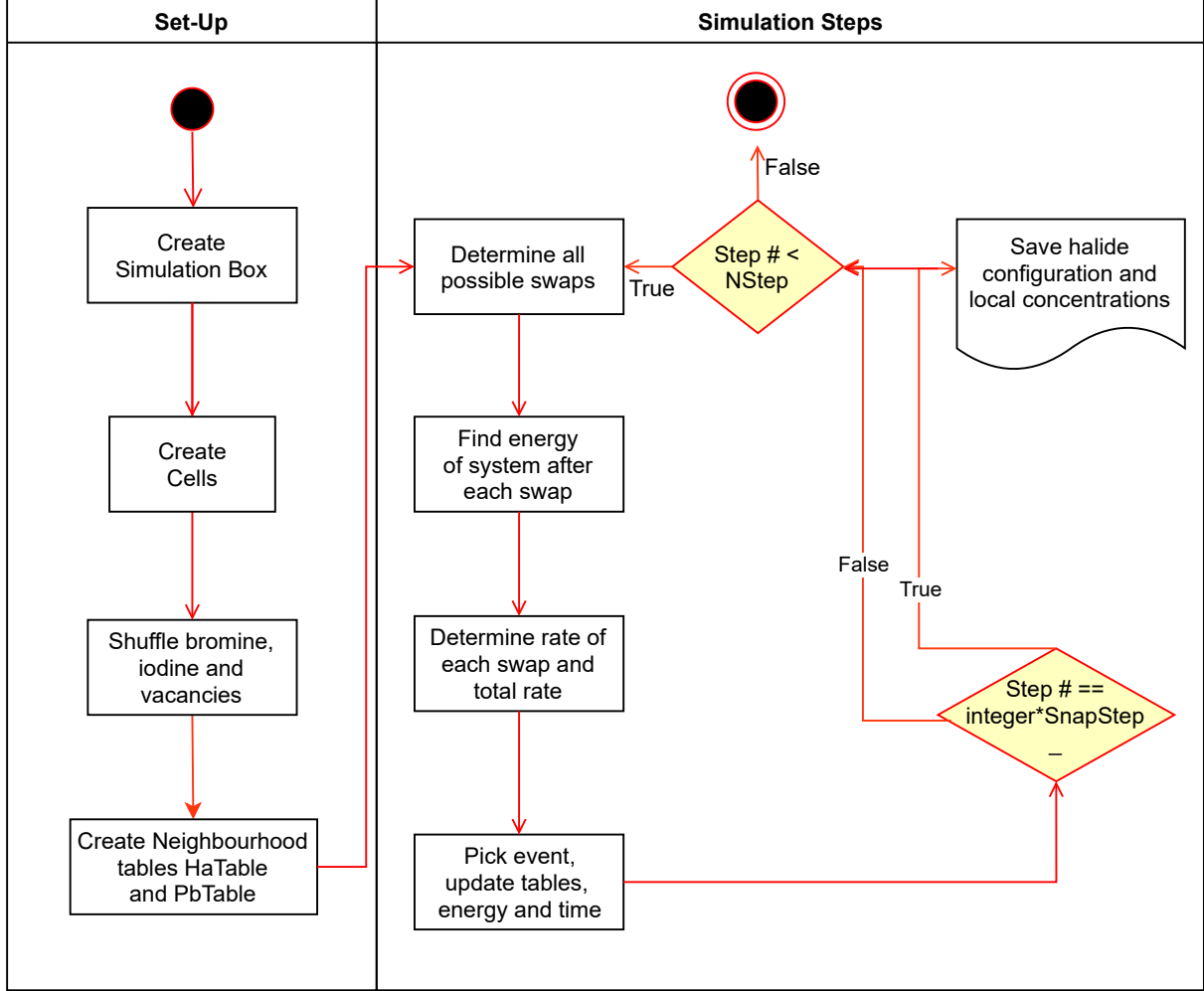


Figure 4: Flow chart of the Python code. The code is subdivided into the set-up and the simulation loop, of which the last one is where the actual hopping and data collection occurs.

The code has two distinct sections, the set-up, which is run once, and the simulation steps, which is run N_{step} times in a loop. In the set-up, first the simulation box is created. The simulation box contains all coordinates of the lead and halide sites, which are saved in two separate lists. Next, so called cells are created. These divide up the volume of the simulation box into smaller blocks. By knowing that the distance between two blocks is larger than r_{eg} , all combinations of halide and lead sites from both boxes do not need to be considered when determining whether they are within r_{eg} of each other, which is relevant for determining the local band gap, or r_{cut} , which is for possible sites that can be hopped to from each site.

The halide coordinate list is then shuffled and added to the box. The first n_{vac} halides sites are the vacancies, then the next n_{Br} bromine and finally n_{I} iodine. The halide type for each index remains constant throughout the simulation, while the coordinates change. Using the cells, the halide and lead tables are constructed. In the halide table (**HaTable**) for each halide site the other halide sites within r_{cut} , the maximal hopping distance, are marked. Note that on both the horizontal and vertical all halide sites are given, so a square, symmetric table is made. The combinations that satisfy the distance criterium are marked with a 1, the others with a 0. The amount of information that is retained in this step is large, which is most likely part of the reason of the performance of the system. Similarly, in the lead table (**PbTable**) for each lead site the halide sites within r_{eg} are marked. The energy of the system is then calculated by calculating the band gap for each site by counting the number of bromine and iodine close to each lead site and using Equation 13. This concludes the set-up phase of the simulation.

Next, the actual simulation starts. The simulation consists of a loop that is repeated N_{step} times. In this loop first the possible hops for the vacancies are determined by getting the information from the **HaTable** for the vacancy sites. Note that a vacancy cannot hop to another vacancy site, so these combinations are ignored. For each possible hop the hop is performed in the table, the energy is recalculated and the hop is reversed. Next, the rate for each hop is determined and the total rate is calculated. The event is picked and the energy, time and tables are updated. The table gets updated by switching the information of the relevant vacancy column with the relevant halide column and same for the rows. For every **SnapStep** step the state of each halide site is added to a file.

4.2 C++ Code

In this section the C++ code is explained with the use of the flow chart depicted in Figure 5. The most significant differences between the Python code and the C++ code are listed.

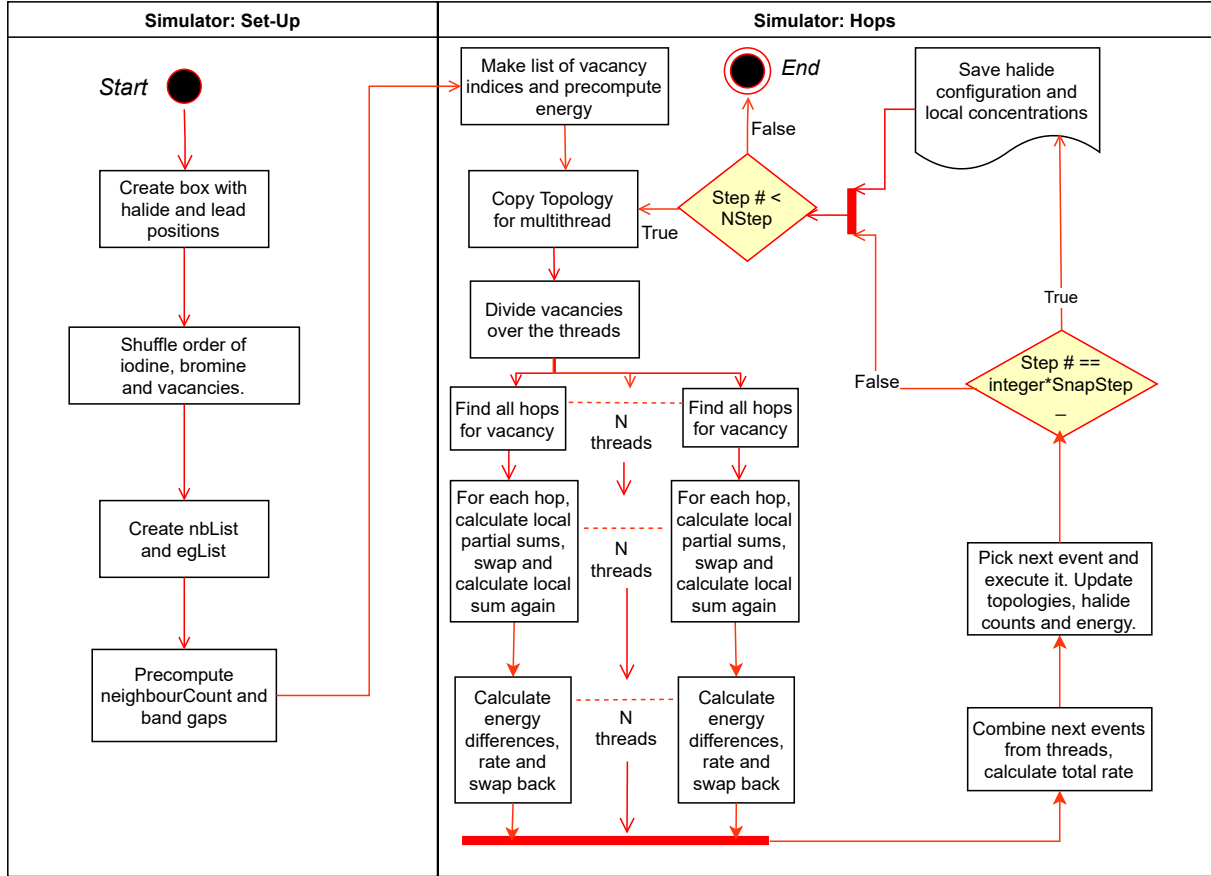


Figure 5: Flow chart of the C++ code. Like the Python code, the code is subdivided into a set-up and simulation loop.

The C++ code is based around the indices for each site. Every halide and every lead site has a fixed index by which it can easily be identified. Information about the sites can be saved in a vector such that the index of the site matches the position within the vector.

To start, the simulation box is created. Similar to the Python code, the box is rectangular and has $n_x n_y n_z$ lead sites and $3n_x n_y n_z$ halide sites. The coordinates of each site are saved in two vectors, the **halidePositions** and **leadPositions** vector.

In the next block, a vector with iodine, bromine and vacancy elements is created, the **halideTypes** vector. The number of iodine, bromine and vacancies in this vector is determined by the input of the model, this way every site has a unique type. This vector is shuffled to create a random starting position. Note that the

first element of the `halidePositions` has the type indicated by the first element of the `halideTypes` vector and so on.

The following block indicates the making of four neighbour lists, the `nbList` and three so called `egLists`. A neighbour list is a vector that contains a vector for each site with all indices of sites that are within a fixed distance and are of the right type, so lead or halide. The first neighbour list is the `nbList`. This list contains for each halide site the indices of the nearest halide site neighbours and thus the sites with which a vacancy at a site can swap. The second and third neighbour lists are the `energyHaPbList` and the `energyPbHaList`. The `energyHaPbList` contains the list of indices of lead sites that are within the band gap radius for each halide site and the `energyPbHaList` the exact inverse, for every lead site the indices of the halide sites. These are used to count the bromine and iodine ions to calculate the local bromine concentration for each lead ion. The last neighbour list is the `energyHaHaList` and does not serve a purpose within the programme itself, but is used to make the visualisation of the system more easy. The `energyHaHaList` contains for each halide site the indices of halide sites within the band gap radius from it. It is a good indication for the local band gap for nearby lead sites and shows the formation of iodine rich regions.

Making use of the `egLists`, in the next step the `neighbourCount` is computed. This vector keeps track of the number of iodine, bromine and vacancies at the sites within the band gap radius r_{eg} of each lead site. The band gap and the corresponding exponent are calculated as well and saved in another vector. This step concludes the set-up of the code.

Next, the code enters the run phase. Before the transitions start, the indices of the vacancies are saved in the `positions` vector. This way the possible transition can easily be determined by looking only at the neighbours of the vacancies. The `positions` vector gets updated after each hop. The energy of the system is computed by using Equation 10 with the use of the precomputed band gaps and exponents.

Next, the code enters the simulation loop. This loop is performed for N_{step} steps. It starts with copying the topology for the different threads. The different threads each calculate the rate for the possible hops for the vacancies, but to do so, they temporarily alter the topology. To prevent mix-ups when multiple threads alter the topology and thus get wrong energies, each thread gets its own topology copy. When each thread has received a topology copy, the vacancies get as evenly as possible divided over the threads.

The possible hops for each vacancy are determined by looking whether its neighbours, saved in the `nbList`, are not another vacancy. If that is not the case, the energy difference of the hop is determined. To prevent the recalculation of the band gap for unaffected regions and to prevent calculating a sum over each site when most parts of the sum do not change, only for the regions within r_{eg} of the two halide sites of the swap are taken into account in the calculation. This is an improvement over the Python code, where the band gaps, exponential and sums are recalculated for the whole crystal. The rate of each hop is calculated and saved with the energy difference and indices of the halide sites in question. This information of the different threads are then recombined to calculate the total rate. With this information, the hop that is going to occur is determined using the PRNG. Then all information of the system is updated, like the topology, time and energy.

The configuration of the system is saved for every `SnapStep` step, for example every thousandth step. For each halide site, the halide type, coordinates and bromine and iodine number of its energy ball is saved in a comma separated file (csv). After the information is saved, the loop starts over again until the desired number of hops has been done.

4.2.1 Topology and next event list

The Python code differs from the C++ code in the way the topology is treated. In the Python code the nearest neighbours are saved in a table. The halide sites are ordered such that the vacancies are always the first in the list. In the table for all halides is stated whether every other halide is a nearest neighbour and when a transition occurs, all this information has to be transferred to the new table. The memory required for the table scaled by n_{cell}^2 , where n_{cell} (-) is the number of unit cells in the simulation. In the C++ code the sites are labelled by an index by putting them in a vector, the C++ array equivalent, and not reordering the list while running the simulation. The information of what each halide site contains is also stored in a vector of equal length, such that the same index for both vectors corresponds to the same site. The nearest neighbours of all sites are stored in a vector of vectors, such that the vector at the index has the indices of its nearest neighbours. This is done in the set-up and does not change during the simulation loop. The information that was saved in the table is saved in the C++ code in these vectors that scale linearly with

n_{cell} . Note that the required memory for the Python code does not depend on r_{eg} , while the number of entries in the `egList` does scale roughly cubic with r_{eg} . However, the information retained is always lower in the C++ code, regardless of r_{eg} , and r_{eg} does not necessarily scale with the size of the simulation box, especially for large simulations, so the C++ code uses less memory even though it scales with a higher factor.

4.2.2 Time

In the Python code and the other previous models the time each step in the model takes is the reciprocal of the total rate [22, 9]. This is the expected value of the exponential distribution with the rate sum as rate. Over a large number of steps the use of the expected value is a reasonable simplification. However, since the rate might change over time in the model, the simplification might no longer be accurate. Therefore it has been opted to use the RNG to draw a time from an exponential distribution with the rate sum as rate. The drawing of the time using the RNG takes more time than taking the reciprocal of the rate sum, so if the exact time passed in the simulation is not vital and speed of simulation is, taking the expected value could be useful.

4.2.3 Parallelisation

In the model multiple vacancies are present and all have different possible swaps. In the Python code all vacancies were done in series, such that the system has to wait on the previous vacancy calculation to start with the next. In the C++ code the program divides the vacancies over multiple threads, which each calculate the rates for its possible swaps, after which the possible swaps are recombined and the next event is determined. Each thread is run on a separate core of the processor. The system can therefore speed up with a factor of little less than the number of threads, as picking the next event only occurs at a single processor. This factor is limited by Amdahl's law, which states that a programme can only speed up to a certain factor due to not properly parallelisable parts of the programme. The parallelisation requires some more memory as the rate of each step depends on the whole topology and while calculating the rate, the topology is temporarily adjusted. If multiple threads adjust the same topology, errors will occur. The topology is therefore copied for each thread and altered after the final event. These small extra steps result in a slightly lower factor than the number of threads.

4.2.4 Information Retention and Summing

To determine the rate of a swap for each lead site the band gap and corresponding exponent need to be known. In the Python code the bromine concentration, band gap and its corresponding exponent are recalculated for all lead sites each step. When a swap occurs at distances larger than the band gap radius from both relevant halide sites, these values do not change. In the C++ code it is therefore implemented that only for the lead sites that are close enough to the swap these values are recalculated. As can be seen in Equation (11) the sum of all the band gap exponents needs to be calculated. Large portions of the sum do not change after a swap. Therefore the sum of the relevant region is calculated before and after the swap and the difference is subtracted from the previous sum. The value of the new sum is also added to the Next Event List such that the information remains available and does not need to be recalculated when this swap, or event, is actually performed. Something similar holds for the sum in Equation (10), as the only value that changes for lead sites sufficiently far away from a swap is the sum in the denominator. Therefore the value of the band gap times the exponent is also saved and the sum is altered like that of the sum of exponents.

To determine for which lead sites the band gap changes, a neighbourlist is used. For every halide site, all lead sites within the band gap radius are added to a vector and saved as such. When the rate for a swap is determined, only for the lead sites that are in the list of at least one of these halide sites the band gap is recalculated and so on.

The volume of interest for each halide site is a fraction of the total volume, so using this method the computational costs are reduced significantly by both reducing the recalculation of costly exponents as by reducing the summing.

4.3 Performance

One of the main goals of this thesis is to improve upon the Python code when it comes to speed. In this section the Python code and the C++ code are compared on speed. It is shown that the C++ code performs better for three configurations. Then the scaling for the C++ code and Python code are compared. The code of C++ is then further characterised. The relation between the required time for the simulation and

the different parameters is determined. Finally, some suggestions are made for further improvements of the C++ code.

4.3.1 Direct comparison

First, for three cases the improvement in time is shown. For three smaller boxes of 4x4x4, 6x6x6, and 8x8x8 unit cells and with the same parameters the Python and C++ code is run on the same computer with a single thread. The total simulation time of these simulations are given in Table 2.

Table 2: A direct comparison of the simulation time between Python and the C++ code. The time for all simulations are given as well as the factor that the C++ code is faster. Simulation parameters: $n_{\text{vac}} = 4$, $n_{\text{I}} : n_{\text{Br}} = 50 : 50$, $r_{\text{eg}} = 1.51$, $N_{\text{step}} = 10^5$, $\text{SnapStep} = 10^3$.

Size: $n_x n_y n_z$	Simulation time Python (s)	Simulation time C++ (s)	Factor
4x4x4	35.49	0.700	50.7
6x6x6	110.70	0.761	145
8x8x8	344.58	0.839	410

The factor that the C++ code is faster increases with the size of the simulation box, this will be further explained in Section 4.3.2. For the system of 8x8x8 unit cells the C++ is 410 times faster than the Python code without parallelisation. This is a good baseline for how much faster the C++ code is, as the scaling of both codes is compared later on. With this baseline, when the C++ code scales better than the Python code, it can be concluded that the C++ code is better for every reasonable configuration.

4.3.2 Order of algorithms

The time the simulation runs depends on the parameters of the simulation, like the size of the system or the number of modelled steps.

Number of sites

One of the main issues the Python code faced was the rapid growth of simulation time as the simulation box increases in numbers of unit cells. This relation can clearly be seen in Figure 6, where the set-up and simulation time are plotted against number of unit cells in the simulation box. The fits are both polynomials of order two. The total time has a quadratic relation with number of unit cells, while the set-up's is linear. This means the most time consuming simulation steps' time have a quadratic relation with the number of unit cells. For larger systems this quickly becomes problematic.

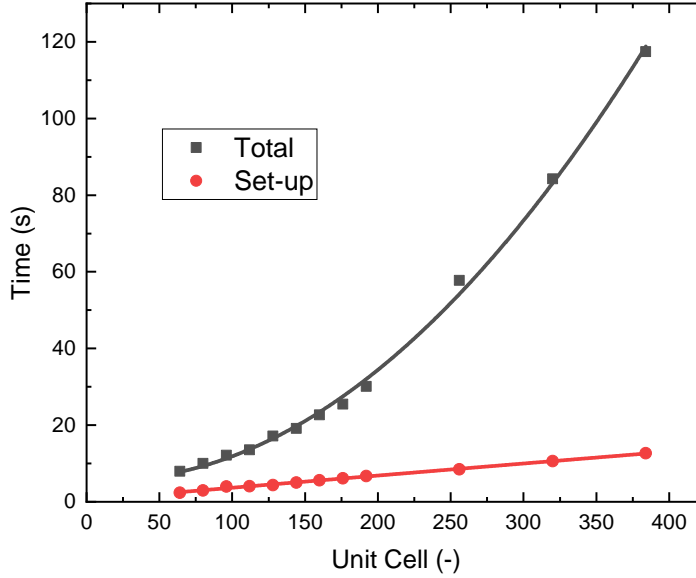


Figure 6: The set-up and total simulation time against the number of unit cells in the simulation box for the Python code. The total time is fitted with an order two polynomial fit, the set-up with a linear fit. $n_{\text{vac}} = 4$, $n_{\text{I}} : n_{\text{Br}} = 70 : 30$, $N_{\text{step}} = 10 \cdot 10^3$

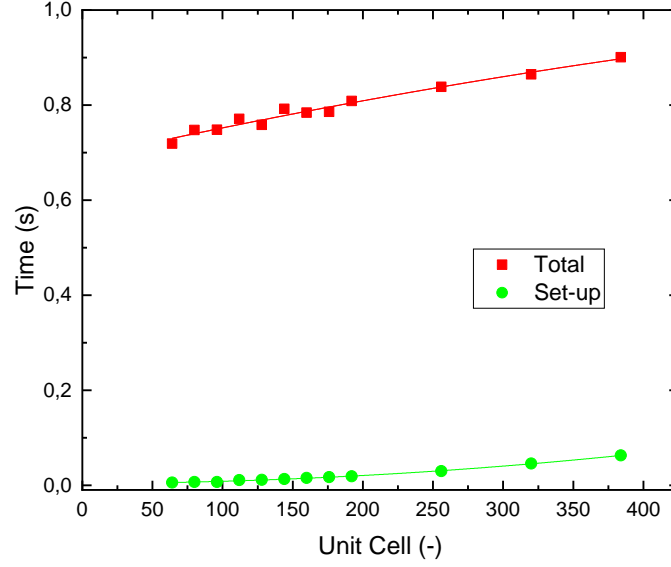
The C++ code saves the positions of the vacancies and the different neighbour lists in a vector and not in a table. This method was designed to be less computationally costly, especially for larger systems. The results suggest that this method works as designed, as can be seen in Figure 7. The order two polynomial fit for the total time shows no (positive) quadratic relation for the total time, but does so for the set-up time. For large systems the simulation steps do not take up significantly more time if the other parameters, like the number of vacancies, stay the same. The quadratic relation that can be seen for the set-up time in Figure 7b can be explained by the way the neighbour lists are constructed. The distance between each pair of sites, lead and/or halide, is determined and compared with r_{eg} and r_{cut} . This is the most time consuming part of the set-up. The number of pairs scales quadratically with the number of unit cells, hence the observed quadratic relation for the set-up time.

The goal to linearise the simulation time unit cell relations has been partially succeeded. The total time scales for small numbers linearly, but the set-up time does not. For larger systems the ratio between set-up and total will be larger, which means the total time will also, significantly, scale quadratically. The linearisation of the set-up time is, however, easily possible, as will be elaborated in Section 4.3.4.

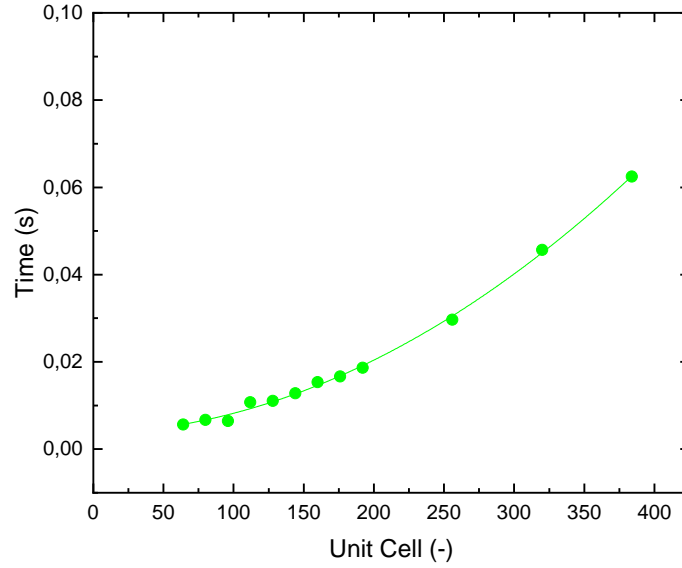
Number of vacancies

The number of vacancies in the simulation box are increased with the goal to lessen the number of steps required to observe segregation. This however comes with a trade-off for the time, as can be seen in Figure 8. The simulation time is linear with the number of vacancies for the three sizes. This means that the increase of the number of vacancies does not necessarily improve the simulation time. The relation between the number of steps required for segregation to occur and the number of vacancies is discussed in Section 5.5.

For the 4x4x4 case the time is a little less than linear for high number of vacancies. This can be explained by the exceptionally high concentration of vacancies, 60 vacancies is roughly 30%, which means less transitions are possible and therefore less energy and rate calculations are performed.



(a) The total and set-up time in seconds against number of unit cells for constant number of vacancies. Fit with an order two polynomial fit.



(b) The set-up time against the number of unit cells for constant number of vacancies. Fit with an order two polynomial fit.

Figure 7: For the C++ code the set-up and total simulation time against the number of unit cells in the simulation box. Both are fitted with an order two polynomial. In Figure 7a it can be seen that the total time is nearly linear, the square factor parameter is negative, which cannot hold for large number of unit cells. In Figure 7b it can be seen that the set-up time goes quadratic with the number of unit cells. Simulation parameters: $n_{\text{vac}} = 4$, $n_{\text{I}} : n_{\text{Br}} = 70 : 30$, $N_{\text{step}} = 10 \cdot 10^3$.

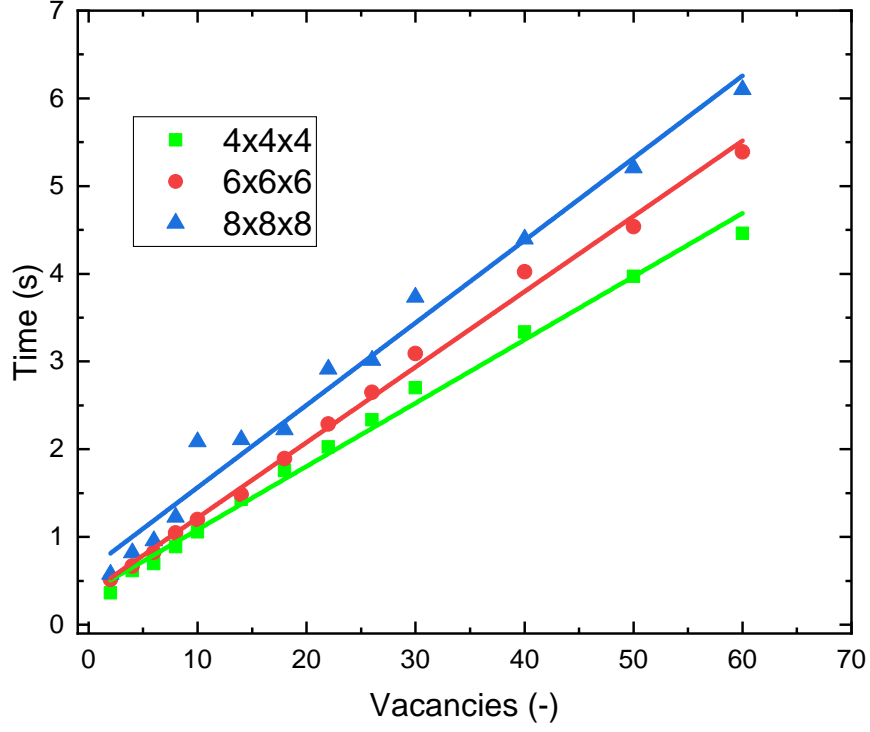


Figure 8: The simulation time against the number of vacancies for three differently sized simulation boxes. For the three simulation box sizes the simulation time is approximately linear with respect to the number of vacancies. The 4x4x4 case has a slower increase for high numbers of vacancies which can be explained by the exceptionally high concentration of vacancies, which means less transitions are possible and less calculations are performed. Simulation parameters: $n_I : n_{Br} = 70 : 30$, $r_{eg} = 1.8$ unit, $N_{step} = 10 \cdot 10^4$, number of threads is 10.

Size energy ball

The model uses the energy ball, with radius r_{eg} , to calculate the local band gap. The C++ code also uses the information of the corresponding neighbour list, `egList`, to prevent the recalculation of the band gap for all lead sites. For the lead sites within this distance from a (possible) swap the band gap needs to be recalculated, but the other sites' band gap are not affected. This means a large decrease of unnecessary calculations, but also that the amount of calculations that are actually performed does depend on r_{eg} . This relation can be seen in Figure 9. The data is fitted with an order three polynomial, which fits well. This makes sense as the number of lead sites for which recalculation is required scales with the volume of the energy ball, which is cubic with r_{eg} .

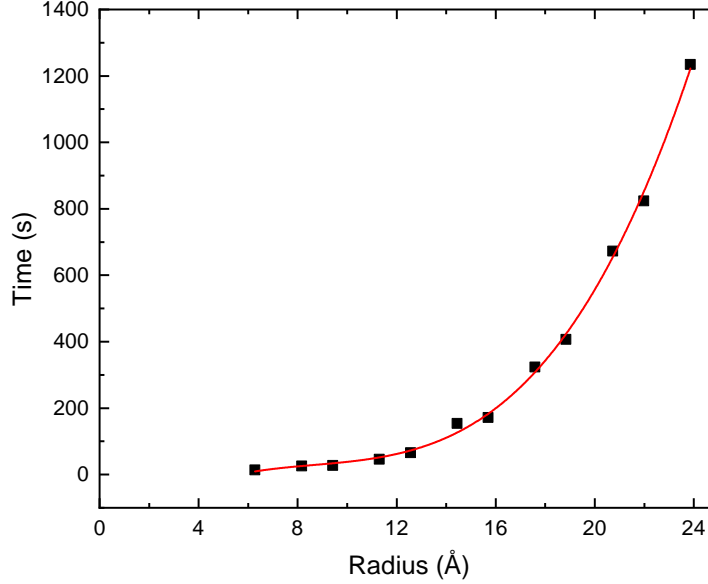


Figure 9: The simulation time against the radius of the energy ball r_{eg} with an order three polynomial fit. Simulation parameters: $8 \times 8 \times 8$, $n_{\text{I}} : n_{\text{Br}} = 50 : 50$, $n_{\text{vac}} = 15$, $N_{\text{step}} = 4 \cdot 10^5$.

What value of r_{eg} is best for simulating reality is not known, but is useful to know what the effect is on the simulation time. The effect of r_{eg} on the segregation is discussed in Section 5.

4.3.3 Test case

The goal of a $20 \times 20 \times 20$ unit cells simulation was set as a proof of concept for scaling. A $20 \times 20 \times 20$ unit cells simulation with parameters $n_{\text{vac}} = 10$, $n_{\text{I}} : n_{\text{Br}} = 70 : 30$, $r_{\text{eg}} = 2.3$ unit, $N_{\text{step}} = 3 \cdot 10^5$, **SnapStep** $=10^3$, and two threads were used. These parameters are around the standard values, so form a reasonable view for the simulation time. The total simulation of this simulation is 231 s, the set-up is 20 s. This was performed on a regular laptop, which shows that the $20 \times 20 \times 20$ unit cells simulations are possible at a large scale. It also shows that for $20 \times 20 \times 20$ unit cells the set-up time is not a bottle neck for the simulation time.

4.3.4 Further speed improvements

Precalculation exponents

The first proposed improvement is the precalculation of the band gap exponents. The calculation of an exponential is slow on a computer as a high order Taylor approximation is used. Therefore it can be useful to minimise the number of times an exponential is calculated. This can be done specifically for the band gap exponent $\exp(-E_{\text{gap},i}/(k_B T))$. As the temperature T remains constant, only the band gap needs to be looked at. The band gap of a lead site is calculated by taking the ratio of bromine and the total number of halide ions within the energy ball of the lead site. Since the number of halides per energy ball is limited, the number of different concentrations, and hence band gaps, is also limited. An upper limit for the number of different band gaps can be found as follows. The number of bromine in the energy ball is capped by the number of halide sites within the energy ball n_{eg} . Note that every energy ball contains the same number of halide sites due to the periodic symmetry of the crystal. Similarly, the number of iodine within the energy ball is also capped by n_{eg} . Hence the number of different combinations of bromine and iodine, and therefore the number of different band gaps, is limited by $(n_{\text{eg}} + 1)^2$. The exact number of possibilities will be lower since the number of bromine and iodine added together also needs to be lower than n_{eg} and the amount of vacancies, sites without bromine and iodine, is also limited. Hence the number of precalculated exponents is less than $(n_{\text{eg}} + 1)^2$.

This upper limit can be compared with an estimate of the number of band gap exponents that are calculated during the simulation. As the simulation runs, the band gap and its corresponding exponent is recalculated for every lead site whose energy ball overlaps with either the vacancy's site or the site it would be swapping with. This would mean roughly $8 \cdot n_{\text{eg}}$ band gap exponents are calculated for every vacancy during each step. So roughly the number of calculated band gap exponents would be around $8 \cdot N_{\text{step}} \cdot n_{\text{eg}} \cdot n_{\text{vac}}$. Since in a typical experiment $8 \cdot N_{\text{step}} \cdot n_{\text{vac}} \gg n_{\text{eg}}$, the number of calculated band gap exponents could be reduced with orders of magnitude. Note that this is only because the time it takes for a value to be found in the memory is orders of magnitude faster than calculating an exponent. By profiling it was seen that roughly 20% of the time is used for calculating the exponentials for band gaps. So the precalculation is estimated to result in roughly a 10-15% reduction in simulation time, which is a significant time save for what still remains possible. It would also be relatively simple and little time consuming to implement in the code.

The exponent that is used to calculate the rate of each possible transition, as seen in Equation (7), cannot be as easily precalculated. This is due to the vast number of different energy difference that can occur in the simulation. It is therefore not worthwhile to pursue the precalculation of the rate.

4.3.5 File export reduction

In the current code the profiling shows that a significant portion of the time is spent on saving text files with the configuration information, around 15% of the time. The number of output files could possibly be reduced by first determining what configurations are useful to save, as now just every configuration with an arbitrary number of steps as interval is saved. This could be done by finding some kind of measure that the programme can determine whether a file needs to be saved, by recognising the start of the segregation for example. It would also help with the processing of the data afterwards, which is not taken into account in the simulation time. Less, but more interesting files to analyse would mean a more time efficient analysis for researchers. The amount of time saved would in the simulation time be at most 10%, but the reduced analysis time would be more relevant, however, hard to estimate.

4.3.6 Set-up linearisation

As can be seen in Section 4.3.2 the set-up time of the code is currently quadratic with the number of unit cells. This is due to how the neighbour lists are constructed. The neighbour lists are determined by calculating the distance between every pair of halide and/or lead site and checking if it is less than r_{cut} or r_{eg} . The number of pairs scales quadratically with the number of unit cells, which causes the quadratic relation between set-up time and number of unit cells. The Python code solves this problem by dividing the system into cells, such that if the cell is at a greater distance than r_{eg} , all sites within the cell do not need to be considered. Since the distance calculations are the most time consuming function within the set-up, by using the cells the code will perform linearly with the number of sites. To implement this method would be relatively simple, as the blue print of this method is already present in the Python code and transfers, most likely, quite well to the C++ code.

If the code is to be used for even greater simulation boxes, a different method could be used. In earlier stages of writing the code, another method of determining the neighbour list that is fully linear with the number of unit cells was attempted. First, another box of $[2r_{\text{eg}}] \times [2r_{\text{eg}}] \times [2r_{\text{eg}}]$ is constructed, where $[a]$ is the ceiling function, centred around the origin. Next, all sites within r_{eg} are determined, such that for any site in the actual simulation box (x, y, z) the sites that are within r_{eg} are known by coordinates by subtracting the coordinates from the original box from (x, y, z) . Then functions that can determine the index of the site by its coordinate (and type) determines the indices of all sites in the neighbour list(s).

This second method linearises the relation between the set-up time and the number of unit cells, which is why it was attempted. It was however incorrectly implemented, as faulty neighbour lists were produced via this method. Since the set-up of the code is currently for the target 20x20x20 unit cells box around 20 seconds on a regular laptop, it was deemed not worthwhile to pursue further in this thesis. If the code is to be used for systems of around 40x40x40 unit cells, the set-up time would take around 60 times as much as for the 20x20x20 system. Then it might be worth pursuing either the previously attempted or another linearisation method of the set-up. The method is far from impossible to implement, but some time needs to be spend on it. Quite possibly, however, the actual simulation steps will still be the bottle neck, although this does depend on the other parameters of the desired simulation.

4.3.7 Parallelisation

The simulation time for large numbers of vacancies has been greatly reduced due to the parallelisation of the energy and rate calculations. It can however be further improved by using different algorithms to parallelise. Over 20% of the time is now spend on copying the `nextEventList` from the different threads to a single list, which is not parallelisable. Therefore, the maximal gain by further parallelisation following Amdahl's law is limited. There are possibilities to improve the parallelisation such that further time saving can be achieved for the energy and rate calculations.

The set-up is currently not parallelised, it is all run on a single thread. The distance calculations could, however, be parallelised quite easily as it has been done before. The sites could be divided over the threads and as such calculate the distance parallel. It could reduce the set-up time with the same factor as for the simulation steps, slightly lower than the number of threads.

5 Modelling Results

5.1 Method and Data Analysis

The code can run for any number of steps and due to the (pseudo)random nature, outcomes might deviate for different numbers of steps. The hypothesis is that at a moment in time due to a series of random events the right configuration for nucleation occurs, after which the segregation occurs relatively quick. Since the computing power is limited, decisions on when to export the configuration and when to stop the simulation need to be made. The parameters of the model, like the number of charge carriers and number of vacancies, are also taken larger than in realistic situations to let the nucleation occur quicker in the simulation to save computing power.

5.1.1 Segregation measure

The model outputs the state of the system for every **SnapStep** time steps. To determine whether there is significant segregation, a measure is needed. What is expected to be observed are iodine rich regions. The concentration of iodine is expected to be high, up to a certain distance and then quickly drop of to the concentration as expected in a well mixed situation. To test this hypothesis, dark conditions are used as a baseline. The average concentration between the centre and a particular distance, depending on the simulation parameters, can be used to determine what influence the light has on segregation.

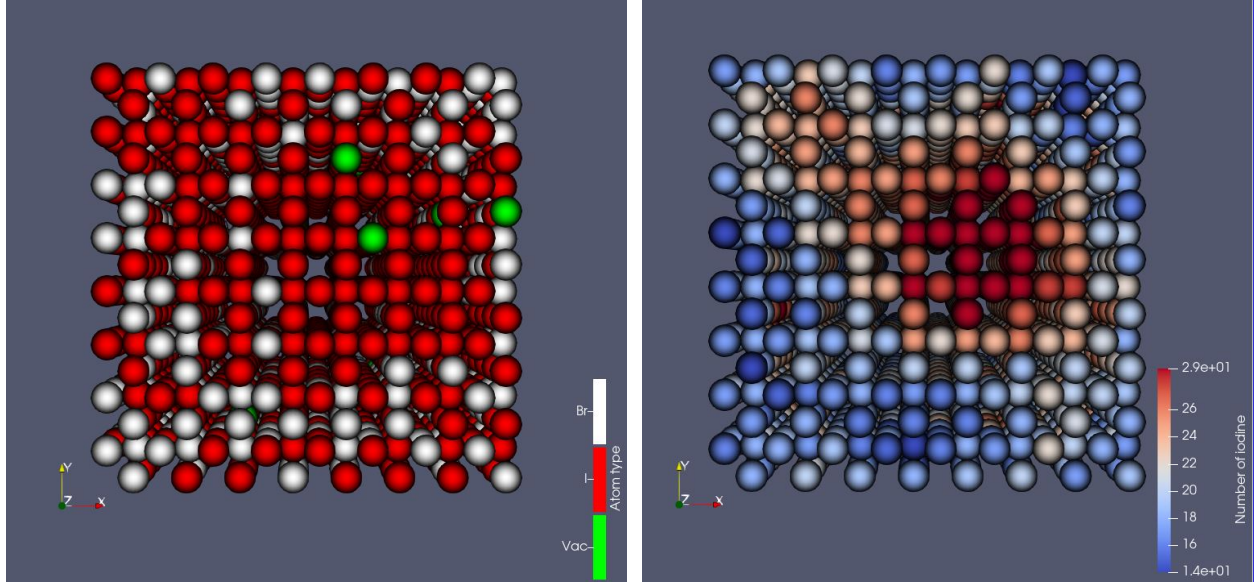
5.1.2 Visualisation

To visualise the system the open source software ParaView is used. The coordinate of every halide site along with its type and the local number of bromine and iodine are loaded into the programme which can represent the state of each site in 3D. An example of a visualisation in ParaView is depicted in Figure 10, which shows two different visualisations of the halide sites for the same state. In Figure 10a the halide type at each site is shown. In Figure 10b the number of iodine ions within r_{eg} from each site is depicted. With the visualisation method from Figure 10a it is non-trivial to detect iodine rich zones, while the visualisation method of Figure 10b clearly shows different concentrations of iodine and an iodine rich zone is visually present. The number of ions within r_{eg} from each site is a value that has no meaning outside visualisation.

To give a quantitative graph of the system, plots of the discrete radial distribution are used. The discrete radial distribution at distance d_n is defined as the the fraction iodine for the halide sites at distances between d_n and d_{n+1} , $d_n < d_{n+1}$. d_n and d_{n+1} are defined such that the number of halide sites is non-zero. In this research, d_n and d_{n+1} are at equal distance for all n . The fraction iodine c_I is defined as

$$c_I = \frac{N_I}{N_I + N_{\text{Br}}}, \quad (15)$$

where N_{Br} and N_I are the number of bromine and iodine ions in the range d_n to d_{n+1} from the centre. The radial distribution is found using a Python script and the graphs are made with the Matplotlib package. In an iodine rich zone, it is expected that the fraction iodine close to the centre is close to 1 and drops to the mixed state fraction far from the centre. An example of such a graph is shown in Figure 11.



(a) The halide sites and its type are shown. White, red and green spheres represent bromine, iodine and vacancy respectively at that site.

(b) The halide sites and the number of iodine ions within a distance of r_{eg} from the site are shown. Dark red indicates high number, while dark blue low numbers of iodine.

Figure 10: Two examples of visualisations of an 8 by 8 by 8 system in ParaView. In 10a the ion type is shown, while in 10b the number of iodine ions within r_{eg} from each site is shown. An iodine rich region can be seen in both visualisations. The dark red region in 10b is an iodine rich region, it can also be seen at the same location in 10a, mainly by the lack of white spheres.

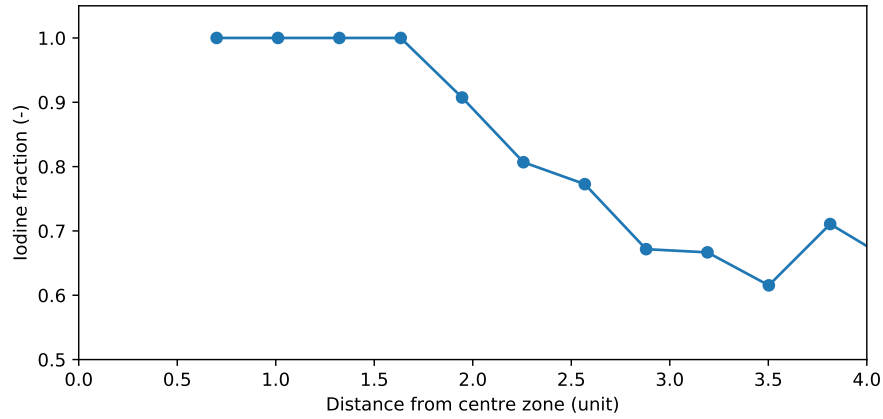


Figure 11: An example of a radial distribution plot. It shows the fraction of iodine of the halides per distance from the centre of the most iodine rich zone. In this simulation $r_{eg} = 2.3$. The iodine rich zone extends to somewhere between 2.5 and 3, as the fraction at that distance drops under the overall fraction of iodine. The fraction of iodine of the total simulation box can usually be seen by the average fraction at a distance of multiple r_{eg} , in this case the overall iodine fraction is 0.7. Simulation parameters: $n_{vac} = 4$, $n_I : n_{Br} = 70 : 30$, $r_{eg} = 2.3$, $N_{step} = 3.0 \cdot 10^5$.

An iodine rich zone is not precisely defined, however, in this research for the edge of the iodine zone is, as a rule of thumb, the minimal distance for which the iodine fraction is equal to that of the whole crystal. This leaves undefined how strongly segregated the system is, e.g. if the iodine fraction is 1 or 0.5 within the

iodine rich zone for an overall fraction of 0.3. To deal with this variation, if the iodine fraction is higher for one zone than the other, then it is said to be more strongly segregated. An iodine rich zone is therefore also not automatically a sign of segregation, as variation is present throughout the crystal. The most iodine rich zones are selected when processing the data, so a dark simulation is a good reference point, as no segregation is present at these temperatures.

5.2 Band gap radius

The radius of the energy ball r_{eg} , or band gap radius, determines what region is considered when calculating the local band gap. The band gap radius is a parameter that cannot be found experimentally or be deduced from theory. It is therefore interesting to see what influence changing this parameter has on segregation. To determine the influence of r_{eg} some simulations for different values of r_{eg} are run for sufficiently high N_{step} such that the radial distribution does not change drastically anymore. It is found by trial and error that for the 8x8x8 simulation with $n_{car} = 20$ and $n_I : n_{Br} = 70 : 30$, $n_{vac} = 15$ $N_{step} = 4 \cdot 10^5$ is sufficiently high for $1.3 \leq r_{eg} \leq 3.5$. The average radial distribution of the most iodine rich zone between step $3.49 \cdot 10^5$ and $3.99 \cdot 10^5$ for different r_{eg} is shown in Figure 12.

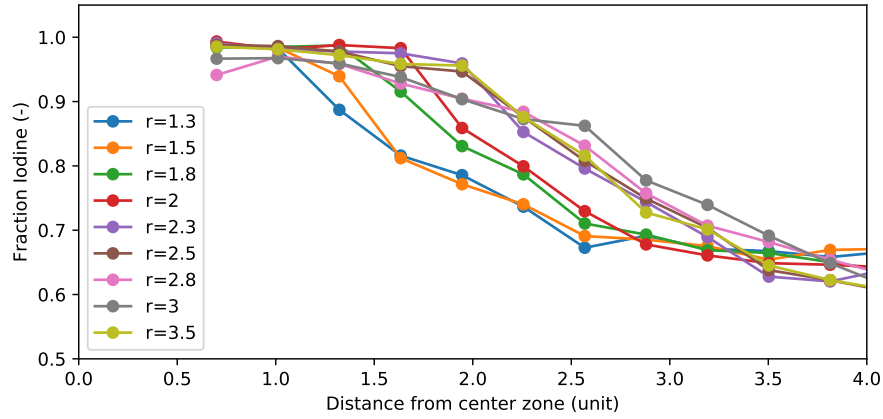


Figure 12: The averaged radial distribution for different values of r_{eg} for high number of charge carriers. The average of the radial distributions are taken from $3.49 \cdot 10^5$ to $3.99 \cdot 10^5$ with intervals of $0.01 \cdot 10^5$. There is a clear increase of radius of the iodine rich zone for larger values of r_{eg} , as for small values (orange and blue) the percentage drops quickly at ca. 1.3 units from the centre, while for larger values (grey and olive) the percentage of iodine remains high for a distance over 2 units. The ratio $n_I : n_{Br} = 70 : 30$ makes sure that the drop is not due to a lack of available iodine to have high percentages at more units. Simulation parameters: 8x8x8, $n_{vac} = 15$, $n_I : n_{Br} = 70 : 30$, $n_{car} = 20$, $N_{step} = 4 \cdot 10^5$.

The size of the iodine rich zone can be seen in Figure 12 to grow in size with increasing r_{eg} , but the size of the zone is never larger than the corresponding r_{eg} . This is a remarkable result as iodine rich zones are hypothesised to grow after a so called nucleus has randomly formed. This cannot be seen in these results. To verify this observation and determine whether it is not caused by the averaging, the progression for $r_{eg} = 1.5, 2, 2.5, 3$ units is shown in Figure 13.

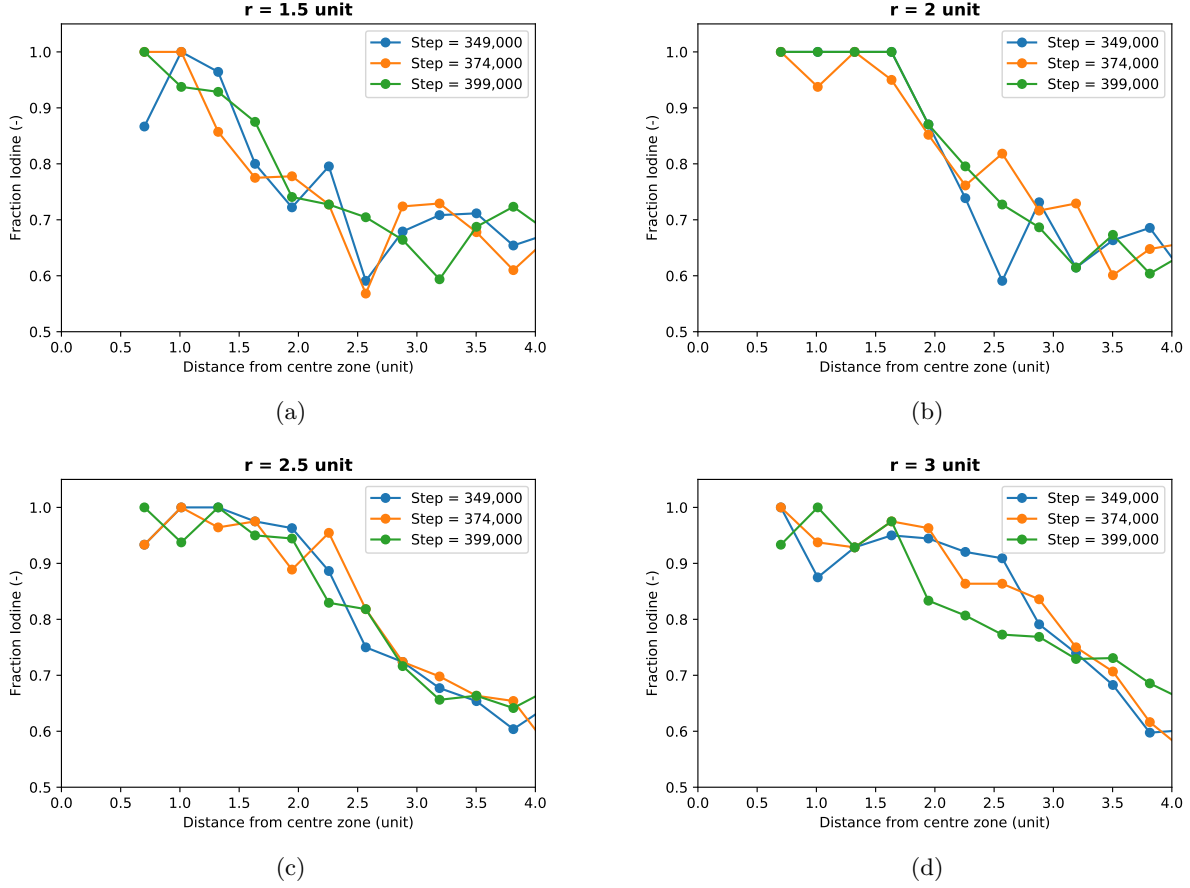


Figure 13: The radial distribution at three different time steps when segregation is present for $r_{\text{eg}} = 1.5, 2, 2.5, 3$. The fraction of iodine drops to the crystal iodine fraction of 0.7 quickly when the distance from the centre is larger than r_{eg} , which underwrites the statement that no iodine rich zones have a significantly larger radius than r_{eg} . Simulation parameters: $8 \times 8 \times 8$, $n_{\text{vac}} = 15$, $n_{\text{I}} : n_{\text{Br}} = 70 : 30$, $n_{\text{car}} = 20$, $N_{\text{step}} = 4 \cdot 10^5$.

No increase of radius of the iodine rich zone over time can be seen in the four graphs in Figure 13. This could be a result of the limited box size in the simulation. Possibly the iodine rich zone would grow if the box was larger, as more iodine would be present in the box. Therefore, larger simulations would be useful to run in the future. It is also interesting to note that the iodine rich zones are not fully iodine or fully stable over time. The percentage of bromine around the centre of the iodine rich zone varies over 10% for $r_{\text{eg}} = 1.5, 3$ units. This is even with a relatively high number of carriers, for which strong, and therefore more stable, segregation is expected.

5.3 Carrier density

The carrier density of the system influences what energy difference is caused by the halide hopping into a vacancy. With large energy differences, which correspond to high carrier densities and therefore strong illumination, faster and possibly stronger segregation is expected. The radial distribution is averaged for $1.49 \cdot 10^5$ to $1.99 \cdot 10^5$ steps, with an interval of $0.01 \cdot 10^5$ steps. The results are shown in Figure 14.

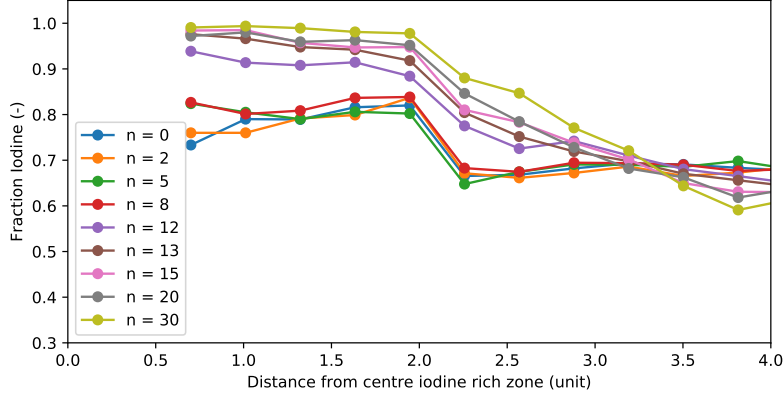


Figure 14: The averaged radial distribution between $N_{\text{step}} 1.49 \cdot 10^5$ to $1.99 \cdot 10^5$ steps and interval of $0.01 \cdot 10^5$ steps for different numbers of charge carriers. The segregation, i.e. high concentrations close to the centre, are seen for high numbers of charge carriers, while for low values these are not seen. For these low charge carrier cases they resemble the control experiment with zero charge carriers, the dark situation. Simulation parameters: $8 \times 8 \times 8$, $n_{\text{vac}} = 15$, $n_{\text{I}} : n_{\text{Br}} = 70 : 30$, $r_{\text{eg}} = 2.3$ unit.

For low charge carrier numbers, and hence densities, the radial distributions resemble that of the dark case at $n_{\text{car}} = 0$. For the higher cases the averaged radial distribution show an increase in iodine concentration for distances up to ca. r_{eg} , which suggests that higher charge carrier densities cause stronger and more stable segregation. The segregation for $n_{\text{car}} \geq 12$ were checked to be fully segregated, i.e. no significant increase or decrease occurred for $N_{\text{step}} > 2 \cdot 10^5$. The other hypothesis with respect to the charge carrier densities is that higher numbers cause faster segregation. In Figure 14 the segregation is complete for every case, at less steps this is not the case and the segregation is in progress for lower numbers of carrier density. This is shown in Figure 15, where the averaged radial distribution over $0.3 \cdot 10^5$ to $0.34 \cdot 10^5$ steps and interval of $0.01 \cdot 10^5$ steps is shown. Clearly, the fully segregated cases of Figure 14 are not all fully segregated and a progression with higher numbers of charge carriers can be seen in the radial distribution for these steps.

To understand why the averaging done in Figure 14 and Figure 15 is useful, the radial distribution for single steps at different numbers of steps are shown in Figure 16. The segregation is basically complete, but deviations occur for every distance at specific time steps. The averaging shows how stable the segregation is. It shows that some bromine remains in the fully segregated regions due to the randomness of the process. In Figure 14 higher numbers of carriers correlate with on average less bromine mixed in which corresponds to a higher level of segregation.

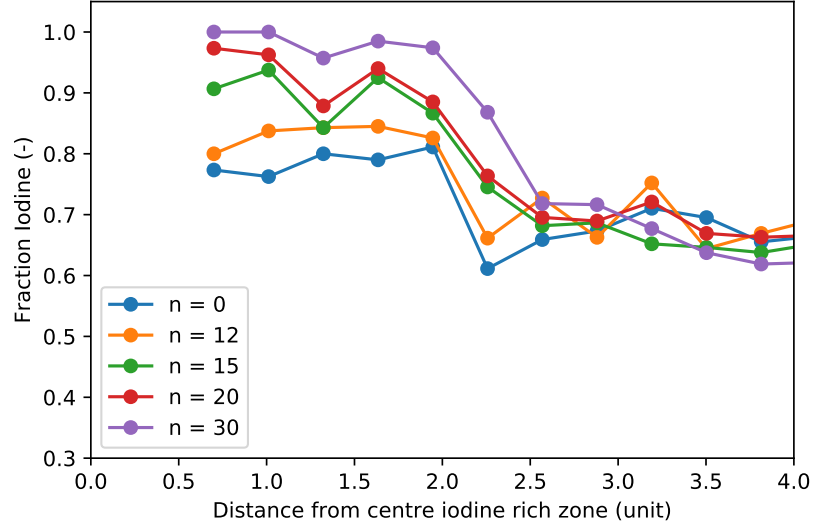


Figure 15: The averaged radial distribution between $0.3 \cdot 10^5$ to $0.34 \cdot 10^5$ and interval of $0.01 \cdot 10^5$ for different numbers of charge carriers. For all but the dark case the system has shown segregation at $N_{\text{step}} = 4 \cdot 10^5$, as shown in Figure 14. Here it shows that higher charge carrier numbers lead to quicker segregation, as for the higher charge carrier numbers, the segregation already has occurred and not for the lower charge carrier numbers. Simulation parameters: $8 \times 8 \times 8$, $n_{\text{vac}} = 15$, $n_{\text{I}} : n_{\text{Br}} = 30 : 70$, $r_{\text{eg}} = 2.3$.

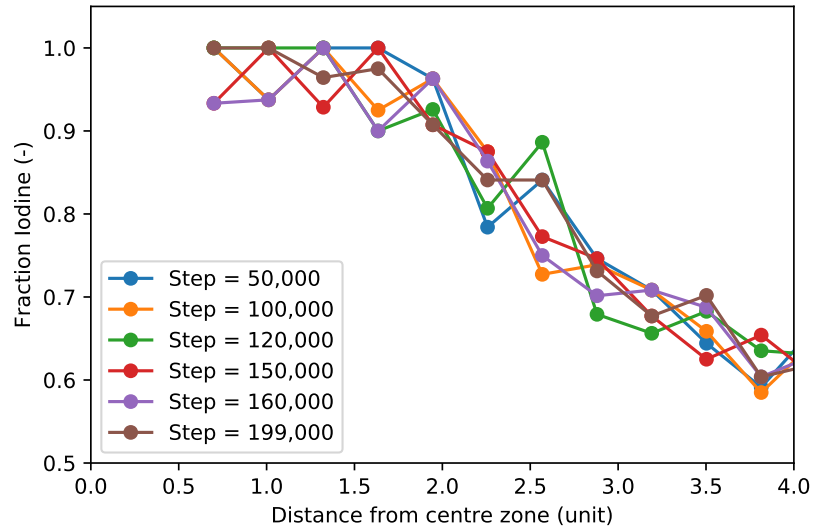


Figure 16: The progression of segregation for $n_{\text{car}} = 20$. At some point the segregation does not further progress, but random deviation do occur. This can be seen by the deviations at every measured distance. The averaging helps in the visualisation when comparing different charge carrier numbers. Simulation parameters: $8 \times 8 \times 8$, $n_{\text{vac}} = 15$, $n_{\text{I}} : n_{\text{Br}} = 30 : 70$, $r_{\text{eg}} = 2.3$ unit.

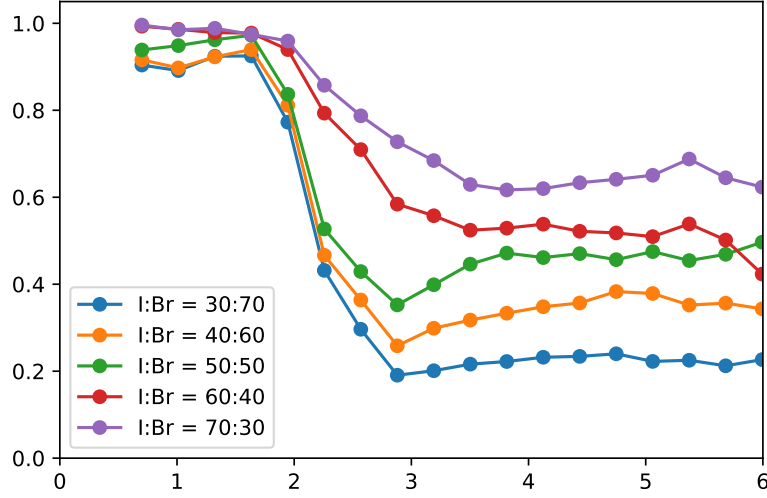


Figure 17: The radial distribution for fully segregated systems with varying ratios of bromine to iodine. The segregation is stronger for higher concentrations of iodine. The Simulation parameters: $8 \times 8 \times 8$, $n_{\text{car}} = 20$, $n_{\text{vac}} = 15$, $r_{\text{eg}} = 2.3$ unit $N_{\text{step}} = 3.5 \cdot 10^5$ to $4.0 \cdot 10^5$

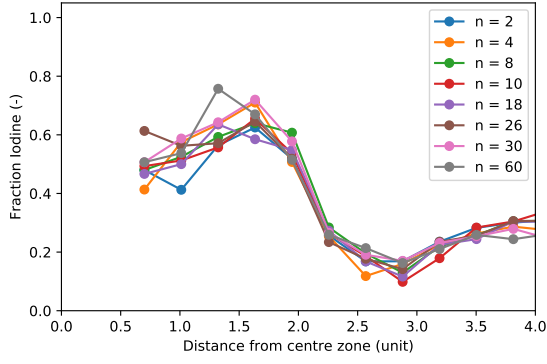
5.4 Halide ratio

Segregation is hypothesised to be activated by a randomly occurring, relatively, iodine rich zone. When the concentration of iodine is higher in the model, these iodine rich zones are bound to occur more often. There is also less bromine in the system to randomly be diffused into the iodine rich zone. In Figure 17 the average radial distribution for $N_{\text{step}} = 3.5 \cdot 10^5$ to $4.0 \cdot 10^5$ with an interval of $0.1 \cdot 10^5$ is shown for different concentrations, when they are fully segregated.

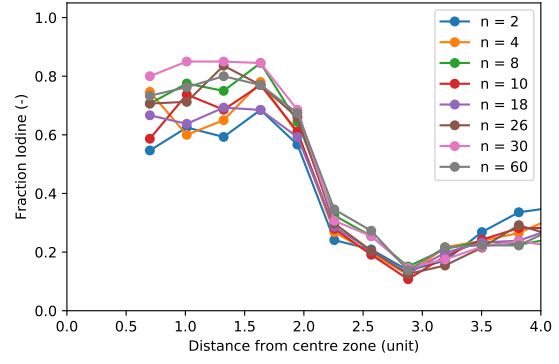
For higher concentrations of iodine in the simulation, less bromine diffuses into the iodine rich zone as expected. This implies that the iodine rich zones for lower concentrations of iodine are less stable and it would be interesting to see whether tuning of both the charge carrier density and iodine concentration could lead to mixed and demixed stable situations in the model and whether there is some turning point that can be found.

5.5 Vacancy density

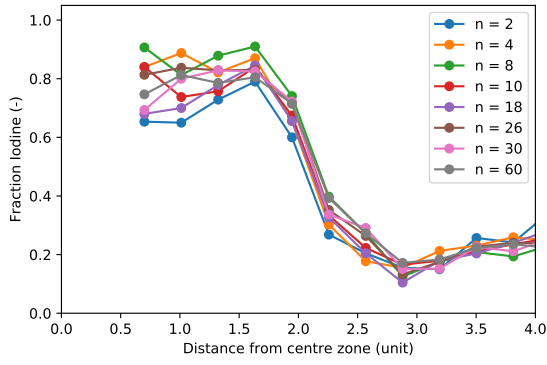
One of the parameters that is set higher in the model than realistic values is the vacancy density. The concept behind this parameter picking is that more vacancies leads to faster segregation in experiments. While this might be true for the time that is simulated, it is looked into what number of steps are required for segregation to occur. The hypothesis that less steps are required for higher numbers of vacancies is tested by plotting the radial density for different numbers of vacancies at certain time steps, as is done in Figure 18. It can be seen that the required number of steps for segregation does not appear to correlate with the number of vacancies in the simulations. It appears that the iodine rich zones are less stable for really high numbers of vacancies, as can be seen by comparing the sixty vacancies case to the others in Figure 18e and Figure 18f. No physical conclusions can be drawn from this, as these situations will not occur in realistic situations, but are nevertheless remarkable as they go against the hypothesis.



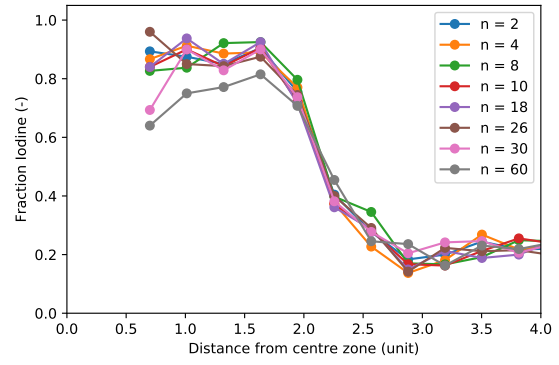
(a) $N_{\text{step}} = 5 \cdot 10^3$ to $10 \cdot 10^3$



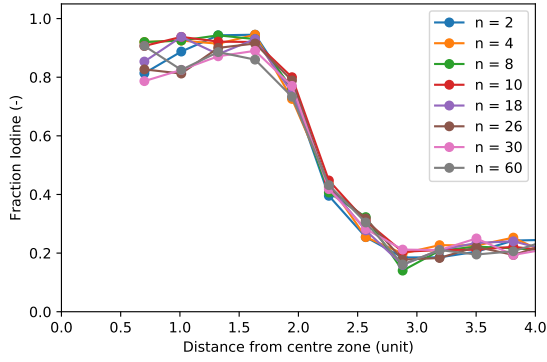
(b) $N_{\text{step}} = 25 \cdot 10^3$ to $30 \cdot 10^3$



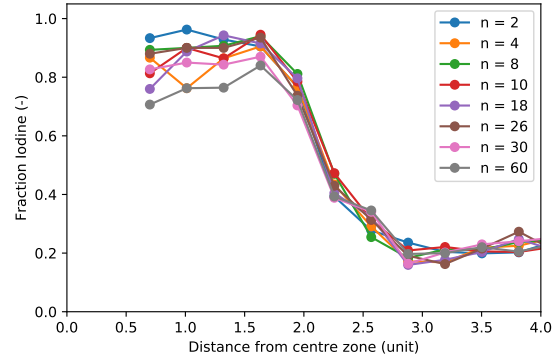
(c) $N_{\text{step}} = 45 \cdot 10^3$ to $50 \cdot 10^3$



(d) $N_{\text{step}} = 95 \cdot 10^3$ to $100 \cdot 10^3$



(e) $N_{\text{step}} = 145 \cdot 10^3$ to $150 \cdot 10^3$

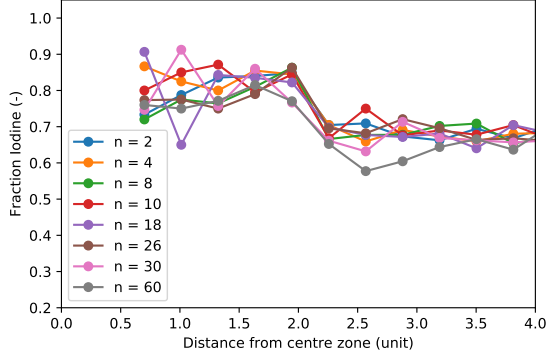


(f) $N_{\text{step}} = 345 \cdot 10^3$ to $350 \cdot 10^3$

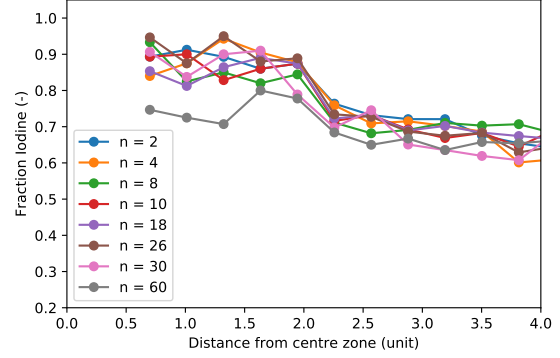
Figure 18: The radial distribution of iodine for different numbers of vacancies, each at different time steps. The radial distribution of five time steps, spaced by 10^3 , is averaged in these plots. The segregation can be seen as more simulation steps are performed. Interestingly, the number of vacancies does not have any clear influence on what rate with respect to the number of steps this occurs. The original concentration of iodine is 0.3.

As seen in Section 4.3.2, the simulation time is roughly linear with the number of vacancies. It is therefore highly useful to note that the number of steps required before segregation occurs does not seem to depend on the number of vacancies within the simulation. It would mean that a low number of vacancies is enough to simulate the crystal, which saves significant simulation time and computation power. The simulations

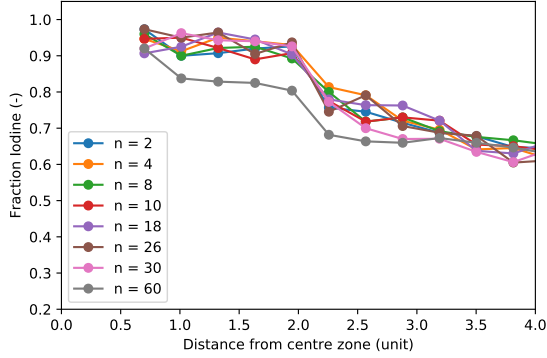
are performed for 8x8x8 unit cells, so there might be a lower limit for the number of vacancies that cannot be seen in this relatively small simulation. It is also likely that for systems like 20x20x20 unit cells multiple iodine rich zones can form in a single simulation. With a single vacancy, this is unlikely.



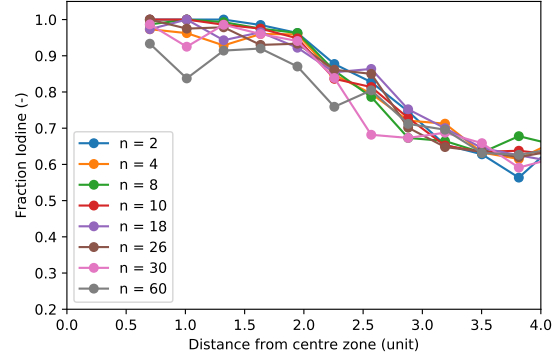
(a) $N_{\text{step}} = 5 \cdot 10^3$ to $9 \cdot 10^3$



(b) $N_{\text{step}} = 15 \cdot 10^3$ to $19 \cdot 10^3$



(c) $N_{\text{step}} = 25 \cdot 10^3$ to $29 \cdot 10^3$



(d) $N_{\text{step}} = 45 \cdot 10^3$ to $49 \cdot 10^3$

Figure 19: The averaged radial distribution of iodine for different numbers of vacancies, each at different time steps. The radial distribution of five time steps, spaced by 10^3 , is averaged in these plots. The segregation can be seen as more simulation steps are performed. Interestingly, the number of vacancies does not have any clear influence on what rate with respect to the number of steps this occurs. The original concentration of iodine is 0.7. The results are similar to that of an original concentration of 0.3.

Note that the simulation time increases with a higher number of vacancies, but the time that is simulated, i.e. the modelled real life equivalent, is proportional to the inverse of the number of vacancies. This kind of simulations can be used to test the time simulated for segregation to occur for more vacancies decreases in future research.

6 Conclusion and outlook

Perovskite solar cells have shown great potential for high efficiency solar cells. Especially mixed halide perovskites are of interest for their application in tandem solar cells due to their tunable band gap. However, perovskite solar cells face stability issues, one of which is light-induced halide segregation in mixed halide perovskites. This segregation is not well understood, although some models exist. One of these models cannot produce useful results due to speed and scaling issues of the code. This project's goal was to improve upon the current code by making it faster and scale for larger systems. By restructuring the code, switching from Python to C++, and parallelising the rate calculation, the project goal has been achieved. In the verification of the model some behaviour not corresponding to experimental observations has been seen.

It is shown that the C++ code is 410 times faster for a single threaded simulation than the Python code, for a simulation with 8x8x8 unit cells. This is used as a base case for the comparison between the C++ code and Python.

The Python code showed issues for scaling with the number of unit cells and this is confirmed, as the simulation time scales quadratically with the number of unit cells. The C++ code shows improvement on scaling as the total simulation time is linear with the number of unit cells in the simulation box. The set-up time of the C++ code does, however, show a quadratic relation with the number of unit cells. This could potentially be relevant for systems larger than 20x20x20 unit cells, as for 20x20x20 unit cells it was shown to still be less than 10% of the total simulation time, which means that it does not influence the simulation starkly. Since the C++ code has been shown to perform in linear time for the number of unit cells, excluding the set-up time, and the Python code performs quadratically, the base factor of 410 is expected to only increase for larger systems than 8x8x8 unit cells. As an example, the simulation for 20x20x20 unit cells would take weeks with the Python code and only minutes with the C++ code.

The C++ code performs linearly with the number of vacancies, while the number of steps required to see segregation does not appear to correlate with the number of vacancies in the model. The parallelisation of the simulation steps makes that the total simulation time increase, caused by more vacancies, can be largely counteracted by using multiple threads. The factor that using multiple threads speeds up the simulation is slightly below the number of threads, which is due to Amdahl's law, since, among other things, the copying of the `nextEventList` of each thread and the single thread tasks like picking the next event cannot be properly parallelised. For small numbers of vacancies, these effects are larger. Furthermore, this only works if all threads perform roughly equal work. All threads perform roughly equal work in the simulation steps if the number of vacancies is divisible by the number of threads. The cores are then used the most efficient. It is therefore recommended to perform simulations with a low number of vacancies or a larger number of vacancies that is a multiple of the number of threads used.

Further speed and scaling improvements are possible. It is estimated that a speed increase of ca. 30% for systems larger or equal to 20x20x20 unit cells can be achieved by implementing the improvements regarding the precalculation of the exponentials, the linearisation of the set-up and improving the parallelisation. These improvements are most likely not highly time consuming. Improvements further than 30% are most likely more time consuming to implement than time saving and therefore not worthwhile.

The behaviour of the model has also been looked at. The expected increase of segregation for increasing photocarriers has been shown. For some photocarrier densities no increase has been shown, which indicates a minimal photocarrier density in the system, but this has not been shown thoroughly and requires further investigation. An unexpected result was the lack of growth of the iodine rich zones, while this is expected by the thermodynamic theory or the experimental observations. The iodine rich zones did not grow significantly larger than r_{eg} , hence more research is required into this model before this model can be used to draw conclusions about the halide segregation in MAPBI.

6.1 Outlook

The goal of this thesis is about improving the code and performing some simple verification of the model. Therefore, the actual application of the code will happen after this project. A more thorough verification of the model, for example by matching the results of the code with the experimental observations, is also recommended before using the code and model to draw conclusions about the behaviour of MAPBI under

illumination. Unexpected behaviour like the lack of growth of the iodine rich zones needs to be understood or gotten rid of before the model can be used for predictions.

The C++ code has been shown to function best for large systems with low vacancy densities, although with higher vacancy densities the parallelisation of the simulation loop can resolve this. For low vacancy densities the set-up becomes the bottle neck, which means that parallelisation of the set-up would be worthwhile. The current set-up is far from optimised as for the set goal of 20x20x20, the set-up is not the bottle neck. The finding that the number of vacancies does not influence the required number of steps was also done late in the project, so high vacancy density optimisation was prioritised. The parallelisation of large parts of the set-up is a relatively simple task, with the distance calculations for each pair of sites being the bottle neck. The sites can easily be divided over the different threads, especially if the 'dividing the box into cells' principle of the Python code is implemented. For the details of this method, see Section 4.1 and Section 4.3.4.

The low vacancy densities and large systems resemble the experimental circumstances. It would be interesting to see if, with carrier densities resembling experiment, the observed segregation also occurs. In Section 5.3 for low carrier density, still higher than realistic, no segregation is seen, but it might be the case that with longer simulation times or different seeds the segregation would occur. By repeating the experiment for different seeds and with large numbers of steps, this can be investigated.

The unified theory proposed in Chen et al. (2021) produces predictions that can be tested with this model [14], once it matches experimental results. The thermodynamic theory that is proposed predicts when halide segregation occurs and when no halide segregation occurs for a multitude of bromine concentrations and illumination levels, and these circumstances can all be tested with this model and code.

Since only roughly every 1000th configuration is saved to save computing power and time, the exact configuration for which the system starts to segregate cannot be determined. An advantage of the use of a seeded PRNG is that the exact same run can be repeated with more configurations being saved around the moment of segregation. Using this method the exact requirements for segregation, according to the model, can be studied by first having an exploratory run with large intervals between each saved configuration and then a finer run with smaller intervals.

2D perovskites are an upcoming research topic due to their lack of light-induced halide segregation [33]. The code can be altered to model 2D perovskites and further the understanding of why 2D perovskites are so stable.

References

- [1] Nicola Armaroli and Vincenzo Balzani. “Solar electricity and solar fuels: status and perspectives in the context of the energy transition”. In: *Chemistry—A European Journal* 22.1 (2016), pp. 32–57.
- [2] Patrick Tonui et al. “Perovskites photovoltaic solar cells: An overview of current status”. In: *Renewable and Sustainable Energy Reviews* 91 (2018), pp. 1025–1044.
- [3] NREL NREL. *Best Research-Cell Efficiency*. 2020.
- [4] Christopher Eames et al. “Ionic transport in hybrid lead iodide perovskite solar cells”. In: *Nature communications* 6.1 (2015), pp. 1–8.
- [5] S Guha et al. “Band-gap profiling for improving the efficiency of amorphous silicon alloy solar cells”. In: *Applied Physics Letters* 54.23 (1989), pp. 2330–2332.
- [6] Masafumi Yamaguchi et al. “Multi-junction III–V solar cells: current status and future potential”. In: *Solar Energy* 79.1 (2005), pp. 78–85.
- [7] Martin A Green and Stuart R Wenham. “Novel parallel multijunction solar cell”. In: *Applied Physics Letters* 65.23 (1994), pp. 2907–2909.
- [8] Byung-wook Park et al. “Chemical engineering of methylammonium lead iodide/bromide perovskites: tuning of opto-electronic properties and photovoltaic performance”. In: *Journal of Materials Chemistry A* 3.43 (2015), pp. 21760–21771.
- [9] Manoj Jaysankar et al. “Minimizing voltage loss in wide-bandgap perovskites for tandem solar cells”. In: *ACS Energy Letters* 4.1 (2018), pp. 259–264.
- [10] Yousra El Ajjouri et al. “Tunable Wide-Bandgap Monohalide Perovskites”. In: *Advanced Optical Materials* 8.17 (2020), p. 2000423.
- [11] Eric T Hoke et al. “Reversible photo-induced trap formation in mixed-halide hybrid perovskites for photovoltaics”. In: *Chemical Science* 6.1 (2015), pp. 613–617.
- [12] Sergiu Draguta et al. “Rationalizing the light-induced phase separation of mixed halide organic–inorganic perovskites”. In: *Nature communications* 8.1 (2017), pp. 1–8.
- [13] Xi Wang et al. “Suppressed phase separation of mixed-halide perovskites confined in endotaxial matrices”. In: *Nature communications* 10.1 (2019), pp. 1–7.
- [14] Zehua Chen et al. “Unified theory for light-induced halide segregation in mixed halide perovskites”. In: *Nature Communications* 12.1 (2021), pp. 1–10.
- [15] Michael C Brennan et al. “Light-induced anion phase segregation in mixed halide perovskites”. In: *ACS Energy Letters* 3.1 (2017), pp. 204–213.
- [16] Connor G Bischak et al. “Origin of reversible photoinduced phase separation in hybrid perovskites”. In: *Nano letters* 17.2 (2017), pp. 1028–1033.
- [17] Alexander J Knight et al. “Electronic traps and phase segregation in lead mixed-halide perovskite”. In: *ACS Energy Letters* 4.1 (2018), pp. 75–84.
- [18] Rebecca A Belisle et al. “Impact of surfaces on photoinduced halide segregation in mixed-halide perovskites”. In: *ACS Energy Letters* 3.11 (2018), pp. 2694–2700.
- [19] Alex J Barker et al. “Defect-assisted photoinduced halide segregation in mixed-halide perovskite thin films”. In: *ACS Energy Letters* 2.6 (2017), pp. 1416–1424.
- [20] Denis Barboni and Roger A De Souza. “The thermodynamics and kinetics of iodine vacancies in the hybrid perovskite methylammonium lead iodide”. In: *Energy & Environmental Science* 11.11 (2018), pp. 3266–3274.
- [21] Cheng Li et al. “Unravelling the role of vacancies in lead halide perovskite through electrical switching of photoluminescence”. In: *Nature communications* 9.1 (2018), pp. 1–8.
- [22] Anthony Ruth et al. “Vacancy-mediated anion photosegregation kinetics in mixed halide hybrid perovskites: coupled kinetic Monte Carlo and optical measurements”. In: *ACS Energy Letters* 3.10 (2018), pp. 2321–2328.
- [23] Luc Devroye. “Sample-based non-uniform random variate generation”. In: *Proceedings of the 18th conference on Winter simulation*. 1986, pp. 260–265.
- [24] Makoto Matsumoto and Takuji Nishimura. “Dynamic creation of pseudorandom number generators”. In: *Monte-Carlo and Quasi-Monte Carlo Methods 1998*. Springer, 2000, pp. 56–69.
- [25] David Jones. “Good Practice in (Pseudo) Random Number Generation for Bioinformatics Applications”. In: (2010).

- [26] Lucie McGovern et al. “Understanding the stability of MAPbBr₃ versus MAPbI₃: Suppression of methylammonium migration and reduction of halide migration”. In: *The journal of physical chemistry letters* 11.17 (2020), pp. 7127–7132.
- [27] Dane W DeQuilettes et al. “Photoluminescence lifetimes exceeding 8 μ s and quantum yields exceeding 30% in hybrid perovskite thin films by ligand passivation”. In: *ACS Energy Letters* 1.2 (2016), pp. 438–444.
- [28] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2020. Chap. 6.
- [29] Ji-Hui Yang et al. “Fast self-diffusion of ions in CH₃NH₃PbI₃: the interstitially mechanism versus vacancy-assisted mechanism”. In: *Journal of Materials Chemistry A* 4.34 (2016), pp. 13105–13112.
- [30] Andrei Buin et al. “Materials processing routes to trap-free halide perovskites”. In: *Nano letters* 14.11 (2014), pp. 6281–6286.
- [31] Chen Wang et al. “Giant phonon tuning effect via pressure-manipulated polar rotation in perovskite MAPbI₃”. In: *The journal of physical chemistry letters* 9.11 (2018), pp. 3029–3034.
- [32] Yi Zhang et al. “Trash into treasure: δ -FAPbI₃ polymorph stabilized MAPbI₃ perovskite with power conversion efficiency beyond 21%”. In: *Advanced Materials* 30.22 (2018), p. 1707143.
- [33] Chaohui Li et al. “Vertically aligned 2D/3D Pb–Sn perovskites with enhanced charge extraction and suppressed phase segregation for efficient printable solar cells”. In: *ACS Energy Letters* 5.5 (2020), pp. 1386–1395.

A Table of parameters

Parameter	Elaboration	Value indication
n_x, n_y, n_z	The number of modelled unit cells in the x-, y- and z-direction respectively.	Between 8 and 20.
a_x, a_y, a_z	The length of the lattice vector in the x-, y- and z-direction in Angstrom.	6.28 Å for MAPBI around 300 K [32]
n_b, n_i, n_v	The number of bromine, iodine and vacancies in the model respectively. Note that these numbers must satisfy $n_b + n_i + n_v = 3 \cdot n_x \cdot n_y \cdot n_z$, as each halide site is one of these three states	$10\% < n_b, n_i < 90\%$ of halide sites, $n_{\text{vac}} < 1\%$ of halide sites.
r_{cut}	The maximal distance in nanometers between halide sites for a swap to be possible. In this model taken at the distance of the nearest halide neighbours.	4.44 Å
r_{eg}	The radius in Angstrom of the sphere around each lead for which the local concentration of bromine and iodine is determined. Also known as band gap radius. No exact value or realistic value is known.	Between 4.44 Å and 30 Å.
E_b	The binding energy in electrovolts of the halides in the crystal. For both iodine and bromine the same values are used, although they might differ in reality.	0.25 eV
T	The temperature of the system in kelvin. Taken as constant during the simulation.	300 K.
n_{car}	The number of excited charge carriers in the system.	Between 10^{-3} and 10^{-1} per unit cell.
k_0	The prefactor for the rate of each event in per seconds. The exact value is not know, but most likely in the order of 10^{12} , similar to phonon frequencies.	10^{12}
$N_{\text{step}}, t_{\text{max}}$	The maximal number of steps or model time for which the simulation runs. In this model only N_{step} is actively used. The number of steps for similar results scales with the number of unit cells.	10^2 to 10^3 per unit cell

B Manual code

The C++ code can be found on a private repository on GitHub¹. On the repository it is described how the code and the necessary libraries can be installed. Once these packages have been properly installed, the general work flow of performing experiments is as follows. First, create a folder in which the output files need to come. This can be done via Windows File Explorer or using the shell, using 'cd <path>'. Note that if two simulations are run in the same folder, the output files get overwritten if they have the same name. It is therefore recommended to use a separate folder for each simulation.

Second, copy the options.xml file from the folder in which you downloaded the repository to the output file folder. Open the file and change the values to their desired values. Note that the number of vacancies is determined by the number of sites minus the number of bromine and iodine. The options.xml file is shown in Figure 20.

```
<?xml version="1.0"?>
- <options>
  <SEED help="SEED for random number generator">12345</SEED>
  <maxStep help="int, indicating max cycles">10000</maxStep>
  <unitcellcopies help="The number of unit cells in the x, y and z direction respectively (must be ints)">20 20 20</unitcellcopies>
  <numberOfBromine>11998</numberOfBromine>
  <numberOfIodine>11998</numberOfIodine>
  <kbt>0.025852029</kbt>
  <cutoffEnergy help="Cut off used to determine the local bandgap">2.3</cutoffEnergy>
  <cutoffNeighbour help="Cut off for hopping neighbours">0.71</cutoffNeighbour>
  <Eb help="Energy barrier">0.25</Eb>
  <nrOfCarriers help="nr of excited states / photocarriers in the system.">25</nrOfCarriers>
  <outputFileName>test</outputFileName>
  <outputInterval help="Print output file every .. step">1000</outputInterval>
</options>
```

Figure 20: An example of an options.xml file.

Third, the programme needs to be build. By opening the shell, or using a terminal in Visual Studio, go to the repository folder using 'cd <path>'. Then type './build.bat' (on Windows, on Linux ./build), press enter, and the programme is being build. Once complete, the built programme is in the bin folder within the repository folder.

Fourth, the programme needs to be run inside the folder with the options file. Navigate to the folder in the shell or terminal using 'cd <path option folder>'. Then run the programme using '<path programme, ends with .exe> -o options.xml -t <number of threads>'. It is recommended to use a divisor of the number of vacancies as the number of threads to have all cores equally running and thus using the time most efficient.

Fifth, all data files are now available in the output folder where code was run. There is also a file with the time for each of the output files.

¹<https://github.com/rubengerritsen/perovskiteKMC/>