Development of Generalized KMC Code to Study Defect Mobility in Ceramic Materials towards Nuclear Fuel Applications

Aaron Oaks

May 11, 2010

Contents

Chapter 1

Introduction

Fundamental understanding of nuclear fuel materials has of major interest in order to predict fuel performance. One major area of work is the study of diffusion of defects, as these can affect the mechanical properties of the fuel elements. Conventional methods and fuel performance codes have relied mostly on chemical rate theories and experimental data to model fuel performance. More recently, as computational power has evolved, modeling work of nuclear fuel materials at the atomic level has been carried out. However, this subject remains a big challenge today, due to the unavailability of reliable interatomic atomic potentials. These potentials can be used to predict multiple mechanical and thermodynamic properties, defect chemistry and defect transport phenomena, but because of the way they are derived (often by fitting parameters to experimental data, or from first principles calculations), they are often unable to accurately predict multiple material properties simultaneously. For example, using a potential derived by fitting parameters to experimentally determined lattice constants in an atomic scale simulation might very accurately reproduce measured lattice constants for different systems, but might fail to accurately reproduce the measured specific heat for the same systems. When attempting to simulation defect diffusion, it therefore critical that the interatomic potential used be validated before it is used to predict results.

The goal of this study is to develop a generalized kinetic Monte Carlo (KMC) code to simulate defect diffusion that can be used both as a validation tool and as a predictive tool. This code uses migration energies that depend on the migrating particle's local atomic environment. These migration energies are calculated in advance, using Molecular Dynamics (MD) simulations with the desired interatomic potential. As a validation tool, the code would be run using migrations energies generated from the desired interatomic potential and the results would be compared to experimental data. As a predictive tool, the code would be run using migration energies generated from a validated interatomic potential in order to predict data that has not yet been obtained experimentally.

Uranium oxide (UO_2) is the primary fuel used in most nuclear power plants in the United States today. However, because it is radioactive, there are many

practical difficulties involved with studying it experimentally. Cerium oxide (CeO_2) has attracted a lot of attention as a surrogate for uranium oxide because it exhibits many of the same material properties, but is not radioactive. Thus, but studying defect transport and interactions in cerium oxide, similar insight can be gained into defect transport and interactions in uranium oxide. To demonstrate this code as a validation tool, the effect of dopant concentration on oxygen diffusivity in lanthanum-doped cerium oxide is calculated using three different interatomic potentials and compared to experimental data. Lanthanum was chosen as the dopant species mainly because lanthanum is a common fission product created by nuclear fission, so understanding the effects of lanthanum in cerium oxide helps in the understanding of the effects of lanthanum in uranium oxide fuel, and also because, as will be explained later, lanthanum introduces a controllable level of oxygen vacancies, which can help clarify the hypostoichiometric effects in cerium oxide and uranium oxide.

Chapter 2

Background

Most of the past interest in pure ceria (CeO_2) and doped ceria materials have involved their applications in the automotive industry. Ceria has been studied extensively for its use in catalysis and oxygen sensing in automobile exhaust/emission systems [1]. It has also gained recent attention as a candidate for fuel cell electrodes [1] because of its high electrical conductivity and high thermal stability. These studies have provided significant insight into the understanding of material properties and defect chemistry of ceria and doped ceria materials.

2.1 Material Properties and Defect Chemistry of Ceria

Cerium(IV) oxide (CeO_2) crystallizes in the fluorite crystal structure with lattice constant $a = 5.41134(12)$ Å. The fluorite structure consists of a face-centered cubic (f.c.c.) unit cell of cations with anions occupying the octahedral interstitial sites. This can also be seen as a superposition of an f.c.c. lattice of cations (Ce^{4+}) with lattice constant a, and a simple cubic (s.c.) lattice of anions (O^{2-}) with lattice constant $a/2$. In this structure (shown schematically in Figure 2.1), each cerium cation is coordinated by eight nearest-neighbor oxygen anions, while each oxygen anion is coordinated by four nearest-neighbor cerium cations.

Fluorite structure oxides exhibit similar material properties, such as high radiation tolerance and high thermal stability. Cerium oxide is attractive as a surrogate for uranium oxide because it has the same fluorite crystal structure and many similar material properties, including melting temperature ($UO_2 :\sim 2870\,^\circ C$, $CeO_2 :\sim 2600\,^\circ C$) and thermal diffusivity, and has been well characterized experimentally up to $700\,^\circ C$ [2][3][4][5][6].

While ionic conductivity is believed to be negligible in pure ceria, it increases significantly when ceria is doped with an aliovalent oxide like Y_2O_3 and La_2O_3. The open structure of the fluorite lattice is able to tolerate the high level of atomic disorder that would be introduced by this type of doping. When ceria is doped with a trivalent ion like Lanthanum which forms La_2O_3, a local charge

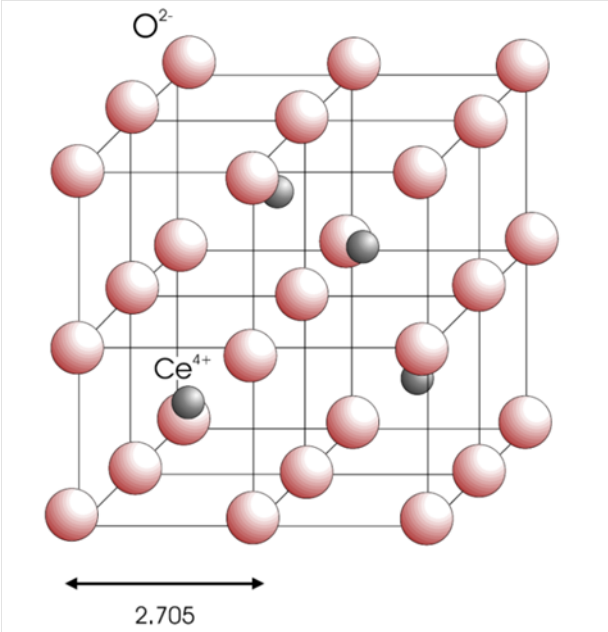

Figure 2.1: Schematic of CeO_2 unit cell.

imbalance is created. The lattice must compensate for this excess negative charge using one of three mechanisms: vacancy compensation, dopant interstitial compensation, and cerium interstitial compensation [7]. The mechanisms can be represented in Kröger-Vink notation:

$$M_2O_3 \overset{CeO_2}{\longleftrightarrow} 2M'_{Ce} + V_O^{\cdot\cdot} + 3O_O^{\times} \tag{2.1}$$

$$2M_2O_3 \overset{CeO_2}{\longleftrightarrow} 3M'_{Ce} + M_i^{\cdot\cdot\cdot} + 6O_O^{\times} \tag{2.2}$$

$$Ce_{Ce}^{\times} + 2O_O^{\times} + 2M_2O_3 \overset{CeO_2}{\longleftrightarrow} 4M'_{Ce} + Ce_i^{\cdot\cdot\cdot\cdot} + 8O_O^{\times} \tag{2.3}$$

In vacancy compensation (Equation 2.1), an anion vacancy is produced for every two dopant ions placed on the host cation sites. In dopant interstitial compensation (Equation 2.2), one dopant cation is placed in on interstitial site for every three dopant cations that are placed on the host cation sites. In cerium interstitial compensation (Equation 2.3), a cerium Frenkel pair is produced and the displaced cerium cation is placed on an interstitial site for every four dopant cations placed on the host cation sites. Empirical calculations performed by Minervini et al. show that for large dopant cations (cation radius > 0.8 Å), vacancy compensation is the preferred charge compensation mechanism [7]. Blumenthal et al. also ruled out the formation of interstitial as the compensation mechanism by measuring true density and comparing it with calculated values [8]. In terms

of dopant solute concentration, the vacancy compensation mechanism can be represented by:

$$xMO_{1.5} + (1-x)CeO_2 \longleftrightarrow xM'_{Ce} + 0.5xV_{\ddot{O}} + (1-x)Ce_{Ce} + (2-0.5x)O_O \quad (2.4)$$

This reactions implies that when $x/2$ moles of dopant oxide (M_2O_3) are added, cation sites are filled with x moles of dopant cation (M^{3+}) and $(1-x)$ moles of host cation (Ce^{4+}), while anion sites are filled with $(2-x/2)$ moles of host anion (O^{2-}) and $x/2$ moles of anion vacancies ($V_{\ddot{O}}$). Therefore, a predictable concentration of oxygen vacancies can be introduced into the crystal by controlling the concentration of the lanthanum dopant added. This process can be used to create an oxygen vacancy environment that is similar hypostoichiometric configuration of pure ceria.

2.2 Previous Computational Work on Ceria related Materials

Simulation techniques have been used to study a wide range of material properties over a wide range of length and time scales, including thermodynamic properties, defect structure and clustering, defect clustering, and transport phenomena. Techniques ranging from Density Functional Theory (DFT), a quantum mechanical theory used to study electron structure over a couple of atoms for a matter of picoseconds, to Finite Element Method (FEM), which can used to study structures as large as a reactor core over a matter of years, each have their own applications and advantages/disadvantages. Figure shows the classic illustration of the application of various modeling techniques of the vast time and length scales of interest. This study links the results from several different time/length scales. At the lowest level, the starting point, are the potentials for the La-doped ceria system that have been developed, either through DFT calculations or from fitting to experimental data. These potentials are fed into Molecular Dynamics (MD) simulations to calculate local configuration-dependent oxygen vacancy migration energies. These migration energies are then fed into KMC simulations to calculate oxygen diffusivity.

Since its development in the 1970s, Molecular Dynamics has been widely used to predict various material properties. However, while it has been used extensively in metal and metal alloy systems, its use in ceramic oxide systems has been comparatively lacking. This is likely due to a lack of confidence in available interatomic potentials. While the potentials in metal and metal alloy systems have been well developed and validated by experimental results, the potentials for ceramic alloys generate results that are not always consistent with experimental data.

Since the the understanding of atomic level interactions is very important in nuclear fuel research, a series of studies have been done to model and understand the thermodynamics and defect chemistry of uranium oxide (UO_2) [9][10][11][12][13][14][15][16][17][18]. An extensive literature review was provided

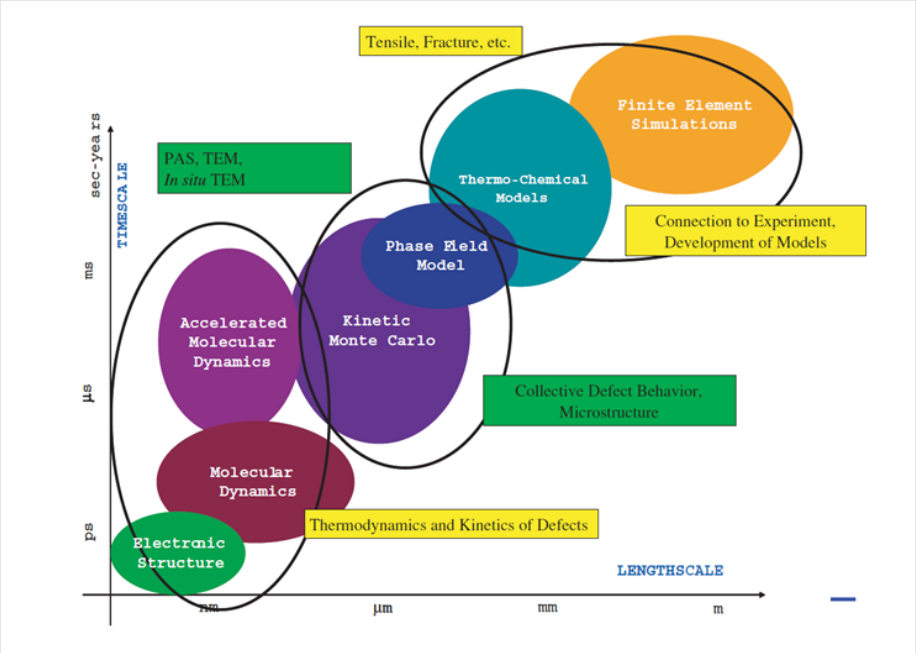

Figure 2.2: Visualization of computational modeling techniques shown over the length and time scales that they apply to.

by Grovers et al. [19][20], where all of the available interatomic potentials on UO_2 were compared. They found that none of the potentials reviewed could predict all of the thermodynamic, mechanical, and defect transport properties. For example, they found that oxygen vacancy migration energies obtained from simulations ranged from 0.1 eV to 0.7 eV, while the experimentally measured energy was found to be 0.5 eV. Similarly, oxygen interstitial migration energies obtained from simulations ranged from 0.1 eV to 3.6 eV, while the experimentally measured energies ranged from 0.9 eV to 1.3 eV. These discrepancies could be attributed to the fact that all of the potentials reviewed were obtained by fitting parameters to experimentally measured properties, such as lattice constant, lattice energy, dielectric constants, etc. As most of these potentials were fitted without considering defect energetics, their inability to accurately predict these values is possibly unavoidable. This survey also demonstrated the importance of testing and validating potentials before using them for predictive purposes.

While there are nineteen interatomic potentials available for UO_2, there are only a few potentials available for CeO_2, and even fewer with dopant parameters. Through an extensive literature survey conducted by Dr. Di Yun, only eight potentials for CeO_2 were found. From those eight, three were selected for comparison in this study. Similar to what was found Grovers' literature review for UO_2 [19][20], all of the potentials found for CeO_2 were in two forms.

The first potential form consists of the addition of a Buckingham term to the basic Coulomb potential. This form can be described by:

$$V_{ij}(r) = \frac{q_i q_j e^2}{4\pi\epsilon_0} + A_{ij} \exp\left(-\frac{r}{\rho_{ij}}\right) - \frac{C_{ij}}{r^6} \tag{2.5}$$

where V_{ij} is the pair potential between atoms i and j, r is the distance between atoms i and j, and q_i,q_j are the charges of atoms i and j, respectively. The pair parameters A_{ij}, ρ_{ij}, and C_{ij} are free parameters that are obtained by fitting to material properties as discussed earlier. This potential can be extended to take the polarization effects of the nucleus and electron shell into account by using the shell-core model developed by Dick and Overhauser [21]. In this model, the charged nucleus and electron cloud are treated as a massless, negatively charged shell bound by a spring to a massive, positively charged core. The spring constant for this model is determined by the polarizability of the atoms being modeled.

The second potential form consists of the addition of a Morse potential to the Buckingham potential form in Equation 2.5, which is used to describe the covalent bonding between the anions and cations. This form can be described by:

$$V_{ij}(r) = \frac{q_i q_j e^2}{4\pi\epsilon_0} + f_0(b_i + b_j) \exp\left(\frac{a_i + a_j - r}{b_i + b_j}\right) - \frac{C_i C_j}{r^6} \tag{2.6}$$

where a_i,a_j,b_i,b_j,c_i, and c_j are the free parameters that are obtained by fitting to material properties as discussed earlier. Table 2.1 lists the fitted parameters for the potentials considered in this study.

Previous work has been done using both modeling and experimental techniques to investigate the clustering of oxygen vacancies around dopant ions in ceria doped with trivalent ions (in this system, the lanthanum trapping effect). Wang et al. showed that in the dilute range, charged dimers ($M'_{Ce} : V_O^{\cdot\cdot}$) form [24]. Gerhardt-Anderson and Nowick extended this work on other $CeO_2 : M_2O_3$, and suggested that conductivity and dimer binding energy vary inversely with dopant cation radius [25]. Kilner and Brook found that due to the size mismatch between the host and dopant cations, elastic strain energy makes a large contribution to the binding energy of the dimer [26]. Subsequent theoretical studies also showed the importance of defect clustering in the determination of free charge carrier concentration in fluorite oxides [27][28][29].

Previous work has also been carried out to validate this type of local configuration dependent kinetic Monte Carlo technique. Murray et al. carried out one of the earliest investigations modeling oxygen vacancy conductivity in yttria-doped cerium oxide [30]. They showed that the oxygen vacancy barrier is sensitive to the local dopant environment and that a first nearest neighbor approximation could generate ionic conductivity results that are somewhat consistent with experimental results. Pornprasertsuk et al. used a similar KMC approach by calculating the binding energies of the oxygen vacancies and dopant ions in Yttria-stabilized zirconia [31]. They showed that the association energy of oxygen vacancies and dopant ions was big enough compared to the oxygen

Table 2.1: Shell-core parameters for Buckingham form pair potentials in CeO_2.

Parameters	Units	Potentials		
		Gotte [22]	Minervini [7]	Sayle [23]
O shell charge	e	-6.5667	-2.04	-6.1
O core charge	e	4.5667	0.04	4.1
O spring constant	eV/Å^2	1759.8	6.3	419.9
Ce shell charge	e	4.6475	4.2	7.7
Ce core charge	e	-0.6475	-0.2	-3.7
Ce spring constant	eV/Å^2	43.451	177.84	291.75
O - O interactions				
A	eV	9533.421	9547.96	22764.3
ρ	Å	0.234	0.2192	0.149
C	Å^6	224.88	32	43.83
O - Ce interactions				
A	eV	755.1311	1809.68	1986.83
ρ	Å	0.429	0.3547	0.35107
C	Å^6	0	20.4	20.4
O - La interactions				
A	eV		2088.79	
ρ	Å		0.346	
C	Å^6		23.25	

vacancy migration barrier to make the vacancy migration energy depend on the local dopant environment. These studies helped to validate the use of local configuration-dependent migration energies in kinetic Monte Carlo simulations, but these codes were written specifically for the systems in question. The goal of this study was to create a generalized code that could be used to simulate arbitrary systems with arbitrarily complex local configurations, so long as the migration energies for these configurations are known.

Chapter 3

Molecular Dynamics

The primary goal of this work is to develop a generalized KMC code, however it is also important to verify and demonstrate the functionality of the code after it is completed. In this study, the code is demonstrated by comparing using diffusivity results generated from KMC simulations of three different interatomic potentials. This requires not only an explanation of the KMC code developed, but also an explanation of the Molecular Dynamics simulations carried out to produce migration energies from the selected potentials.

For this study, molecular dynamics simulations were performed to determine local configuration dependent migration energies using the selected interatomic potentials.

3.1 Background

At the most basic level, Molecular Dynamics (MD) evolves an atomic system using Newton's classical equation of motion:

$$\vec{F} = m\vec{a} \tag{3.1}$$

The system is initialized to some initial state, where each particle in the system has an initial position and an initial velocity. Even without the presence of an external force field (e.x. gravity), each particle experiences some force due to its configurational potential energy:

$$\vec{F}\left(\vec{r}\right) = -\nabla V\left(\vec{r}\right) \tag{3.2}$$

In general, the form of this potential energy function could be arbitrarily complex, but in most cases the potential is simplified down to a two or three particle interaction. This is done primarily because of the computational expense of evaluating the potential energy function. In this study, the selected interatomic potentials are all pairwise interactions. The form of these potentials is given in Equation 2.5, with the corresponding fitted parameters given in Table 2.1. Once

the form of the potential energy function is known, the corresponding force can be derived from Equation 3.2. The acceleration of each particle at a given time step can then be calculated by rearranging Equation 3.1:

$$\vec{a} = \frac{\vec{F}}{m} \tag{3.3}$$

Once the acceleration of each particle is known, an integrator is used to evolve the position and velocity of each of the particles in the system by one time step. One of the most popular integrators, which is also the integrator used by the code in this study, is the Verlet leapfrog algorithm given by:

$$\vec{r}(t+h) = \vec{r}(t) + h\vec{v}\left(t + \frac{1}{2}h\right) \tag{3.4}$$

$$\vec{v}\left(t + \frac{1}{2}h\right) = \vec{v}\left(t - \frac{1}{2}h\right) + h\vec{a}(t) \tag{3.5}$$

where t is the current time and h is the time step used. The name "leapfrog" comes from the fact that the positions and velocities are calculated at staggered time steps, so the positions and velocities appear to "leapfrog" over each other. These calculates are fairly easy to evaluate, so most of the computational time from an MD simulation is spent in the evaluation of forces. Since the configurational force on a single particle depends on the position of every other particle in the system, evaluating this force is very computationally expensive especially as the complexity of the potential is increased.

3.2 Migration Energy Calculations

In order to calculate the oxygen vacancy migration energies from MD, a simulation volume consisting of $4 \times 4 \times 4$ conventional unit cells of CeO_2 (768 atoms) is used. Periodic boundary conditions are applied to this volume to extend the system infinitely in each direction, allowing for the calculation of bulk material properties by eliminating boundary surface effects. A vacancy defect is generated and a local dopant environment is created by substituting La dopant atoms as necessary at the appropriate neighboring cation sites (shown schematically in Figure 3.1). The defect energy is calculated for the system at varying points along the migration pathway (so that the minimum (stable) energy and maximum (saddle-point) energy for the migration can be found. The migration energy for the configuration is then calculated as the difference between the saddle point energy and stable configuration energy.

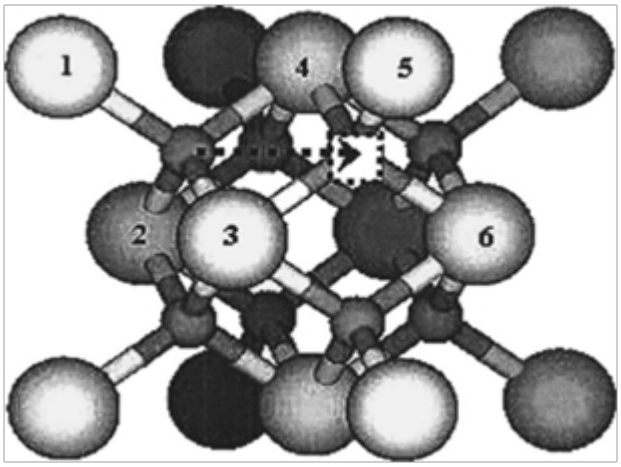

Figure 3.1: Schematic representation of the migrating vacancy and local dopant environment used to model vacancy migration in La-doped CeO_2 [31]. The arrow shows the migration pathway of the oxygen atom (opposite pathway of the oxygen vacancy), the square shows the initial position of the vacancy after migration, and the 6 numbered cation sites show the local configuration sites that are considered.

3.3 GULP Defect Energy Calculations

The MD simulations in this study are performed using the GULP (General Utility Lattice Program) code [32], which uses the Mott-Littleton approach [33] to estimate defect energies. In this approach, the environment around the defects is broken up into three regions by two concentric spheres. In the smallest concentric sphere (Region I), the ions are assumed to be strongly perturbed by

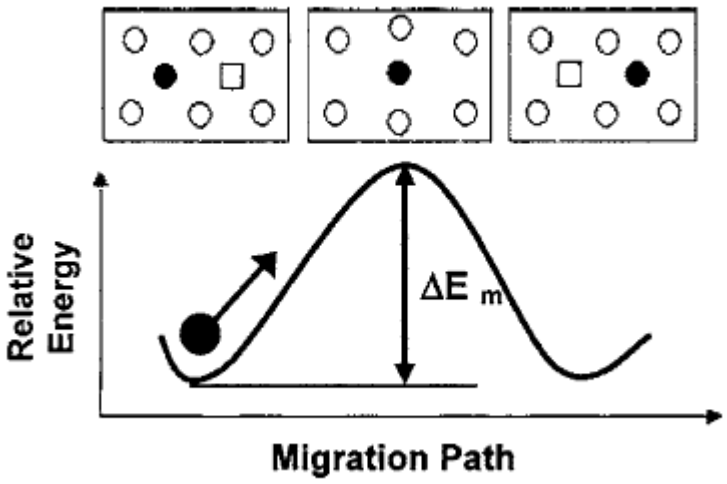

Figure 3.2: Defect energy along the oxygen vacancy migration pathway [31]. Migration energy is calculated as the difference between the saddle point and minimum energies.

the presence of defect and are explicitly relaxed with respect to their Cartesian coordinates. In the second concentric sphere (Region Ia), the ions are assumed to be weakly perturbed by the presence of the defect, so their displacements can be approximated. The area outside the second concentric sphere (Region IIb) is treated as a dielectric medium. Using this approach, both the energy minima (corresponding to stable defect configurations) and energy maximum (corresponding to the migration saddle point plane) can be considered. Since the migration energy is the difference between the saddle point energy and the stable configuration energy, the Mott-Littleton approach is very convenient for this work.

Since varying the Region I and Region II sizes strongly affects the defect energy calculation, it is important to be sure that the these sizes are chosen correctly. As the region sizes increase, the approximates made will become increasingly valid, but at the same time the computational cost will increase substantially. It is important to find a compromise between these two, as region sizes much be large enough that the defect energies are converged, but small enough to be computed in a reasonable amount of time. To find the best region sizes, the defect minimum and saddle point energies were calculated with varying region sizes in the two extreme dopant cases: will no nearest neighbor lanthanum dopant ions and with all six nearest neighbor dopant ions. These simulations were performed by Dr. Di Yun, using yttrium as the dopant, but the results are transferable to other dopant cases. The results of these simulations are given in

Table 3.1: Defect energies for varying Region I and II sizes. Region sizes are in Å, energies are in eV. Results courtesy of Dr. Di Yun.

Sizes	Defect energy				Migration Energy	
	no dopant		6 dopants			
	minimum	saddle	minimum	saddle	no dopant	6 dopants
9-21	15.5	15.864	214.627	215.361	0.364	0.7338
10-23	15.488	15.8462	214.569	215.304	0.3582	0.7351
11-24	15.4737	15.8273	214.553	215.293	0.3536	0.7393
12-25	15.4695	15.8209	214.522	215.262	0.3514	0.7395
13-30	15.4654	15.8165	214.513	215.252	0.3511	0.7386
14-31	15.4616	15.8136	214.501	215.238	0.352	0.7372
15-33	15.4623	15.8114	214.498	215.237	0.3491	0.7389
16-35	15.4624	15.8168	214.495	215.236	0.3544	0.7409

Table 3.1. These results show that defect energy drops with increasing region sizes, as long as the region size is small. After the region sizes reach 12Å-25Å, the energies converge to reasonably stably values and stay stable for further region size increases. The results also show that the oxygen vacancy migration energies become stable above region sizes of 12Å-25Å. Increasing the Region I size from 12Å to 13Å resulted in an almost 43% increase in computational time, so the final region sizes of 12Å-25Å were selected for calculating the migration energies.

3.4 Results

With the optimal Region I and II sizes determined, the defect energies and resulting oxygen vacancy migration energies were calculated using the three selected interatomic potentials. These simulations were performed by Dr. Di Yun. The calculated migration energies are provided in Table 3.2. There is a significant amount of symmetry in this system, so there are only 30 unique configurations that need to be considered, but the results energies for all 64 configurations are presented here for the sake of consistency. These configuration dependent migration energies can now be used in KMC simulations.

Table 3.2: Calculated oxygen vacancy migration energies for all nearest-neighbor cation configurations. Positional numbers are based on the schematic shown in Figure 3.1. Results courtesy of Dr. Di Yun.

1	2	3	4	5	6	Gotte [22]	Minervini [7]	Sayle [23]
Ce	Ce	Ce	Ce	Ce	Ce	0.3276	0.3063	0.7489
La	Ce	Ce	Ce	Ce	Ce	0.1477	0.1278	0.5366
Ce	La	Ce	Ce	Ce	Ce	0.1477	0.1278	0.5366

Continued on Next Page...

Table 3.2 – Continued

1	2	3	4	5	6	Gotte [22]	Minervini [7]	Sayle [23]
Ce	Ce	La	Ce	Ce	Ce	1.0259	1.1403	1.4723
Ce	Ce	Ce	La	Ce	Ce	1.0259	1.1403	1.4723
Ce	Ce	Ce	Ce	La	Ce	0.5232	0.3758	0.6324
Ce	Ce	Ce	Ce	Ce	La	0.5232	0.3758	0.6324
La	La	Ce	Ce	Ce	Ce	0.0666	0.0928	0.1889
La	Ce	La	Ce	Ce	Ce	0.7812	0.9167	1.4425
La	Ce	Ce	La	Ce	Ce	0.7812	0.9167	1.4425
La	Ce	Ce	Ce	La	Ce	0.3461	0.1885	0.4775
La	Ce	Ce	Ce	Ce	La	0.3444	0.2181	0.8875
Ce	La	La	Ce	Ce	Ce	0.7812	0.9167	1.4425
Ce	La	Ce	La	Ce	Ce	0.7812	0.9167	1.4425
Ce	La	Ce	Ce	La	Ce	0.3444	0.2181	0.8875
Ce	La	Ce	Ce	Ce	La	0.3461	0.1885	0.4775
Ce	Ce	La	La	Ce	Ce	1.789	2.1444	1.0307
Ce	Ce	La	Ce	La	Ce	1.193	1.1141	2.231
Ce	Ce	La	Ce	Ce	La	1.193	1.1141	2.231
Ce	Ce	Ce	La	La	Ce	1.193	1.1141	2.231
Ce	Ce	Ce	La	Ce	La	1.193	1.1141	2.231
Ce	Ce	Ce	Ce	La	La	0.7604	0.4723	1.0875
La	La	La	Ce	Ce	Ce	0.487	0.6244	0.8691
La	La	Ce	La	Ce	Ce	0.487	0.6244	0.8691
La	La	Ce	Ce	La	Ce	0.1307	0.1646	0.2703
La	La	Ce	Ce	Ce	La	0.1307	0.1646	0.2703
La	Ce	La	La	Ce	Ce	1.4549	1.8173	1.7113
La	Ce	La	Ce	La	Ce	0.9503	0.8765	1.0021
La	Ce	La	Ce	Ce	La	0.9539	0.9127	1.0109
La	Ce	Ce	La	La	Ce	0.9503	0.8765	1.0021
La	Ce	Ce	La	Ce	La	0.9539	0.9127	1.0109
La	Ce	Ce	Ce	La	La	0.5811	0.2754	0.4875
Ce	La	La	La	Ce	Ce	1.4549	1.8173	1.7113
Ce	La	La	Ce	La	Ce	0.9539	0.9127	1.0109
Ce	La	La	Ce	Ce	La	0.9503	0.8765	1.0021
Ce	La	Ce	La	La	Ce	0.9539	0.9127	1.0109
Ce	La	Ce	La	Ce	La	0.9503	0.8765	1.0021
Ce	La	Ce	Ce	La	La	0.5811	0.2754	0.4875
Ce	Ce	La	La	La	Ce	1.8705	1.949	1.9202
Ce	Ce	La	La	Ce	La	1.8705	1.949	1.9202
Ce	Ce	La	Ce	La	La	1.3835	1.0907	1.3448
Ce	Ce	Ce	La	La	La	1.3835	1.0907	1.3448
La	La	La	La	Ce	Ce	1.0656	1.4646	1.3181
La	La	La	Ce	La	Ce	0.6648	0.6118	0.7476
La	La	La	Ce	Ce	La	0.6648	0.6118	0.7476
La	La	Ce	La	La	Ce	0.6648	0.6118	0.7476

Continued on Next Page...

Table 3.2 – Continued

1	2	3	4	5	6	Gotte [22]	Minervini [7]	Sayle [23]
La	La	Ce	La	Ce	La	0.6648	0.6118	0.7476
La	La	Ce	Ce	La	La	0.3634	0.1166	0.3282
La	Ce	La	La	La	Ce	1.559	1.6416	1.4669
La	Ce	La	La	Ce	La	1.54	1.6403	1.4692
La	Ce	La	Ce	La	La	1.1541	0.881	1.0491
La	Ce	Ce	La	La	La	1.1541	0.881	1.0491
Ce	La	La	La	La	Ce	1.54	1.6403	1.4692
Ce	La	La	La	Ce	La	1.559	1.6416	1.4669
Ce	La	La	Ce	La	La	1.1541	0.881	1.0491
Ce	La	Ce	La	La	La	1.1541	0.881	1.0491
Ce	Ce	La	La	La	La	1.8676	1.6653	1.7879
Ce	La	La	La	La	La	1.5779	1.3804	1.3728
La	Ce	La	La	La	La	1.5779	1.3804	1.3728
La	La	Ce	La	La	La	0.8811	0.6319	0.7995
La	La	La	Ce	La	La	0.8811	0.6319	0.7995
La	La	La	La	Ce	La	1.1748	1.2985	1.0975
La	La	La	La	La	Ce	1.1748	1.2985	1.0975
La	La	La	La	La	La	1.245	1.0795	1.0531

These calculated energies already provide some confirmation of the lanthanum trapping effect. For example, comparing the 2nd and 3rd configurations (which correspond to an oxygen vacancy moving towards a lanthanum cation) with the 6th and 7th configurations (which correspond to an oxygen vacancy moving away from a lanthanum cation) shows that in all three potentials it is easier (lower migration energy) for the oxygen vacancy to move toward a lanthanum cation than it is to move away from a lanthanum cation. These energies confirm the tendency of oxygen vacancies to cluster around the lanthanum ions.

Chapter 4

Kinetic Monte Carlo

4.1 Background

The conventional methods for validating potentials are mainly through molecular dynamics simulations. The potentials are used to evolve a system in MD, and the results are compared with experimental data such as lattice expansion/-contraction with varying temperatures or with varying dopant concentrations. However, due to the time scale limitations of MD simulations (usually limited to evolving on the order of picoseconds), it is difficult to use MD results to compare with experimental results for longer term diffusion or defect structure evolution. Kinetic Monte Carlo (KMC) provides a way to solve this problem. By simulating the atomic jumps between lattice sites as unit events, the time scale can be extended by orders of magnitude compared to the small atomic vibrations simulated in MD.

Kinetic Monte Carlo simulations follow a relatively straightforward algorithm. For a system where some processes (in this case atomic migrations) can occur with known rates r_i, the KMC evolution of the system is governed by the following algorithm:

1. Initialize simulation time to $t = 0$.
2. Form a list of all possible rates in the system r_i.
3. calculate the cumulative function $R_i = \sum_{j=1}^{i} r_i$ for each rate $i = 1, \ldots, N$. Let $R = R_N$ be the total rate sum. (This can be thought of as calculating a discrete cumulative distribution function from the discrete probability density function r_i, although this is not technically true because the probability density function is not normalized.)
4. Generate a uniform random number $\rho_1 \in (0, 1]$.
5. Find the event to carry out i by finding the i such that $R_{i-1} < \rho_1 R \leq R_i$. (This can be thought of as inverting the previously calculated cumulative distribution function.)

6. Carry out event i.

7. Recalculate all rates r_i that may have changed due the event that was carried out. If necessary, remove or add any new rates to the list of possible rates.

8. Generate a new random number $\rho_2 \in (0, 1]$.

9. Update the simulation time with $t = t + \Delta t$, where $\Delta t = -\ln(\rho_2)/R$.

10. Return to step 2 and repeat until the desired number of steps have been carried out.

In this context, the rates in question are the oxygen vacancy migration rates. These migration events are thermally activated, with a migration rate given by [34]:

$$r_i = \nu_0 \exp\left(-\frac{\Delta E_m^i}{k_b T}\right) \tag{4.1}$$

where r_i is the rate of migration for the event i, ν_0 is the migration attempt frequency, taken to be $1.0 \times 10^{13} Hz$, ΔE_m^i is the migration energy for the event i, k_b is the Boltzmann constant, and T is the absolute temperature of the system.

4.2 Original Code

In order to facilitate the development of the generalized code, a very basic KMC simulation code was provided by Dr. Chaitanya Deo at the Georgia Institute of Technology. This code was written specifically to study oxygen interstitial diffusion in uranium oxide. The design of this code was such that attempting to extend it would have been more trouble that it was worth, so a new code was written based loosely on the structure of the original code. In order to better explain the work done in this study, the structure of this original code is explained here. In these explanations certain words are capitalized to indicate that they refer to the corresponding data object in the code: Entity, Action, Event, Arrangement, Restriction.

This code starts by reading in the system parameters from an input file. These values include things like simulation system dimensions, system temperature, number of steps to run, and particle populations, and are specified as key-value pairs (see Listing 4.1). A list is created for the particle objects, and for each particle, a random position initial position is generated such that the position falls on the required sublattice for that particle. Both the lattice type and sublattice types for each particle type are hard-coded, which presents a problem for generalization. If a new interacting particle is to be added to the system, the code must be modified and rebuilt so the input parsing routine can read the particle's population and so the initial position generator can tell which sublattice the particle should have a position on.

Listing 4.1: Original code configuration input file.

```
## kmc control input file
size_X   15
size_Y   15
size_Z   15
Temperature       1073
number_vacancy   0
number_hydrogen 0
number_IOx        34
monte_carlo_steps         500000
boltzmann         8.61738e-5
lattice 1
```

Listing 4.2: Original code actions input

```
#[0]type [1]Entity [2]Sublattice [3]E [4]v0 [5-7] d[012]
migration   3   1   1.30   1e13   2   0   2
migration   3   1   1.30   1e13   2   0   -2
migration   3   1   1.30   1e13   2   2   0
migration   3   1   1.30   1e13   2   -2   0
migration   3   1   1.30   1e13   0   -2   2
migration   3   1   1.30   1e13   0   2   2
migration   3   1   1.30   1e13   0   -2   -2
migration   3   1   1.30   1e13   0   2   -2
migration   3   1   1.30   1e13   -2   0   2
migration   3   1   1.30   1e13   -2   0   -2
migration   3   1   1.30   1e13   -2   -2   0
migration   3   1   1.30   1e13   -2   2   0
```

After the list of particles objects is created, the list of possible migration actions is read in from an input file. This includes the associated particle type-/sublattice, attempt frequency, migration direction, and migration energy, and is specified in a multicolumn format (see Listing 4.2). When this data is read in, it is stored as a list of Action objects, which contain the specified information, as well as the migration rate for this action. Since these migration actions are assumed not to depend on the local configuration, once the action has been read it, all of the required information to calculate the migration rate from Equation 4.1 is known, so the migration rates are calculated and stored with the Action object. This presents a problem for generalization, as when the energy is allowed to depend on the local configuration, the migration rates could potentially change value at every time step for each particle in the system.

After both the Entity list and Action list have been initialized, the system is ready to be evolved. An event catalog of all of the possible migration events (step 2 of the general KMC algorithm in Section 4.1) is generated by scanning

through the Entity list and Action list and checking each possible combination. For each combination of Entity and Action, a list of checks is performed to see if the pair is a valid migration event. This includes checking that the particle type that the action acts upon and the particle type of the paired Entity match and checking that the sublattice that the actions acts upon and the sublattice of the paired particle match. This is necessary as the lists can contain Entities and Actions for different particles on different sublattices, and it is important that the Entity and Actions selected for an Event are compatible (for example, attempting to apply an oxygen vacancy migration action on the fluorite tetrahedral interstitial sublattice to an oxygen interstitial on the fluorite octahedral interstitial sublattice would be a mistake). A check is also performed on the proposed final position of the particle to ensure that the final position is currently unoccupied (as moving a particle to a position where a particle already exists is not possible). As it will be shown later, this step, while necessary, is very computationally expensive because of the way the Entity objects are stored. If an Entity/Action pair passes all of the checks, it is considered a possible event, and Event object consisting of the Entity/Action pair is added to the event catalog.

Once the list of all possible events has been generated, a single event is chosen to be carried out. Since the migration energy for each event was assumed to be constant, the rates for all of these events are already known, so the partial sums of these rates are calculated and an event is randomly selected (steps 3-5 of the general KMC algorithm in Section 4.1). The event is carried out (step 6 of the general KMC algorithm in Section 4.1) by applying the Action to the Entity's position and updating the Entity object's internal position information. The simulation time is then updated using according to steps 8-9 of the general KMC algorithm in Section 4.1. At this point, the event catalog is no longer consistent with the system, so it is emptied in preparation for the next time step. The system loops over this process for the desired number of time steps.

Once the system has finished its evolution (i.e. the specified number of time steps have been carried out), the diffusion coefficient is calculated by:

$$D = \frac{< x^2 >}{6t} \tag{4.2}$$

where D is the diffusion coefficient, $< x^2 >$ is the average squared displacement of the diffusing particles, and t is the calculated simulation time.

4.3 Generalized KMC Code

While the overall idea behind the original code was well thought out, the implementation of the idea was not going to be sufficient for extension in the areas this work was looking to study. As a result, a new code was written that used the same overall idea from the original code (representing particles and actions Entity and Action objects, selectively pairing them into Events, then picking

one and evolving the system), but restructured in a way that corrected the limitations described in Section 4.2, along with other implementational details. The major changes and extensions are outline here.

4.3.1 Positional Grid

Of primary concern was the handling of the particle positions. In the original code, while it was straightforward to get a specific particle's position, the reverse was not so straightforward. This caused a significant increase in computational complexity in the overall code run, specifically because of the Event validation checks. After a specific Entity and Action are confirmed to be a compatible pair, a check is performed to see if the final position after migration is already occupied. If it is occupied, the move is rejected, and if it is not, the move is accepted. Unfortunately, in the original code there is no direct way to resolve an Entity object from a given lattice position. Thus, in order to verify that a given position is unoccupied, the list of Entities must be scanned sequentially, with each each Entity's position compared to the position in question. If the position is indeed unoccupied, this method requires iteration of the entire list to confirm it as such. Because of this, the occupancy check is considered to have linear complexity with respect to the particle count. Since this check is performed while the event catalog is being constructed, which in itself is of linear complexity with respect to the particle count because each Entity must be checked against every Action, the construction of the Event catalog ends up being of quadratic complexity with respect to the particle count. Since the event catalog is reconstructed at each KMC time step, this quadratic complexity significantly impacts the overall performance of the code.

This quadratic complexity problem is addressed in the new KMC code by the addition of a bookkeeping data structure. As the code is written in C++, access to data structures' memory addresses (pointers) is permitted. With this capability available, what is essentially a map from 3-dimensional positions to Entity pointers is constructed. Since the code is Lattice Kinetic Monte Carlo, particles can only exist on specific lattice sites. As such, the lattice positions can be scaled in such a way as to make all the lattice coordinates non-negative integers. This allows the position map to be created as a list of lists of lists of Entity pointers. Since list access is constant time, this reduces the occupancy check complexity from linear to constant time with respect to particle count, which reduces the overall code complexity from quadratic to linear with respect to particle count. This structure is initialized as the particles are added to the Entity list at the beginning of the simulation, and with the constant time access can be easily updated after each migration event. This results in a huge performance gain compared to the original code, as simulations that would take hours in the original code could finish in minutes in the new code. This performance gain is shown explicitly in Section 5.1.

4.3.2 Per-Event Migration Rates

A major limitation in the design of the original code was the lack of proper per-event rates. In the original code, it was assumed that migration energies were constant for a given Action. For example, a oxygen vacancy attempting to the neighboring interstitial site in the +z direction will have the same migration energy regardless of the migrating vacancy's local environment. Since the migration energy associated with a given action is assumed never to change, the migration rate is only calculated once while the system is being initialized, and both the migration energy and the resulting migration rate for each action are stored as properties of the Action. This limitation was a major problem for this study, as the ability to change migration energies based on the local environment is what this study is all about.

Since actions in the new code are allowed to have different energies based on the local environment, the data structures and the main algorithm need to be changed in the new code. The Event class is extended to hold a migration energy and migration rate, in addition to the references to the Entity and Action it is associated with. As the migration energy and migration rate are now associated with the Events in the event catalog, they need to be calculated each time the event catalog is populated. An additional step is added to the original code algorithm after the event catalog is populated to iterative over the events in the catalog and for each Event, determine the correct migration energy and calculate the corresponding migration rate.

4.3.3 XML Input/Output

In order to facilitate the generalization both to allow arbitrary particle types to be placed in the system and to allow arbitrarily complex local environments and corresponding energies to be specified, the systems for reading and writing data needs to be modified. In the original code, configuration entries were specified as key/value pairs (see Listing 4.1), while the action entries were specified in a multicolumn format (see Listing 4.2). These formats are sufficient for the simplified system the original code was designed for, but it is not sufficient for the complex interactions that need to be specified for this study. Attempting to use these simple tab-delimited formats for complex specifications would result in extremely complex input specifications and even more complex parsers and generators to write them. As I/O specification becomes more complicated it becomes very difficult to read/write input/output files without making mistakes. It also becomes more difficult to access these files programmatically, which makes both generating series of input files and analyzing the resulting output files more difficult.

This problem is solved by the introduction of XML as the method of data transport. XML schemas can be written for arbitrary data transport, which makes it ideal for customized data transport in custom codes such as this. The XML protocol is also very well supported, with XML interfaces either built into or readily available for almost every commonly used programming language. As

such, it becomes almost trivial to read and write data from the program (e.g. instead of reading the system temperature from the output file by iterating down the lines of the file to the third line, then reading the fifth column of data or whatever, the entire data file can be automatically loaded into a data structure and the temperature can be obtained by some logical data structure accessors like `$data->{config}->{temperature}`).

Customized XML schemas for the various input and output files were developed, so input can be generated easily from any programming language with an XML module (almost every commonly used language). An additional input file for Entity naming and sublattice maps we also developed so that arbitrary particle types can be specified and properly added to the system without any modification of the code (see Listing 4.3), Also, when combined with bi-directional maps added to the code, this allows for logical particle names to be used in the input and output files instead of their internal numerical representation (e.g. the tetrahedral vacancy particle type can be represented by the logical name `tet_vacancy` instead of its internal numerical identifier 4). The configuration input file specifies all of the system configuration parameters like the original code (see Listing 4.4). The actions input file specifies the information related to each migration action (see Listing 4.5), and is extended in the next section to support arbitrary local configuration dependent migration energies. Finally, the output file contains the calculated diffusion coefficient of each of the particles in the system, along with various runtime statistics (see Listing 4.6).

Listing 4.3: Generalized code sample entities input

```
   <?xml version="1.0" encoding="utf-8"?>
<entities>
  <entity>
    <name>sub_lanthanum</name>
    <sublattice>substitutional</sublattice>
  </entity>
  <entity>
    <name>tet_vacancy</name>
    <sublattice>tetrahedral</sublattice>
  </entity>
</entities>
```

Listing 4.4: Generalized code sample configuration input

```
   <?xml version="1.0" encoding="utf-8"?>
<configuration>
  <boltzmann_constant>8.61738e-5</boltzmann_constant>
  <dimensions>
    <x>15</x>
    <y>15</y>
    <z>15</z>
  </dimensions>
  <initial_populations>
    <population>
      <count>200</count>
      <entity>sub_lanthanum</entity>
    </population>
    <population>
      <count>100</count>
      <entity>tet_vacancy</entity>
    </population>
  </initial_populations>
  <lattice_parameter>5.411e-8</lattice_parameter>
  <lattice_type>face_centered_cubic</lattice_type>
  <simulation_steps>10000</simulation_steps>
  <temperature>1073</temperature>
</configuration>
```

Listing 4.5: Generalized code sample actions input (truncated to save space)

```
   <?xml version="1.0" encoding="utf-8"?>
<actions>
  <action>
    <direction>
      <x>0</x>
      <y>0</y>
      <z>2</z>
    </direction>
    <energy>0.7489</energy>
    <entity>tet_vacancy</entity>
    <frequency>1.0e13</frequency>
    <sublattice>tetrahedral</sublattice>
    <type>migration</type>
  </action>
  <action>
    <direction>
      <x>0</x>
      <y>0</y>
      <z>-2</z>
    </direction>
    <energy>0.7489</energy>
    <entity>tet_vacancy</entity>
    <frequency>1.0e13</frequency>
    <sublattice>tetrahedral</sublattice>
    <type>migration</type>
  </action>
  <action>
    <direction>
      <x>2</x>
      <y>0</y>
      <z>0</z>
    </direction>
    <energy>0.7489</energy>
    <entity>tet_vacancy</entity>
    <frequency>1.0e13</frequency>
    <sublattice>tetrahedral</sublattice>
    <type>migration</type>
  </action>
</actions>
```

Listing 4.6: Generalized code sample output

```
   <?xml version="1.0" encoding="utf-8"?>
<results>
  <configuration>
    <temperature>1073</temperature>
    <lattice_type>face_centered_cubic</lattice_type>
    <lattice_parameter>5.411e-008</lattice_parameter>
    <dimensions>
      <x>15</x>
      <y>15</y>
      <z>15</z>
    </dimensions>
  </configuration>
  <entities>
    <entity>
      <name>sub_lanthanum</name>
      <initial_population>200</initial_population>
      <diffusivity>0</diffusivity>
      <square_displacement>0</square_displacement>
      <rms_displacement>0</rms_displacement>
    </entity>
    <entity>
      <name>tet_vacancy</name>
      <initial_population>100</initial_population>
      <diffusivity>8.821e-007</diffusivity>
      <square_displacement>1.570e-012</square_displacement>
      <rms_displacement>1.253e-007</rms_displacement>
    </entity>
  </entities>
  <time_series />
  <run_statistics>
    <run_time>552</run_time>
    <simulation_steps>10000</simulation_steps>
    <simulation_time>2.96782e-009</simulation_time>
    <average_time_step>2.96782e-013</average_time_step>
    <max_time_step>3.69887e-012</max_time_step>
    <min_time_step>2.8271e-017</min_time_step>
    <action_list_size>6</action_list_size>
    <arrangement_count>756</arrangement_count>
    <avg_event_catalog_size>520</avg_event_catalog_size>
    <max_event_catalog_size>594</max_event_catalog_size>
    <min_event_catalog_size>500</min_event_catalog_size>
  </run_statistics>
</results>
```

4.3.4 Arbitrary Local Environments

In order to support migration energies that depend on a local environment, there needs to be a mechanism to check these local environments. Since this involves looking for particles that may or may not occupy various local lattice sites, this procedure will have the same lookup efficiency issues as the migration site occupancy check. Fortunately, the implementation of the positional grid map discussed in Section 4.3.1 makes this lookup relatively efficient. While it is possible to hardcode specific neighbor sites and entity types in the code without too much additional work, this requires the code be modified every time different local configurations are to be simulated. Thus, a more general approach needed to be determined.

Many different ideas were considered to implement this, but only one was sufficiently general to allow an arbitrary number of arbitrary configurations to be specified completely in the action input file without the need to modify or rebuild the code. This method is described here.

To begin, in a given local configuration (referred to as an Arrangement), each position to be considered is specified (referred to as a Restriction). A Restriction consists of a position/entity-type pair, where the position specifies the 3-dimensional position relative to the initial position of the migrating particle, and the entity-type is the type of Entity that should be present in that position. Each Arrangement contains an arbitrary length list of these Restrictions, along with the migration energy that corresponds to this set of Restrictions. An Arrangement can thus consider any arbitrary local configuration that is specified. The Action class is extended to contain an arbitrary length list of these Arrangements. Each Action can thus consider any number of arbitrary local configuration specified. For the local environment used in this study (Figure 3.1), there are 6 cation sites that influence the migration energy, so each Action contains 64 Arrangements corresponding to the 64 different possible configurations of the 6 cation sites, where each of these Arrangements contains 6 Restrictions that specify the positions and entity-types of the configuration, along with the migration energy that correspond to that configuration. A sample Action containing a local arrangement is shown in Listing 4.7 (note that since only the defects are simulated, not the background CeO_2 lattice, an "empty" cation site corresponds to a Ce atom). Constructing such an action file can be quite daunting if done manually, but since it is written in XML, it can be constructed logically in a programming language data structure and then exported to XML using the language's XML interface.

After this information is loaded into the Action data structure, it can be used during the migration rate calculation step discussed in Section 4.3.2. During this step, rather than just taking the migration base migration energy specified in the action file, the code first checks the associated Arrangements to see if any of them apply. It does this by iterating over the list of Arrangements sequentially, and using the energy associated with the first Arrangement that applies to the event. It checks each Arrangement by iterating over the list of Restrictions for the Arrangement and checking the positions and entity types specified by these

Restrictions. If any of the Restrictions do not match, the Arrangement does not apply to the Event, and if all of the Restrictions match, the Arrangement does apply to the Even. If after checking all of the Arrangements none of them are found to match, the base migration energy (this can thought of as the "default" energy) for the Action is used. Once the migration energy has been determined, the migration rate is calculated in the normal way from Equation 4.1.

Listing 4.7: Generalized code sample action with arrangement

```
   <?xml version="1.0" encoding="utf-8"?>
<actions>
  <action>
    <arrangements>
      <arrangement>
        <energy>0.5366</energy>
        <restrictions>
          <restriction>
            <direction>
              <x>-1</x>
              <y>1</y>
              <z>-1</z>
            </direction>
            <entity>no_entity</entity>
          </restriction>
          <restriction>
            <direction>
              <x>1</x>
              <y>-1</y>
              <z>-1</z>
            </direction>
            <entity>sub_lanthanum</entity>
          </restriction>
          <restriction>
            <direction>
              <x>-1</x>
              <y>-1</y>
              <z>1</z>
            </direction>
            <entity>no_entity</entity>
          </restriction>
        </restrictions>
      </arrangement>
    </arrangements>
    <direction>
      <x>0</x>
      <y>0</y>
      <z>2</z>
    </direction>
    <energy>0.7489</energy>
    <entity>tet_vacancy</entity>
    <frequency>1.0e13</frequency>
    <sublattice>tetrahedral</sublattice>
    <type>migration</type>
  </action>
</actions>
```

Chapter 5

Results and Discussion

5.1 Computational Complexity

As was discussed in Section 4.3.1, because of the way positions are stored in the original code there is no direct way to resolve a lattice position to a particle, which results in quadratic complexity with respect to the particle count. This is solved in the generalized code by the introduction of a position-entity map, which reduces position lookups to constant time, and overall code complexity to linear time. To demonstrate this, the generalized code is used to run the same simulation environment that the original code is capable of running. This is a fluorite UO_2 lattice with interstitial oxygen atoms added in various concentrations to study hyperstoichiometric effects. The migration energy and attempt frequency are taken to be the same as what was provided with the original code ($E_m = 1.1$ eV, $\nu_0 = 1 \times 10^{13}$ Hz). The simulation system was taken to be a $15 \times 15 \times 15$ unit cell cubic system with periodic boundary conditions. The runtime results for these simulations are provided in a linear scale in Figure 5.1 to show the complexity trends, and in a logarithmic scale in Figure 5.2 for a more useful comparison of the actual time values. These results show the enormous improvement in complexity and runtime, which is critical to the performance of the local configuration-dependent simulations. Without this improvement this improvement the more complex simulations would extraordinary long times to complete.

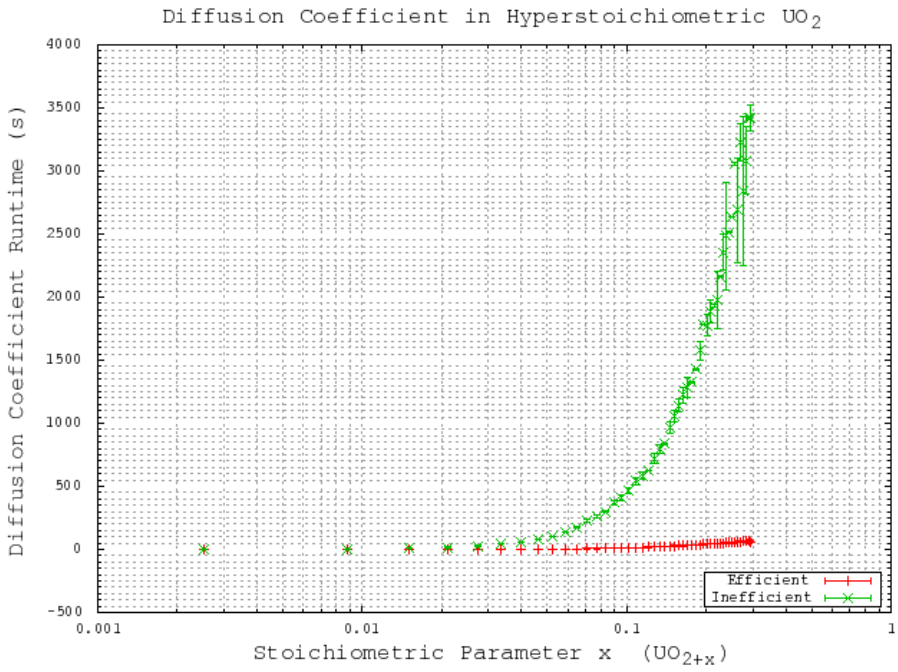

Figure 5.1: Comparison of computational complexity with respect to the particle count (show here as a function of the stoichiometric parameter) shown on a linear scale to demonstrate complexity trends. Here "Inefficient" refers to the original code algorithm, while "Efficient" refers to the generalized code algorithm.

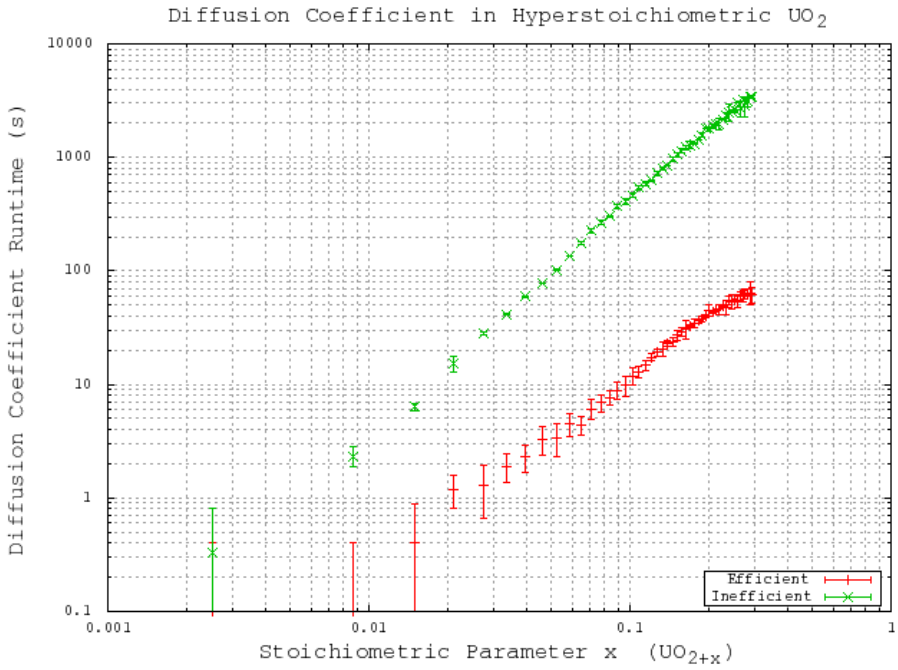

Figure 5.2: Comparison of computational complexity with respect to the particle count (show here as a function of the stoichiometric parameter) shown on a logarithmic scale for accurate comparison of runtime values. Here "Inefficient" refers to the original code algorithm, while "Efficient" refers to the generalized code algorithm.

5.2 Code Verification

First, it is important to verify that the generalized code is able to reproduce the results of the original code, to confirm that the core functionality of the generalized code is consistent with that of the original code. To confirm this, the diffusivity results from the computation complexity simulations are compared. These simulations provided the oxygen interstitial diffusion coefficient D_i from Equation (4.2). The oxygen self-diffusivity is then calculated by:

$$D_O = [O_i'']D_i \tag{5.1}$$

where D_O is the oxygen self-diffusivity, $[O_i'']$ is the oxygen interstitial mole fraction, and D_i is the oxygen interstitial diffusivity. The computed results from the original code and from the generalized code are compared with each other, and with experimentally measures values provided by Contamin [35] and Murch [36]. The results are given in Figure 5.3. These results show excellent agreement between the calculated results from the original code and generalized code, and between the calculated results and the experimentally measured values.

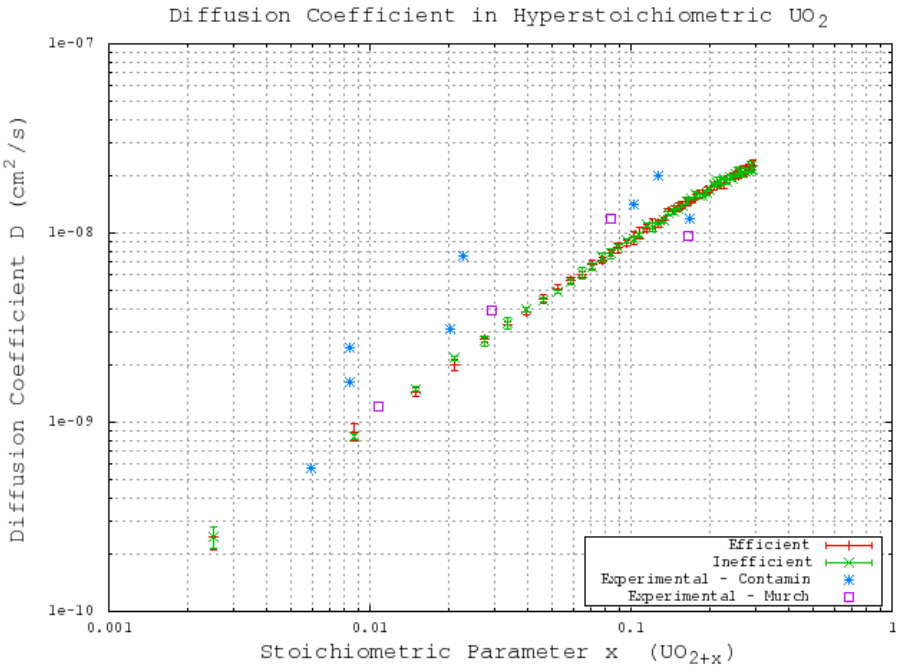

Figure 5.3: Comparison of calculated and experimentally measured oxygen self-diffusivities in hyperstoichoimetric UO_2. Here "Inefficient" refers to the original code algorithm, while "Efficient" refers to the generalized code algorithm.

With the core functionality verified, it is also important to verify the most important extension to the original code: the local configuration depedent energies. The code to read the complex XML input data and store it in the Action data structures, as well as the code to scan for these configurations during migration energy determination is quite extensive. It is important confirm that these configurations are being recognized correctly by the code as it checks through them. To verify this, the code was modified to output the configuration found as it scans through the local environment of each particle. These counts were aggregated for several different non-stoichiometric values x in $La_xCe_{1-x}O_{2-x/2}$. The distribution of these configuration changes according to the energies, to confirm the expected initial random distributions, only the configurations for the initial time step are considered. These results are given in Figure 5.4. The configuration number in these figures corresponds to the row number of the configuration at specified in Table 3.2, and can be thought of as increasing with increasing number of neighbor lanthanum cations. These results show that as expected, for low dopant concentrations the configurations found are concentrated around the 0-1 neighbor lanthanum cation configurations, and as the dopant concentration increases, the configuration distribution spreads out to include the higher neighbor lanthanum cation configurations.

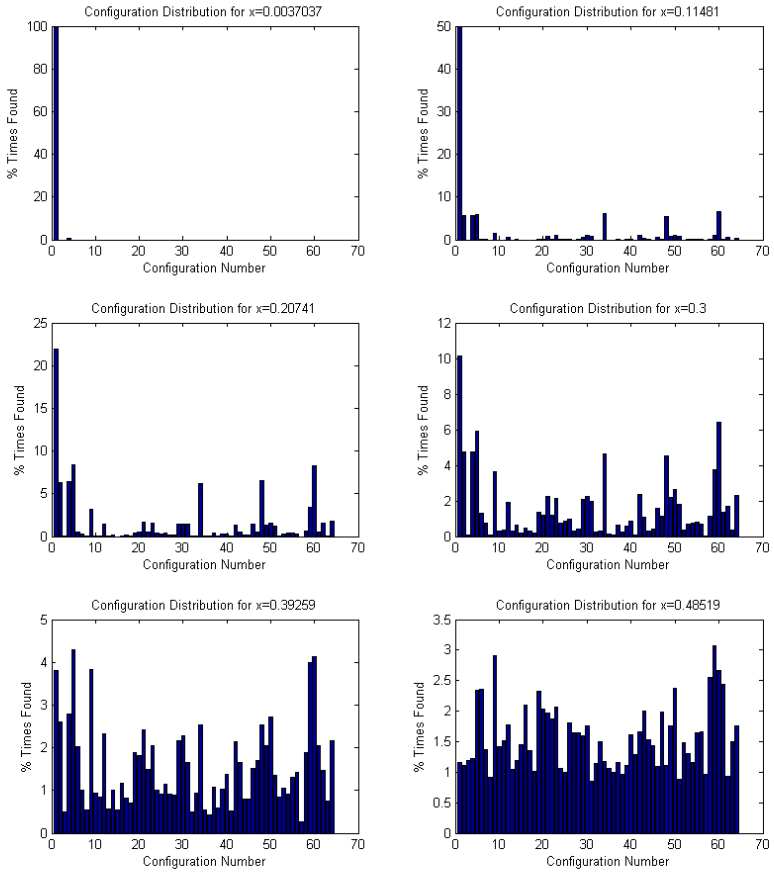

Figure 5.4: Configuration distribution verification for initial configurations at increasing non-stoichiometries. The configuration number corresponds to the row number of the configuration at specified in Table 3.2.

5.3 Potential Validation

To analyze the validity of the potentials described in Table 2.1, KMC simulations are performed using the local configuration dependent migrations energies calculated from these potentials. These simulations are performed on a fluorite UO_2 lattice with lanthanum added to the system in various concentrations to study hypostoichiometric effects. The simulation system was taken to be a $15 \times 15 \times 15$ unit cell cubic system with periodic boundary conditions at 800 °C. Oxygen tetrahedral vacancies and lanthanum substitutional atoms are added according to the vacancy compensation mechanism in Equation 2.4. The local configuration depenent vacancy migration energies for each potential are taken from Table 3.2. To be sure that the event catalog and migration energies are properly sampled, it is important to select a sufficiently large number of simulations steps to run, especially for high dopant concentrations. In order to ensure that a sufficiently large number of steps is used in the validation study, the simulations for one of the potentials were run with increasing simulation steps until the solutions converged. The results are shown in Figure 5.5, and indicate that $500,000$ KMC steps is sufficient to get converged results for this system. While this study was only performed for one potential (Gotte [22]), it is assumed that the conclusion can be transfered to the other potentials since the simulations for these potentials use the same particle concentrations and configurational complexity.

With the sufficiently large number of KMC steps determined, the KMC simulations for varying dopant concentrations are performed using the migration energies derived for each of the three potentials (Table 3.2). The same simulations are performed using a constant migration energy that corresponds to the case of zero neighbor lanthanum cations (pure hypostoichiometric ceria) in order to demonstrate the effect of the lanthanum interactions. The results of these simulations are given in Figure 5.7 (Gotte [22]), Figure 5.8 (Minervini [7]), and Figure 5.9 (Sayle [23]). Diffusivity results from all three potentials clearly show the desired lanthanum trapping effect, so in this sense they are all reasonable potentials. However, each potential shows peak oxygen diffusivity at different dopant concentrations. To see which produces the most realistic result, the peak diffusivities are compared to experimental results by Faber et. al [37]. Faber et al. studied ionic conductivity in ceria doped with several different dopant species. From the ionic conductivity, they calculated the effective activation energy for diffusion. The calculated values for lanthanum doped ceria are shown in Figure 5.6. The overall effective activation energy essentially determines how easy it is for ions to diffuse, so from this it is inferred that the minimum overall activation energy should roughly correspond to the peak oxygen diffusivity. The minimum activation energy as found by Faber occurs at around 5% lanthanum concentration, while the peak diffusivities from Gotte, Minervini, and Sayle occur around 12%, 20%, and 23% respectively. Based on these results, it is concluded that out of the three, the Gotte potential yielded the most realistic diffusion curve.

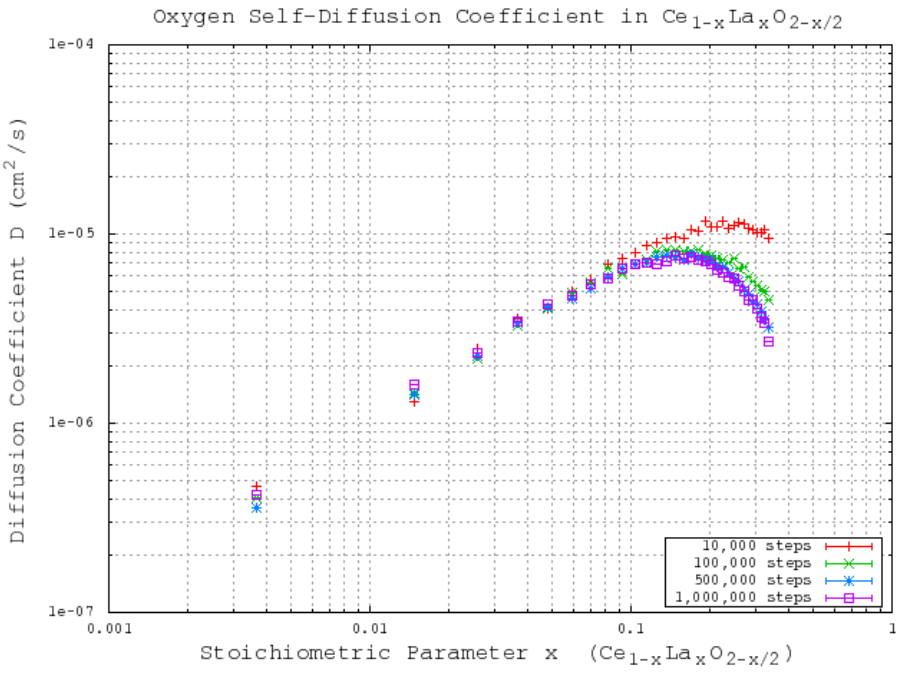

Figure 5.5: KMC simulation steps comparison using migration energies derived from the Gotte [22] potential.

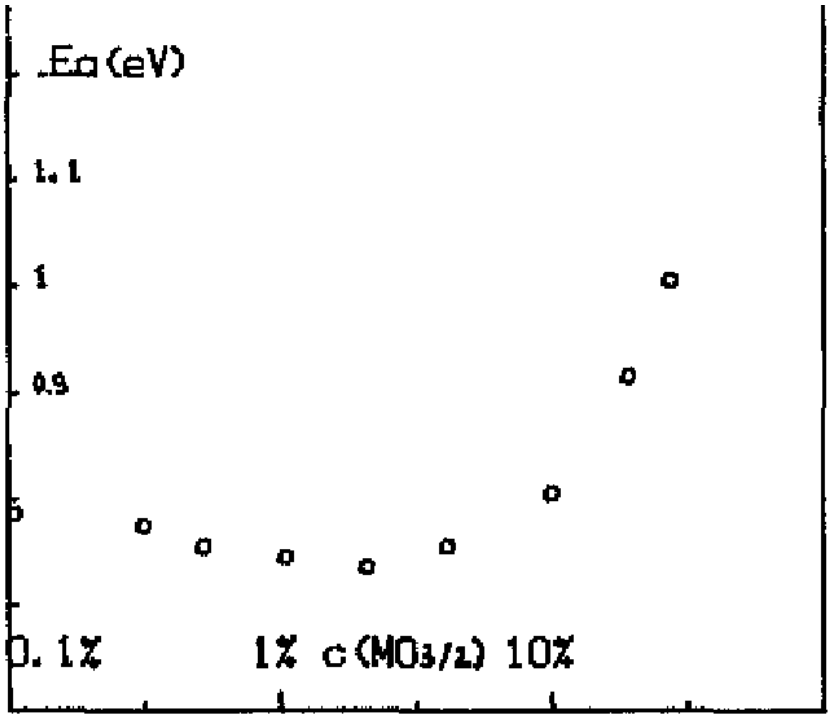

Figure 5.6: Effective activation energy as a function of non-stoichiometry in La-doped ceria [37].

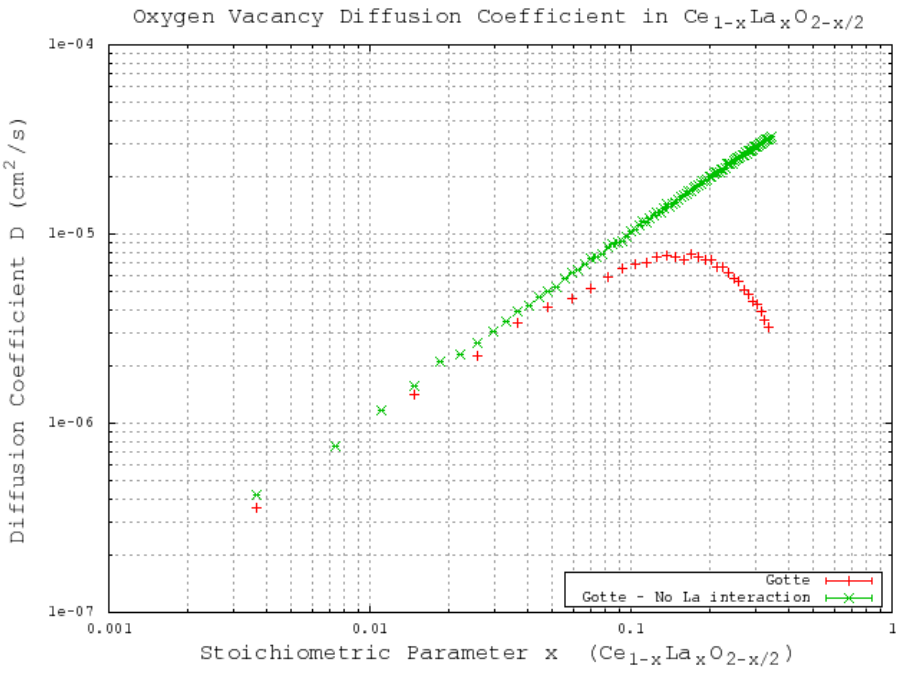

Figure 5.7: KMC simulation results for migration energies derived from the Gotte [22] potential.

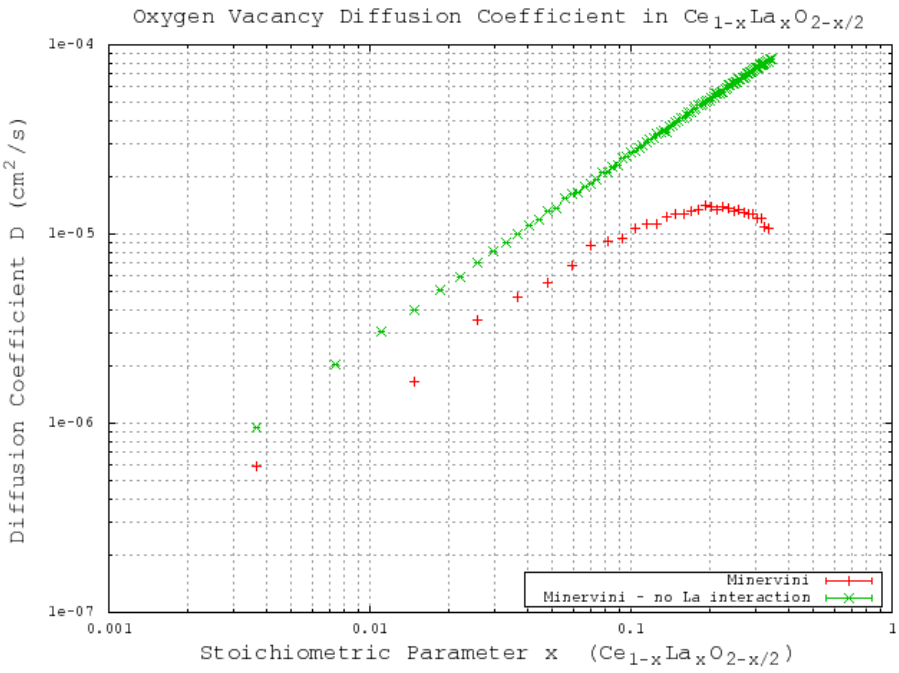

Figure 5.8: KMC simulation results for migration energies derived from the Minervini [7] potential.

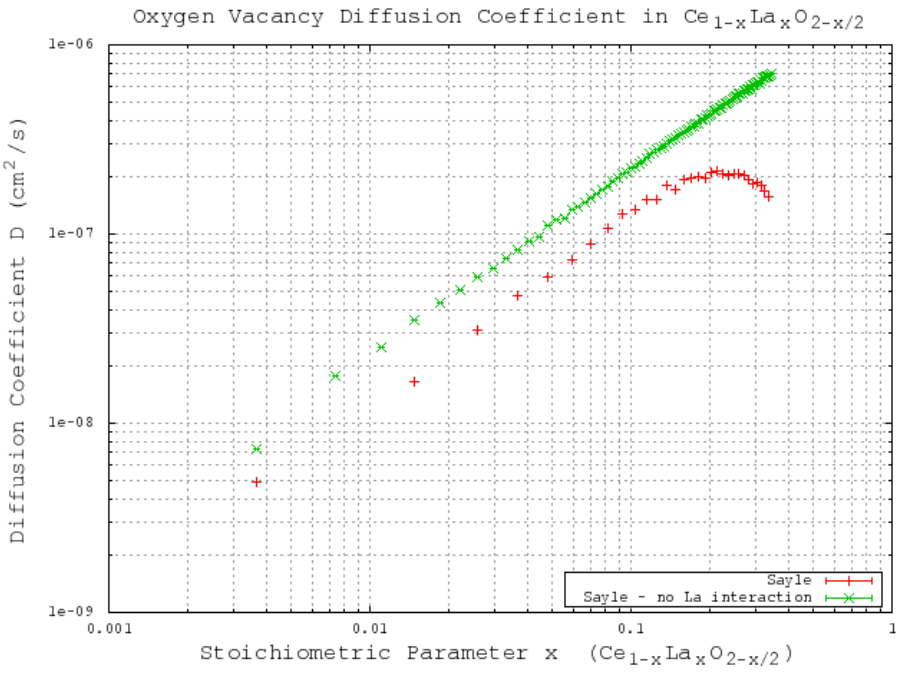

Figure 5.9: KMC simulation results for migration energies derived from the Sayle [23] potential.

Chapter 6

Conclusions and Future Work

To conclude, a kinetic Monte Carlo code has been developed and has been used in validation studies on several interatomic potentials. The following provides a brief summary of the conclusions drawn from this study:

1. A generalized Kinetic Monte Carlo code has been developed to simulate diffusion of defects whose migration energies can vary according to arbitrarily specified local atomic configurations.

2. The Molecular Dynamics code GULP has been used to calculate local configuration dependent migration energies from three different interatomic potentials for lanthanum doped ceria.

3. Kinetic Monte Carlo simulations have been carried out on oxygen vacancy diffusion in lanthanum doped ceria.

4. The lanthanum trapping effect in lanthanum doped ceria has been confirmed both from Molecular Dynamics results and from Kinetic Monte Carlo results.

5. Diffusion results from three different interatomic potentials have been compared to each other and to experimental activation energy data.

There are also some places for improvement and extension in the KMC simulation and the generalized KMC code. The following provides a brief list of possible future work:

- Neighbor tables could be added to each Entity that would contain the atoms that that could be affected by a migration event. This would allow the Event catalog to be selectively modified at the end of each time step instead of being completely rebuilt.

- Oxygen vacancies have a tendency to cluster and form voids. A similar series of Molecular Dynamics calculations could be performed on vacancy-vacancy interactions and these interactions could be added to the simulation.
- Support for additional event types such as particle production (e.g. as a result of radiation damage or fission events) or particle recombination (e.g. Frenkel pair recombination) could be added.
- The use of memory pointers makes parallelization in non-shared memory paradigms difficult. Alternatives that require more work and bookkeeping could be considered in an effort to allow parallelization.

Bibliography

[1] Alessandro Trovarelli. *Catalysis By Ceria and Related Materials*. Catalytic Science Series. Imperial College Press, 2002.

[2] M Battelle and J R Hague. *Refractory ceramics for aerospace : a materials selection handbook / [by the] Battelle Memorial Institute ; Compiled and edited by J. R. Hague... [et al.].* American Ceramic Society, Columbus, Ohio, 1964.

[3] R. J. M. Konings, K. Bakker, J. G. Boshoven, R. Conrad, and H. Hein. The influence of neutron irradiation on the microstructure of Al2O3, MgAl2O4, Y3Al5O12 and CeO2. *Journal of Nuclear Materials*, 254(2-3):135–142, 1998.

[4] T. Sonoda, M. Kinoshita, Y. Chimi, N. Ishikawa, M. Sataka, and A. Iwase. Electronic excitation effects in CeO2 under irradiations with high-energy ions of typical fission products. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 250(1-2):254–258, 2006.

[5] K. Yasunaga, K. Yasuda, S. Matsumura, and T. Sonoda. Nucleation and growth of defect clusters in CeO2 irradiated with electrons. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 250(1-2):114–118, 2006.

[6] T. Sonoda, M. Kinoshita, N. Ishikawa, M. Sataka, Y. Chimi, N. Okubo, A. Iwase, and K. Yasunaga. Clarification of the properties and accumulation effects of ion tracks in CeO2. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 266(12-13):2882–2886, 2008.

[7] Licia Minervini, Matthew O. Zacate, and Robin W. Grimes. Defect cluster formation in M2O3-doped CeO2. *Solid State Ionics*, 116(3-4):339–349, 1999.

[8] R. N. Blumenthal, F. S. Brugner, and J. E. Garnier. The electrical conductivity of cao-doped nonstoichiometric cerium dioxide from 700[degree] to 1500[degree]c. *Journal of The Electrochemical Society*, 120(9):1230–1237, 1973.

[9] C. R. A. Catlow. Fission gas diffusion in uranium dioxide. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 364(1719):473–497, 1978.

[10] R. A. Jackson, A. D. Murray, J. H. Harding, and C. R. A. Catlow. The calculation of defect parameters in uo2. *Philosophical Magazine A*, 53(1):27–50, 1986.

[11] P. Sindzingre and M. J. Gillan. A molecular dynamics study of solid and liquid uo 2. *Journal of Physics C: Solid State Physics*, 21(22):4017, 1988.

[12] R. W. Grimes and C. R. A. Catlow. The stability of fission products in uranium dioxide. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, 335(1639):609–634, 1991.

[13] T. Karakasidis and P. J. D. Lindan. A comment on a rigid-ion potential for uo 2. *Journal of Physics: Condensed Matter*, 6(15):2965, 1994.

[14] Kazuhiro Yamada, Ken Kurosaki, Msayoshi Uno, and Shinsuke Yamanaka. Evaluation of thermal properties of uranium dioxide by molecular dynamics. *Journal of Alloys and Compounds*, 307(1-2):10–16, 2000.

[15] C. B. Basak, A. K. Sengupta, and H. S. Kamath. Classical molecular dynamics simulation of uo2 to predict thermophysical properties. *Journal of Alloys and Compounds*, 360(1-2):210–216, 2003.

[16] L. Van Brutzel, J.M. Delaye, D. Ghaleb, and M. Rarivomanantsoa. Molecular dynamics studies of displacement cascades in the uranium dioxide matrix. *Philosophical Magazine*, 83(36):4083–4101, 2003.

[17] N.D. Morelon, D. Ghaleb, J.M. Delaye, and L. Van Brutzel. A new empirical potential for simulating the formation of defects and their mobility in uranium dioxide. *Philosophical Magazine*, 83(13):1533–1555, 2003.

[18] C. Meis and A. Chartier. Calculation of the threshold displacement energies in uo2 using ionic potentials. *Journal of Nuclear Materials*, 341(1):25–30, 2005.

[19] K. Govers, S. Lemehov, M. Hou, and M. Verwerft. Comparison of interatomic potentials for uo2. part i: Static calculations. *Journal of Nuclear Materials*, 366(1-2):161–177, 2007.

[20] K. Govers, S. Lemehov, M. Hou, and M. Verwerft. Comparison of interatomic potentials for uo2: Part ii: Molecular dynamics simulations. *Journal of Nuclear Materials*, 376(1):66–77, 2008.

[21] B. G. Dick and A. W. Overhauser. Theory of the dielectric constants of alkali halide crystals. *Physical Review*, 112(Copyright (C) 2010 The American Physical Society):90, 1958. PR.

[22] A. Gotte, D. Spngberg, K. Hermansson, and M. Baudin. Molecular dynamics study of oxygen self-diffusion in reduced ceo2. *Solid State Ionics*, 178(25-26):1421–1427, 2007.

[23] Thi X. T. Sayle, Stephen C. Parker, and C. R. A. Catlow. Surface oxygen vacancy formation on ceo2 and its role in the oxidation of carbon monoxide. *Journal of the Chemical Society, Chemical Communications*, pages 977 – 978, 1992.

[24] Da Yu Wang, D. S. Park, J. Griffith, and A. S. Nowick. Oxygen-ion conductivity and defect interactions in yttria-doped ceria. *Solid State Ionics*, 2(2):95–105, 1981.

[25] R. Gerhardt-Anderson and A. S. Nowick. Ionic conductivity of ceo2 with trivalent dopants of different ionic radii. *Solid State Ionics*, 5:547–550, 1981.

[26] J. A. Kilner and R. J. Brook. A study of oxygen ion conductivity in doped non-stoichiometric oxides. *Solid State Ionics*, 6(3):237–252, 1982.

[27] J. A. Kilner and C. D. Waters. The effects of dopant cation-oxygen vacancy complexes on the anion transport properties of non-stoichiometric fluorite oxides. *Solid State Ionics*, 6(3):253–259, 1982.

[28] V. Butler, C. R. A. Catlow, B. E. F. Fender, and J. H. Harding. Dopant ion radius and ionic conductivity in cerium dioxide. *Solid State Ionics*, 8(2):109–113, 1983.

[29] J. A. Kilner. Fast anion transport in solids. *Solid State Ionics*, 8(3):201–207, 1983.

[30] A. D. Murray, G. E. Murch, and C. R. A. Catlow. A new hybrid scheme of computer simulation based on hades and monte carlo: Application to ionic conductivity in y3+ doped ceo2. *Solid State Ionics*, 18-19(Part 1):196–202, 1986.

[31] Rojana Pornprasertsuk, Panchapakesan Ramanarayanan, Charles B. Musgrave, and Fritz B. Prinz. Predicting ionic conductivity of solid oxide fuel cell electrolyte from first principles. *Journal of Applied Physics*, 98(10):103513–8, 2005.

[32] Julian D. Gale and Andrew L. Rohl. The general utility lattice program (gulp). *Molecular Simulation*, 29(5):291 – 341, 2003.

[33] N. F. Mott and M.J. Littleton. Conduction in polar crystals. i. electrolytic conduction in solid salts. *Transactions of the Faraday Society*, 34:485 – 499, 1938.

[34] Chaitanya S. Deo, Maria A. Okuniewski, Srinivasan G. Srivilliputhur, Stuart A. Maloy, Michael I. Baskes, Michael R. James, and James F. Stubbins. Helium bubble nucleation in bcc iron studied by kinetic monte carlo simulations. *Journal of Nuclear Materials*, 361(2-3):141–148, 2007.

[35] P. Contamin, J. J. Bacmann, and J. F. Marin. Autodiffusion de l'oxygene dans le dioxyde d'uranium surstoechiometrique. *Journal of Nuclear Materials*, 42(1):54–64, 1972.

[36] G. E. Murch, D. H. Bradhurst, and H. J. De Bruin. Oxygen self-diffusion in non-stoichiometric uranium dioxide. *Philosophical Magazine*, 32(6):1141 – 1150, 1975.

[37] J. Faber, C. Geoffroy, A. Roux, A. Sylvestre, and P. Abelard. A systematic investigation of the dc electrical conductivity of rare-earth doped ceria. *Applied Physics a-Materials Science & Processing*, 49(3):225–232, 1989. ISI Document Delivery No.: AM139 Times Cited: 54 Cited Reference Count: 15 Springer verlag New york.