ATOMISTIC STUDIES OF POINT DEFECT MIGRATION RATES IN THE IRON-CHROMIUM SYSTEM

Jeffery Hetherly, B.S.

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

August 2010

APPROVED:

Srinivasan Srivilliputhur, Major Professor
Duncan Weathers, Committee Member
Bibhudutta Rout, Committee Member
Narendra Dahotre, Chair, Department of Materials Science and Engineering
James D. Meernik, Acting Dean of the Robert B. Toulouse School of Graduate Studies

Hetherly, Jeffery. Atomistic Studies of Point Defect Migration Rates in the Iron-Chromium System. Master of Science (Materials Science and Engineering), August 2010, 68 pp., 6 tables, 25 figures, references, 34 titles.

Generation and migration of helium and other point defects under irradiation causes ferritic steels based on the Fe-Cr system to age and fail. This is motivation to study point defect migration and the He equation of state using atomistic simulations due to the steels' use in future reactors. A new potential for the Fe-Cr-He system developed by collaborators at the Lawrence Livermore National Laboratory was validated using published experimental data. The results for the He equation of state agree well with experimental data. The activation energies for the migration of He- and Fe-interstitials in varying compositions of Fe-Cr lattices agree well with prior work. This research did not find a strong correlation between lattice ordering and interstitial migration energy.

ACKNOWLEDGEMENTS

I would like to thank Dr. Srinivasan Srivilliputhur and Dr. Duncan Weathers. They have both been a tremendously positive influence in my academic career. Dr. Caro of Los Alamos National Laboratory and Dr. Zapeda-Ruiz of Lawrence Livermore National Laboratory have both been exceptionally helpful and insightful in this research. I would also like to give thanks to the U.S. Department of Energy through which my A.F.C.I. fellowship was sponsored. Both Cathy Dixon and Donna Knight of the University Research Alliance were extremely helpful throughout my graduate experience.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

INTRODUCTION

The growing concern over clean, reliable sources of energy has led our society to reconsider nuclear energy as part of a more diverse energy portfolio. Wider public acceptance of nuclear energy as a viable solution can be accomplished by responsible handling of waste combined with an increase in the reactors' efficiency as defined in [1]. Current plans to make fission reactors more efficient will require that future reactors operate at higher temperatures (up to 1200°C) in order to burn more fuel than current reactors (Fig. 1) [1]. This next generation of more efficient, safer, and more cost effective reactors is called Generation IV reactors (or Gen IV). The current steels (such as 304 stainless steel) used in reactors are incapable of handling the increased irradiation over long periods of time, and at elevated temperatures. They exhibit considerable swelling under the higher temperatures and radiation dose. Therefore new steels are required to build next generation nuclear reactors. Due to their larger corrosion and swelling resistance, Fe-Cr (iron-chromium) based ferritic steels are potential structural materials for the next generation of nuclear reactors.

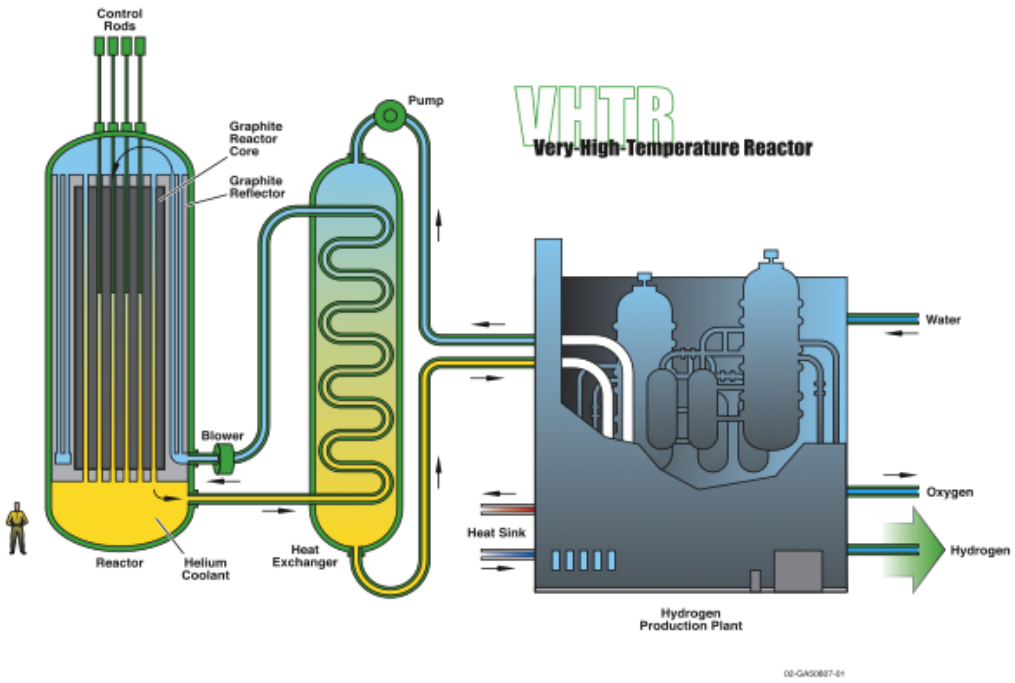


Fig. 1. Diagram of the very-high-temperature reactor [1].

Computer simulations of potential materials for nuclear reactors are more desirable as the cost and production time are typically lower than physical experiments. Radiation damage in particular is well suited for computer modeling since physical experiments are traditionally done after radiation is applied to the sample, and this kind of testing omits detailed examination of processes that might be more apparent if the experiment is conducted while the material is irradiated. While computer simulations may not be nearly as accurate as a physical experiment, they do allow for a detailed inspection while radiation is incident on the material and are typically cost effective in terms of materials and preparation.

However, computational models are not completely accurate in most situations. For instance, a complete, accurate description of Fe-Cr ferritic steels is not currently reliable on the atomistic, mesoscopic, and continuum scale. Information on diffusion, grain boundaries,

and dislocations are obtainable for the Fe-Cr system on the atomistic scale. This is why quantum and atomistic approaches to gathering information on this Fe-Cr system are beneficial at this time. The Fe-Cr system is used in this research as a surrogate system for the more complex Fe-Cr based steels. This is done because the actual steels have about 10 alloying elements, and modeling such a complex system on the atomistic scale is not practical at this time. Fe and Cr are a large portion of the atomic composition of these steels, and gaining information about this ideal, surrogate system can give insight into how the more complex alloy steels behave. Once enough data is collected from these scales and from physical experiments—mesoscale and eventually continuum scale models of real ferritic steels can be developed. Such faithful models of the Fe-Cr system will be useful in describing the material behavior for Fe-Cr based steels under next generation reactor conditions.

As of right now, point defect migration rates are not documented for different compositions of Fe and Cr, nor has a study been preformed on the effects of lattice ordering on the migrations rates of point defects. To this end of developing a more complete model of Fe-Cr systems, this research focuses on key aspects of point defect formation and migration energies relevant to irradiation-induced damage at the atomistic scale. The goal for this research is to gather the migration rates and formation energies of He-interstitials and Fe-interstitials using molecular dynamics to later be used in larger scale (both spatially and temporally) simulations.

BACKGROUND

This section provides introductory information on material and terminology frequently referenced in this research. Topics include: Solid solutions, alloys and the Fe-Cr system, lattice ordering, point defects, formation energy, migration barrier energy and diffusion, voids and He-bubbles, and embrittlement.

Solid Solutions, Alloys and the Fe-Cr System

A solid solution is the solid state equivalent of liquid solution. The solute atoms are dissolved to a solvent to create a solution. This dissolution may take place in a substitutional or interstitial manner. The solute can "substitute" for the solvent atoms on lattice sites if the solute atoms are similar to the solvent (Hume-Rothery rules discussed below). If the solute atoms are dissimilar (i.e. atomic radii differ by more than 15%, electronegativities are different, coordination numbers are different) from the solvent, then an interstitial incorporation of the solute atoms can occur where the solute atoms sit in between lattice sites, such as carbon atoms in steel. In either case, the solute is evenly dissolved within the solvent lattice. If solubility between the solute and solvent is not present, then the solute atoms may have a tendency to cluster and become trapped at defects within the solvent (such as at grain boundaries and dislocations). If solubility is very low, this clustering behavior is thermodynamically predominating.

Alloys contain several components (Table I for example). These additional elements (or alloying elements) help to tailor the properties and performance of the material (steel in this case) to excel under certain conditions. For instance, steels used as structural materials in nuclear reactors must tolerate much higher radiation dose and higher

temperatures than structural steels used in skyscrapers. Even though these two steels are used as a structural material, both the type and relative amounts of elements present in them are different.

Table I. Composition of 304 stainless steel [2].

Composition of SURV 304 Stainless Steel (given in weight percent)										
Material	C	Cr	Cu	Fe	Mn	Mo	Ni	P	S	Si
wt%	.08	18.38	.18	Bal.	.89	.21	10.0	.018	.020	.68

Elements in an alloy can dissolve into the host lattice matrix (α-Fe lattice in the case of the Fe-Cr system) in a variety of ways. One common way that alloys elements mix is when the alloying element sits at lattice sites of the host lattice. This is called a substitutional alloy since the alloying element is substituting for some of the original lattice atoms. The rules for determining the whether or not two element will form a substitutional alloy are called the Hume-Rothery rules. These rules are:

1. The radii of the host lattice atom and alloying element atom cannot differ by more than 15%
2. The crystal structures of both elements must be the same
3. The number of bonds of each element must be similar
4. The electronegativities of each element must be similar

The Fe-Cr system forms a substitutional alloy. All of the Hume-Rothery rules are satisfied by Fe and Cr. The atomic radii of Fe and Cr are 1.27Å and 1.28Å, respectively. Both have the body-centered cubic (bcc) crystal structure, and due to their close proximity to one another on the periodic table, they share similar valencies and electronegativities. The phase diagram for the Fe-Cr system is given in Fig. 2. The α phase in the Fe-Cr system has a bcc crystal structure like α-Fe. γ is an fcc phase that is softer and more ductile than the bcc

phase. Alternating layers of Cr-rich layers and Fe-rich layers distinguish the σ phase [3]. This phase diagram is in units of mass (or weight) percent, and since Fe and Cr are so close to one another in mass, mass percent and atomic percent may be interchanged with minimal error. The atomic percents used in this research range from 0at%Cr (pure α-Fe) to ~15at%Cr and the temperature range is 300K-1000K. In these ranges, only the bcc α phase is present. Within this mass-percent range, the γ phase starts to appear only after ~850°C till ~1400°C with a segregation of the α and γ phases between 10 and 14 mass percent. The melting temperature in this range is ~1500°C. The σ phase only sets in after 20 mass percent in the temperatures used in this research.

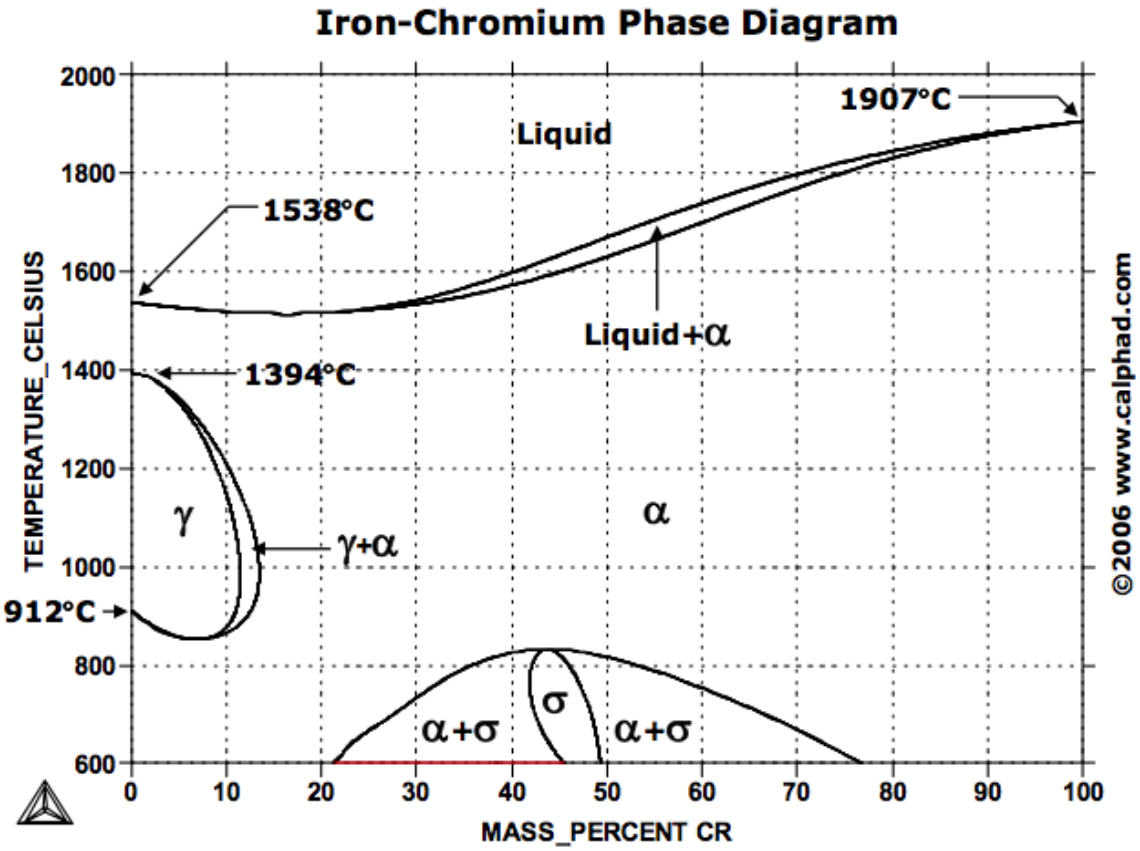


Fig. 2. Phase diagram for the Fe-Cr system [4].

Lattice Ordering

Atoms in substitutional alloys, such as Fe-Cr, order randomly if the probability of finding either element at a lattice position equals their mole fractions in the alloy [5]. For example, in an 80at%Fe and 20at%Cr randomly ordered lattice then any lattice position should reflect this 80-20 split in the surrounding environment. For such a lattice, the number of bonds between Fe and Cr (mixed bonds) can be shown to be

$$P_{FeCr} = N_a z X_{Fe} X_{Cr}$$

Eqn. 1

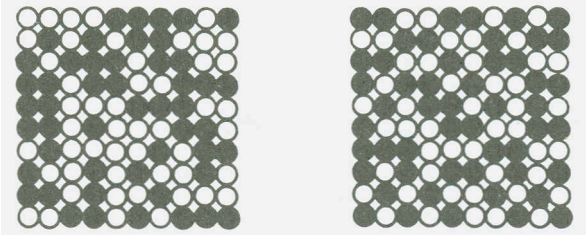
where N_a is Avogadro's number, z is the number of bonds per atom, and X_{Fe} and X_{Cr} are the mole fractions of Fe and Cr. So, in this case, $z = 8$, $X_{Fe} \approx .8$, and $X_{Cr} \approx .2$.

Short range ordering (SRO) occurs when the number of mixed bonds is greater than that of the randomly ordered Fe-Cr systems (more general conditions must be met for an arbitrary system)[5]. The degree of ordering can be quantified in the SRO parameter

$$s = \frac{P_{FeCr} - P_{FeCr}(random)}{P_{FeCr}(\max) - P_{FeCr}(\text{random})}$$

Eqn. 2

This is the ratio of the difference in the number of actual mixed bonds and number of mixed bonds in a randomly ordered system over the difference in the maximum number of possible mixed bonds and actual number of mixed bonds in a randomly ordered system. Fig. 3 is given as an example. One of the goals of this research is to investigate whether the ordering of the Fe-Cr lattice influences point defect migration



Both figures have X_{Fe} = X_{Cr} = .5 and $P_{FeCr}(max)$ = 200. For the random ordering S = 0 and for the SRO S = .32

Fig. 3. Example of random ordering (left) and SRO (right) [5].

Point Defects

Point defects are lattice imperfections that only pertain to a single lattice site. These are classified as vacancies, interstitials, or substitutionals. When a lattice atom is removed from its position, the empty lattice position left behind is called a vacancy. Interstitials are atoms that move between lattice atoms and are not located at lattice sites. Interstitials may be categorized further as self-interstitials when the lattice has only one component and the interstitial is the same type of atom. Substitutional defects are former vacancies that have been occupied by non-lattice atoms. For instance, He is attracted to vacancies in the Fe-Cr system and so tries to occupy vacancies whenever possible.

The formation energy associated with a point defect is the work done create that defect. It represents the energy of the bonds that must be broken with in the lattice to accommodate the defect. Formation energies are found by computing the total energy of the system before and after the defect is present.

Migration barrier energy is the energy required by a point defect to overcome the local inter-atomic bonds and move to another lattice site in the case of vacancies and substitutionals or to another location between lattice sites in case of interstitials. Migration barrier energy is obtained by fitting the motion of the point defect to

the Arrhenius equation (next section). The activation energy in the Arrhenius equation is the migration barrier energy for point defects.

Diffusion of Point Defects

The Arrhenius equation is often used to quantify migration rate when studying the movement of point defects. This equation (Eqn. 3) relates the rate constant D (in units of [length]^2/[time]) of a process to the temperature T and activation energy Q.

$$D = D_0 e^{-\frac{Q}{k_B T}}$$

Eqn. 3

D_0 is the pre-exponential factor and k_B the Boltzmann constant. The objective of this research is to collect atomistic data on the temperature dependence of migration for He-interstitials and Fe-interstitials so that rate constants for these defects can be used in kinetic Monte Carlo simulations. For this research to be relevant to the issue of radiation damage in reactors, finding the dependence of the rate constant on Cr concentration is also important as different steels have varying amounts of Cr.

To obtain the value of the rate constant from raw displacement data from simulations, a few steps must be taken. First, the mean-squared displacement (msd) is calculated for each temperature. Then, a linear slope in fitted to the msd versus time plot. Einstein's equation (Eqn. 4) for diffusion is then used to get a value for the rate constant at that temperature.

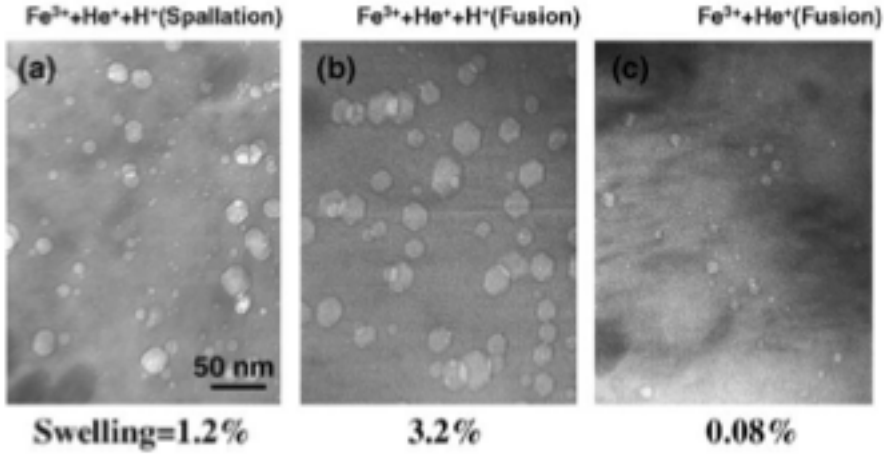
$$D = \frac{1}{6}\frac{d\langle s^2 \rangle}{dt}$$

$$\text{where } \frac{d\langle s^2 \rangle}{dt} \text{ is the slope of the mean - squared displacement versus time}$$

Eqn. 4

Voids and He-Bubbles

Most real materials are made of several, differently oriented crystals rather than one large single crystal. The boundaries between these individual crystals within a material are called grain boundaries. These grain boundaries (GBs) can absorb or generate point defects. Void swelling (Fig. 6) starts with an energetic radiation knocking an atom from its lattice position and creating both a vacancy and interstitial atom pair. The interstitial atoms travel much faster than the vacancy through the lattice, and are often trapped at grain boundaries (GBs). Due to the drastic difference in migration rates of interstitials and vacancies, these defect pairs have low statistical probability for recombination. Thus, the slower vacancies are typically left within the bulk while the faster interstitials are trapped at the GB. These individual vacancies can eventually coalesce to form much larger voids. Irradiation-induced void swelling occurs when enough of these void-forming processes transpire within the material.



Samples of steel irradiated at 470°C to 50 dpa by different means (see reference for details)

Fig. 4. Example of void formation in F82H steel [6].

He is an inert, insoluble substance in the Fe-Cr lattice. Radiation from nuclear reactors introduces He either through α-radiation or through transmutation of the

elements present in the steel by a bombardment of neutrons. Hence, from the aforementioned causes of He in metals, the presence of He in Fe-Cr is typically accompanied with some form of lattice damage. Temperature, He production rate, displacement rate, and accumulated He concentration are the factors critical to He bubble production [7]. A He-substitutional forms when a vacancy traps a former He-interstitial atom. At sufficient temperatures and vacancy concentration for He this He-vacancy pairing is the preferred configuration [7]. If conditions are suitable many of these He-vacancy pairs—some of which have more than one He per vacancy—coalesce to form larger bubbles. These bubbles (Figs. 11&12) are mobile and tend to congregate at grain boundaries and dislocations (Figs. 5&6)[8][9]. This tendency to congregate causes large cavities to form and increases the chance for rupture [10].

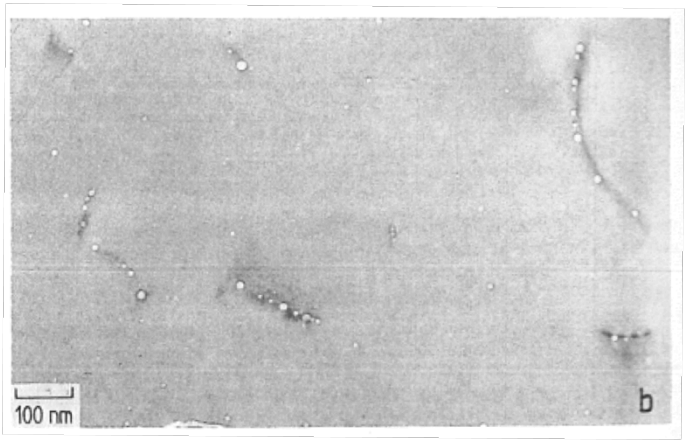


Fig. 5. TEM micrograph of He bubbles at dislocations [8].

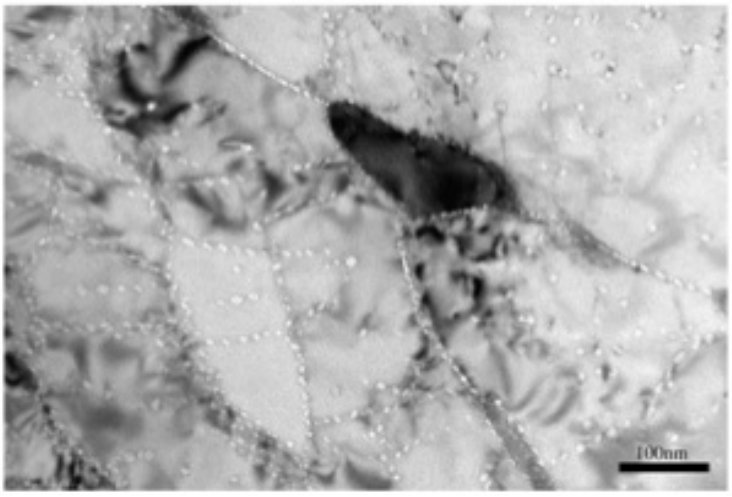


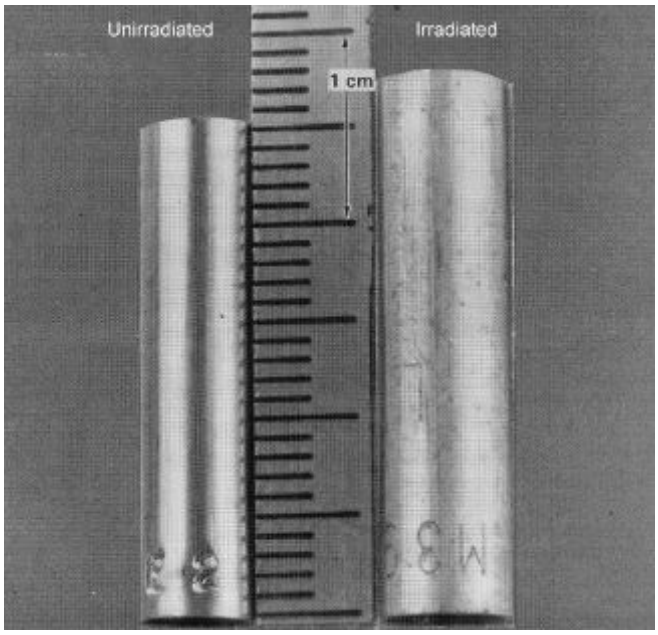
Fig. 6. TEM micrograph of He bubbles at grain boundaries [11].

Embrittlement

A GB can trap larger defect structures, such as voids and He-bubbles. When a GB traps these defects, it acts to embrittle the material and lower their ductility (Fig. 9). This detrimental effect occurs because voids and He-bubbles impede dislocation movement. When a normal metal undergoes plastic deformation, dislocations traverse through the material to accommodate the strain. However, in embrittled metals, the voids and He-bubbles act to impede the dislocations from moving. This increases the metal's yield strength at the cost of its ductility because the trapped dislocations are unable to relieve the strain and the metal eventually fractures.

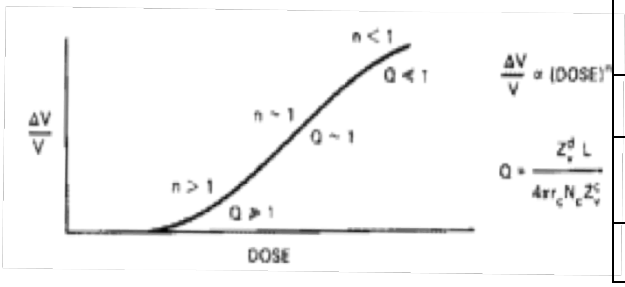
WHY MIGRATION OF Fe-INTERSTITIALS AND He-INTERSTITIALS ARE RELEVANT TO HIGH ENERGY NUCLEAR REACTORS

Radiation damage from fusion or from Gen IV fission reactors in ferritic/martensitic steels creates both swelling and He embrittlement within the steel (see Figs. 4-6) [12]. These two forms of damage degrade the tensile, creep, and fatigue properties of the steel [13]. These undesirable effects can ultimately over time degrade the structural integrity of the reactor as much of the supporting structure of the reactor is made of Fe-Cr steel. The migration rates of self-interstitials, He-interstitials, vacancies, and He-vacancy clusters are among the most important factors in both swelling and embrittlement [14]. Studying these factors will help to further quantify the effects of radiation in these alloys.



Photograph of 20% cold-worked 316 stainless steel rods before (left) and after (right) irradiation at 533°C to a fluence of 1.5×10^{23} neutrons/m^2 in the EBR-11 reactor

Fig. 7. Example of irradiated steel [15].



Z	These quantities measure how well the system captures dislocations and cavities (voids)
L	Dissociation density
r	Average cavity radius (cavity in this sense is a collection of voids)
N	Cavity density

Fig. 8. Dependence of swelling on irradiation dose for alloys [4].

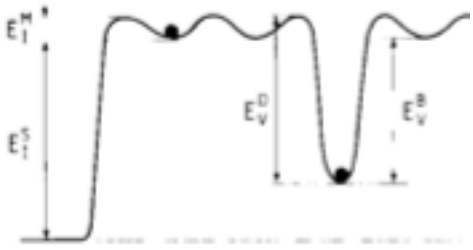
Fe-Interstitial Migration in the Fe-Cr System

Fe-interstitial migration in the Fe and Fe-Cr system takes place when an atom next to the interstitial is displaced just enough so that the center of mass of the lattice atom and interstitial atom occupy the lattice position. This arrangement is called an interstitial dumbbell. These dumbbells move along close-packed directions and can rotate into different close-packed directions when thermally activated.

In the Fe-Cr system, several studies have shown that a mixed-species dumbbell is preferred to a pure Fe or Cr dumbbell [16]. This mixed-species dumbbell preference is supported by the potential used in this research.

He-Interstitial Migration in the Fe-Cr System

He only occurs in metals when forced through "tritium decay, α-injection or by (n,α) reactions of neutrons with matrix nuclei during neutron irradiation"[7] due to the low solubility of He in metals. This low solubility results in more He atoms found in vacancy sites as opposed to interstitial sites (Fig. 9). In formal terms, the formation energy associated with He-interstitials is always higher than the energy for He-substitutionals [13]. He interstitials play a large role in embrittlement and He bubble formation despite being energetically less favorable than He- substitutionals. Fig. 10 shows different types of He interstitial/substitutional movement.



E_I^S	Energy of solution into interstitial sites
E_I^M	Migration energy of interstitials
E_V^D	Dissociation energy from vacancy
E_V^B	Binding energy between He atom and vacancy

Graphical representation of the difference in formation and migration energies of He interstitials and He-substitutionals

Fig. 9. Graph of He migration energy [13].

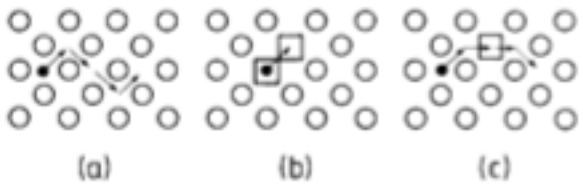
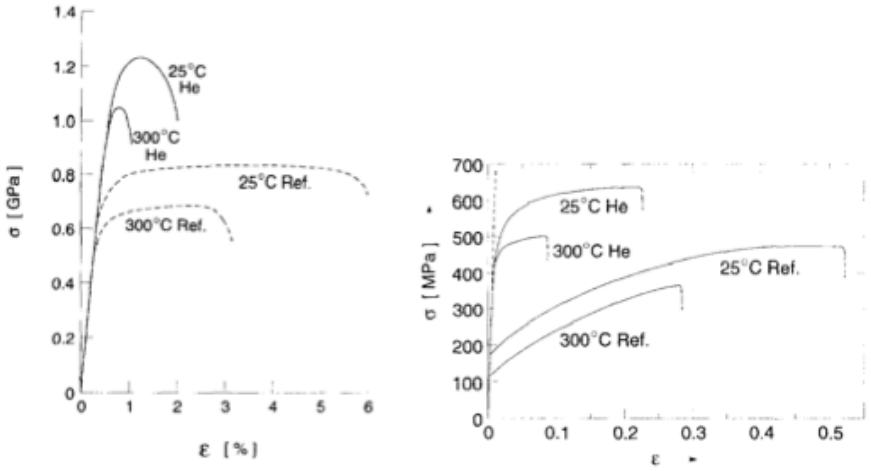


Diagram of (a) interstitial He migration, (b) He vacancy migration, and (c) mixed vacancy/interstitial migration

Fig. 10. Types of interstitial movement [13]

He-Vacancy Migration, Void Formation, and He Bubbles in the Fe-Cr System

The low solubility of He in metals results in a strong tendency for He to precipitate or cluster [13]. These clusters of He are typically called He bubbles within the metal and are a major contributor to degradation of mechanical properties (Figs. 11&12). He bubble characteristics are discussed in this section.



These graphs illustrate the degradation of ductility after He implantation (.55at%He for the left graph and .45%He for right)

Fig. 11. Stress-strain graphs for Manet steel and 316 L steel [17].

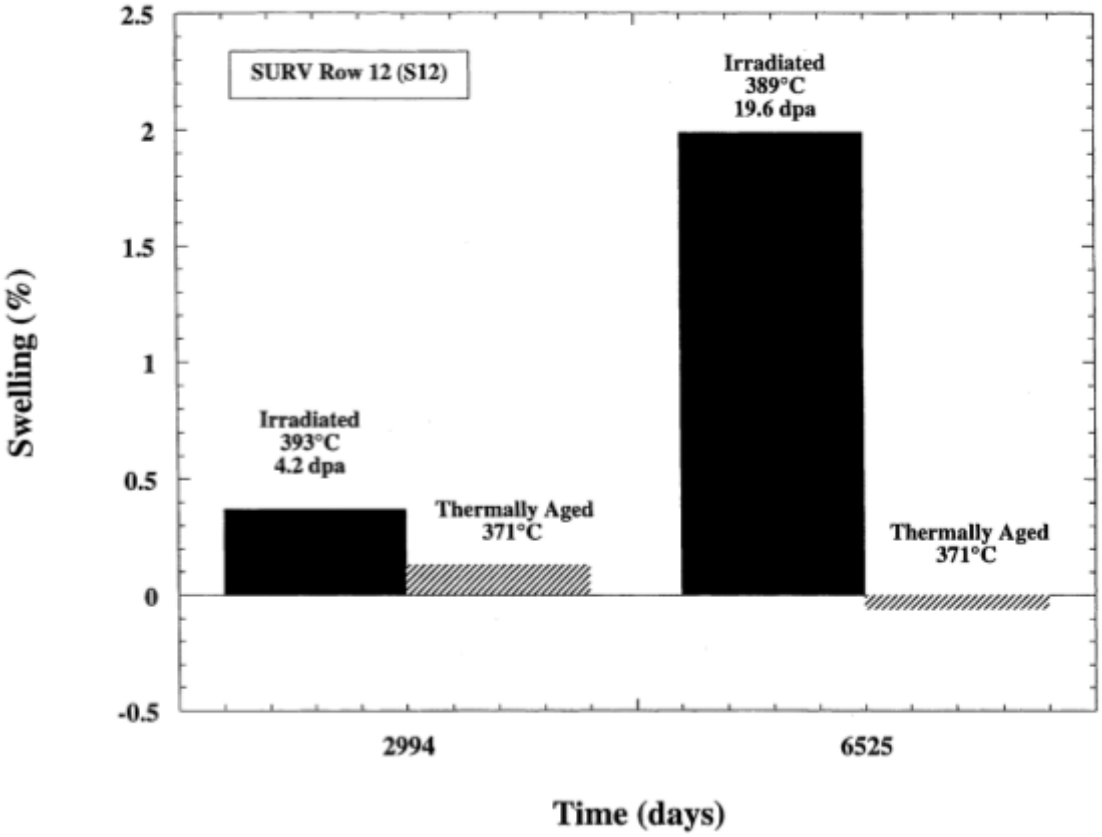


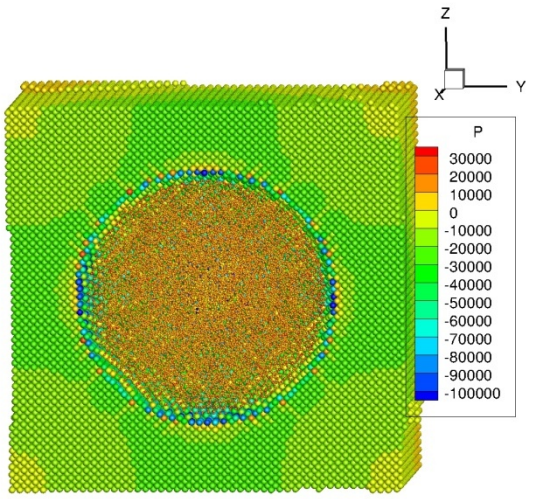
Fig. 12. Swelling due to both thermal aging and irradiation in 304 stainless steel [2].

The equation for the equilibrium pressure (Fig. 14) inside these bubbles is

$$p = \frac{2\gamma}{r}$$

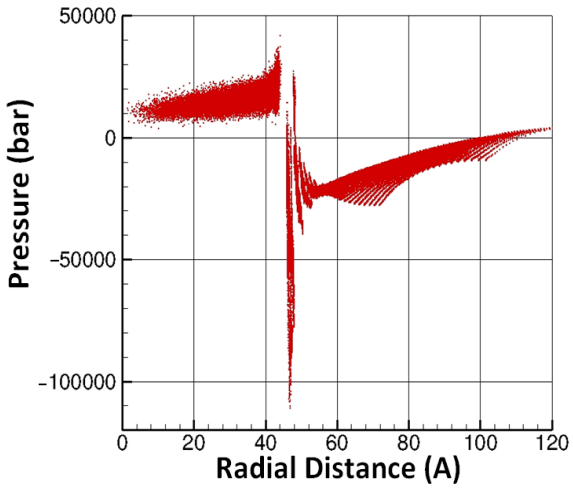
Eqn. 5

where *r* is the radius of the bubble and γ is the surface tension of the bubble [8]. Fig. 15 gives a graph of concentrations of He atoms and He bubbles for typical metals.



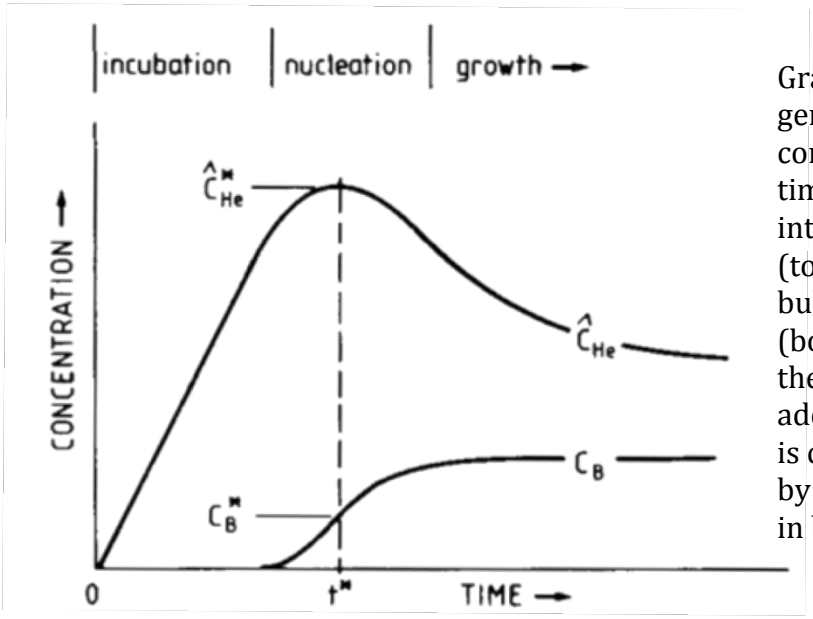
An 8nm He (small spheres) bubble embedded in an Fe (larger spheres) lattice at 0K and density ρ = 1 (Helium)/(Iron Vacancy). The atoms are colored according to their pressure in units of bars. The pressure at the corners of the box is due to periodic boundary conditions.

Fig. 13. Simulation of He bubble in α-Fe [18].



Pressure as a function radial distance from the center of the He bubble in Figure 13. The pressure is more or less constant inside the bubble while outside it exhibits negative, inverse square decay.

Fig. 14. He bubble pressure [18].



Graph showing the general relation of concentration versus time between interstitial He density (top curve) and He bubble density (bottom curve). t^* is the time where any additional atomic He is counterbalanced by absorption of He in bubbles.

Fig. 15. He interstitial and bubble concentration versus time [13].

It is worth noting two characteristics of these He bubbles in Fe-Cr at high temperatures [7]. At low displacement dose (<1 dpa) the bubble density has a very weak dependence on He concentration and time. This indicates at low displacement doses bubbles nucleate in early stages of irradiation, and each bubble slowly increases in diameter instead of newer, smaller bubbles forming. Additionally, He production rate has an inverse and direct relation between bubble size and bubble density, respectively.

ATOMISTIC SIMULATIONS

Computer simulations that focus on the classical, Newtonian motion of individual atoms as a whole are called molecular dynamic (MD) simulations. The molecular dynamics approach does not explicitly work with the wave functions associated with atoms (as in *ab-initio* techniques) or large collections of atoms treated as one object (as in rate theory and finite element approaches). MD approaches the task of simulating several atoms by

deriving a force from a potential function and integrating that force over time. MD is a classical approach in that the atoms are point particles and have no internal structure.

The Embedded Atom Method (EAM)

For metallic systems with 10^2-10^6 atoms, MD, in conjunction with the embedded atom method (EAM), is well suited to accurately reproduce bulk properties of the crystal, and Daw and Baskes developed the EAM method in their seminal paper on the subject [19]. The central idea of this theory is to think of each atom in the lattice as a defect embedded within an otherwise unperturbed lattice. This way of reasoning is physically plausible "[b]ecause the energy of an impurity is a functional of the electron density of the unperturbed host, the cohesive energy of a solid can be calculated from the embedding energy [19]." The embedding energy of an atom in a solid is the energy of that atom in a uniform electron gas relative to that same atom separated from the electron gas, and simply stated is the energy of the atom inside the solid versus outside the solid.

The basic EAM model gives the energy of each atom *i* as

$$E_i = F_\alpha\left(\sum_{j\neq i}\rho_\alpha(r_{ij})\right) + \frac{1}{2}\sum_{j\neq i}\phi_{\alpha\beta}(r_{ij})$$

Eqn. 6

Eqn. 6 has two terms. The first term is the embedding energy of element α of atom *i* due to the total electron density from atoms within a certain cutoff distance. This embedding energy is calculated from individual electron densities ρ_α evaluated at atom *i*'s location. The second term is a pair-wise potential between elements α and β at a distance r_{ij} away from each other.

Additional modifications to the basic EAM model introduce concentration-dependant potentials (CD-EAM)[20]. The technical difference in the concentration-dependent formulation of Eqn. 6 is when dealing with the second term. In the original EAM, the cross term for the inter-atomic potential for Fe-Cr is implemented as

$$\phi_{FeCr}(r_{ij}) = \tfrac{1}{2}(V_{FeFe}(r_{ij}) + V_{CrCr}(r_{ij}))$$

Eqn. 7

The CD-EAM modifies this term to account for local electron densities from surrounding atoms based on their element type [21].

$$V_{FeCr}(r_{ij}) = h(x_{ij})\phi_{FeCr}(r_{ij})$$
$$x_{ij} = \tfrac{1}{2}(x_i - x_j) = \frac{1}{2}\left(\frac{\eta_i^{Cr}}{\eta_i} + \frac{\eta_j^{Cr}}{\eta_j}\right)$$

Eqn. 8

Obviously, if $h = 1$ then the normal EAM formulation is recovered. x_{ij} is the average partial density of Cr at atoms *i* and *j*. Because this potential takes the lattice environment into account when computing forces it more accurately describes multi-component alloys. Further modifications can be made to optimize this potential's usage for large-scale simulations and for use in Monte-Carlo simulations [21].

LAMMPS

LAMMPS stands for Large-scale Atomic/Molecular Massively Parallel Simulator. LAMMPS open-source software package is developed and maintained at Sandia National Laboratory. As its acronym implies, LAMMPS allows users to easily run MD parallel simulations. It is scalable, able to run on one processor as easily as thousands of processors, feature-rich, and able to output data in several standard format as well as custom output.

Parallelization

LAMMPS allows for parallelization through a standard interface known as MPI (Message Passing Interface). In LAMMPS the majority of the MPI standard is implemented is behind the scenes. However, if modifications to LAMMPS are necessary for a particular project a rudimentary understanding of MPI is required.

Simulations such as MD are inherently parallel in that they can be spatially decomposed into domains containing only a few atoms. Initially, a distance (called a cutoff distance) is chosen so that only atoms within this distance are considered when computing forces. Most interactions between metal atoms in a crystal are primarily short-range forces, so a cutoff distance is judiciously chosen to help decrease the amount of computation but still retain physical accuracy. Forces are then calculated for each atom within a domain. Then, cross-communication between surrounding domains for the atoms close the edges of the domain gives the resultant forces on each atom due to all the other atoms within a certain cutoff distance. This method of parallelization is computationally efficient if the number of atoms in each domain is optimized so that the tradeoff between single-processor load and cross-communication is balanced.

Extending the Time Scale: Parallel Replica Dynamics

One of the main shortfalls of the classical MD approach is the short time scale realizable within any practical computational period. For example, on 24 processors with 2001 atoms and 8 hours of computational time, only 1ns of simulation time can be achieved using a standard EAM MD approach. Increasing the number of processors makes only a marginal difference, and eventually, the addition of more processors will degrade performance as inter-node communication increases compared to computation time. This

performance degradation sets practical limitations on what type of phenomenon one can expect to observe using this method. If an event were estimated to take more than 1μs to occur, the standard MD approach on the aforementioned system would take an impractical amount of computational time for observation. Several modifications to MD such as hyperdynamics, temperature-accelerated dynamics, and parallel replica dynamics can speed-up the accessed time [22][23][24]. Parallel replica dynamics is one such approach that is currently implemented in LAMMPS.

Parallel replica dynamics (PRD) is a technique that uses statistical mechanics to sample infrequent events. In statistical mechanics when considering diffusion events, running one experiment for a great length of time is equivalent to running several, similar experiments at the same time, and this is the central idea behind PRD. Instead of running a single simulation for several hours or even days, PRD runs several similar simulations at the same time and checks for a desired event in all of the simulations—thus working around the problem by simply increasing the number of processors on a single system. This method also makes sure that two consecutive events are uncorrelated. Optimizing the number of processors on a single simulation and creating several similar optimized systems often drastically reduces the computational time. The modeling of vacancy diffusion on the Cu(100) surface with only 15 processors yields a 14-fold decrease in computational time [23].

Kinetic Monte Carlo

Kinetic Monte Carlo (KMC) is a rate theory approach to modeling phenomena given information on how those phenomena interact and how often they occur. A particular

adaptation of this scheme called the object kinetic Monte Carlo (OKMC) is used to model point defects migration [25].

> It follows the evolution of a set of objects in time, given the type of events those objects can perform and the probability for each event to occur. In the case of radiation the objects of interest are those defects produced during the irradiation, that is vacancies, self-interstitials, impurities and their clusters. The events these objects can perform are diffusion events, dissolution from a cluster, interaction between different defects or defects with other objects such as grain boundaries or dislocations. The probabilities of these events are given by the migration energies and binding energies of the defects. [26]

This neatly summarizes the use of OKMC in the study of irradiation-induced defects.

OKMC allows for yet another extension of the time scale when modeling radiation effect in metals. Typically, the time and length scale encompassed by the OKMC approach is referred to as the mesoscopic scale. It lies somewhere between the atomistic scale and the continuum scale and acts to bridge the gap between the two regimes. The ultimate goal of this research is to eventually help build a complete, quantitative picture of radiation damage in nuclear reactors' materials. In keeping with this goal, an OKMC model will eventually be needed. However, it should be noted that OKMC requires a pre-cataloging of all the diffusive events and to which we may only hypothesize those events that exist [24].

SIMULATION TECHNIQUE

Using LAMMPS

LAMMPS is easy to use once it is compiled on a particular machine. To use LAMMPS to run MD simulations only three key pieces are needed. These key pieces are an input script, an atom file, and a potential file. While it is true that LAMMPS comes with several different potentials built in and one can create the atoms in the input script, testing new potentials and/or complicated materials does require these three pieces.

First, input scripts are text files formatted in a way as to give instruction to the LAMMPS program. These files give LAMMPS such information as what units to use, coordinates/velocities of atoms, types of atoms, as well as what to do to the atoms such as heating the atoms, minimizing energy, and so on, and then finally how long the simulation will last given in terms of time steps. The typical structure of a LAMMPS script is as follows (taken from [27]):

1. Initialization
2. Atom Definition
3. Settings
4. Run a Simulation

Initialization tells LAMMPS what units to use, what type of atom potential is used, and how to parse the use of the processors for parallelization. Defining atoms is done in one of three ways. Specific LAMMPS commands are either given from the input script to make and load atoms in to memory, read atom positions and velocities from a specially formatted text file, or read in a restart file. For any atomic arrangement other than the most basic structures, atoms are typically not created using the input script. Reading a specifically formatted text

file containing the information about the atoms' position and velocities allows flexibility in creating defects. Reading restart file is valuable if the script is continuing a simulation that was previously run. LAMMPS restart files are binary files that contain all the information about a previous simulation. Settings allow the user to adjust several parameters in simulation such as temperature, pressure, labeling, and tracking certain groups of atoms, and to specify the format of the output, or constrain the system in many other ways. This is essentially where the user tells LAMMPS what actions to take. Finally, running the simulation requires *minimize*, *temper*, *prd*, or *run* commands. Some of these commands will be discussed in detail later in the thesis, but for now the *run* command simply tells LAMMPS to start evaluating forces on each of the atoms and integrate over time a specified number of steps. Steps three and four in the overall process may be repeated until a desired result is obtained.

Secondly, atom files are the text files that specify either just the location of the atoms or both the location and velocities of the atoms. The first two lines of the atom file are comment lines that are meant to clarify what the file contains. The next three lines give the dimension of the simulation box or terms of unit given in the input script. Lines six and seven tell LAMMPS how many atoms and types of atoms are included in the simulation. The following lines are the heading followed by the list of atoms. Atoms are specified by their ID, type, and 3-dimensional location. After the atoms are created, velocities may be provided with a heading and list similar to the positions.

Third, a potential file is necessary to be read if LAMMPS does not contain the potential already built in. There are more than 50 styles of pair potentials that can be read in to LAMMPS, however, each of these styles must be formatted correctly. LAMMPS

common design for every potential file is that the potential is tabulated. The reason for this is computational speed. When a potential is in its analytical form it takes several compute cycles to evaluate the resultant force. Yet, if the potential is tabulated for values of the potential the simulation is likely to encounter, all that needs to be done when computing forces is to look up a value in a table and interpolate for the specific value requested. This tabulated approach proves to be computationally far faster as LAMMPS has state-of-the-art interpolating functions.

LAMMPS Commands Relevant to This Research

This section explains in some detail the relevant LAMMPS commands used in this research.

- *units* –tells LAMMPS what units to use: e.g. the keyword *metal* specifies distance in angstroms, time in picoseconds, pressure in atmospheres, and so on
- *atom_style* – determines what type of information is associated with each atom
- *boundary* – specifies what type of boundary to implement on each face of the simulation box; this can have values of periodic (if an atom goes outside the boundary it shows up on the opposite side; see Fig. 16), fixed (atoms are lost if they go past the boundary), or shrink wrapped (non-periodic, but atoms are not lost if the go outside the boundary)
- *newton* – specifies whether or not to use Newton's 3rd law (two atoms exert equal and opposite forces on one another) when evaluating forces; turning this on gives a slight boost in performance in most circumstances
- *read_data* or *read_restart* – these two commands read the atomic data from a text file or a binary restart file, respectively

- *pair_style* – tells LAMMPS what type of potential is being used; *eam/cd* specifies a concentration dependant EAM potential is to be used
- *pair_coeff* – this simply loads the potential file in to memory and specifies what type of atoms are used in the potential file
- *velocity* – gives the atoms some initial velocity based on a Gaussian distribution of speed as determined by the temperature
- *fix* – this command has a variety of purposes: for instance, this command can tell LAMMPS to raise the temperature from 300K to 600K
- *unfix* – removes the specified fix command
- *thermo* and *thermo_modify* – tells LAMMPS to print thermodynamic information and modifies how or what is printed
- *timestep* – specifies the how much simulation time to integrate over
- *compute* – tells LAMMPS to compute certain quantities at every time step
- *dump* – in addition to a standard log file LAMMPS can create custom output files using this command
- *run* – run a simulation for a specified number of time steps
- *minimize* – adjusts the coordinates of each atom in such a way as to minimize the total energy of the system (also known as quenching)
- *prd* – runs a parallel replica dynamics simulation with parameters specifying how to check for infrequent events, and how to partition the processors are done in the input script and the command line

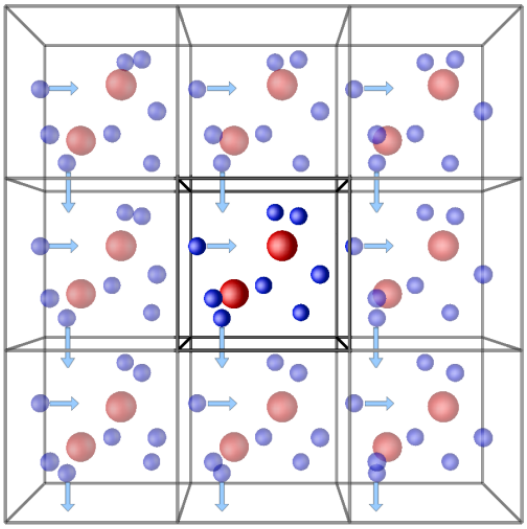


Fig. 16. Representation of periodic boundary conditions [28].

Time Scale

Computational times for the simulations in this research range from 4-8 hours. Simulated times are on the order of .1ns-10ns. In each simulation, the simulated time is long enough to reach valid statistics for the objects of interest (e.g. enough interstitial jumps took place). The simulated time for the He interstitial runs were each .2ns while the simulated times for Fe interstitials were 1ns. The reason for the difference in simulation time is that the time step for each type of simulation is different. The total number of time steps for each simulation is the same at 1000000. However, the time step for He-interstitial simulations is .2ps while Fe-interstitial simulations have a standard 1ps time step. The difference is due to the how quickly each atom thermally vibrates. Fe is heavier and vibrates more slowly than He does, hence Fe can have a longer time step than He while still retaining physical accuracy.

Computational Environment

Parts of this research were done on various machines. The majority of the work on self-interstitial dumbbells was done at LLNL on the Zeus supercomputer. The work on He interstitials was done at UNT on the Teragrid network of computers. The work on parallel replica dynamics was done at UNT on the Talon supercomputer.

Post-Processing Routines

After each simulation, the raw data needs to be processed to yield meaningful results. Each simulation outputs an entire snapshot of the positions and types of atoms at regular intervals. This is the raw data that is to be processed after the simulation is completed. For the He interstitial simulations all that needs to be done is to track the location of the helium atom and take the root-mean square average to determine diffusion coefficients. Both the complexity and time required to post-process the raw data are low. To correctly analyze the self-interstitial simulations more attention is required, and accurately tracking the desired object (dumbbell) can be challenging. Samples of post-processing code are given in the Appendix.

The first step in the processing of the interstitial simulations is to identify either the He atom (in the case of a He-interstitial) or a Fe self-interstitial dumbbell. For He this is easy as there is simply only one He atom in the simulation. For the Fe-interstitial a more lengthy process is required. Since the original additional Fe atom typically becomes a lattice atom after a few time steps above 400K, simply tracking the atom ID number is not an option. The key here is to think of each lattice position as being occupied by only one atom, but at the location of the dumbbell there are two atoms occupying one lattice position. The precise process of finding this lattice position for each lattice snapshot is

lengthy both in explanation and post-processing time. The position of the interstitial is recorded and used to compute the mean-squared displacement. This computation is then fed into Einstein's equation of diffusion to yield a value of the rate constant for each temperature (8 total temperatures). From this data, logarithmic plots (called Arrhenius plots) are created in order to find the rate equation for that specific composition of Cr.

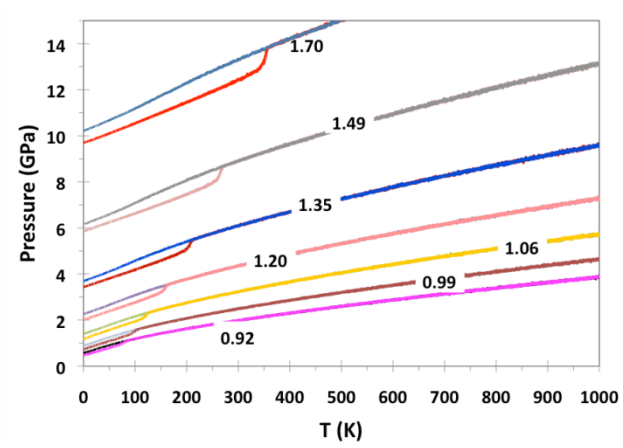
The post-processing for the He equation of state simulations is much more straightforward. The data from the simulation is analyzed to find a melting point for the solid-to-fluid transition and a freezing point for fluid-to-solid transition point. These points give appropriate thermodynamic information to yield an equation of state relevant to the temperature and pressure conditions in an actual metal.

RESULTS

Equation of State for He

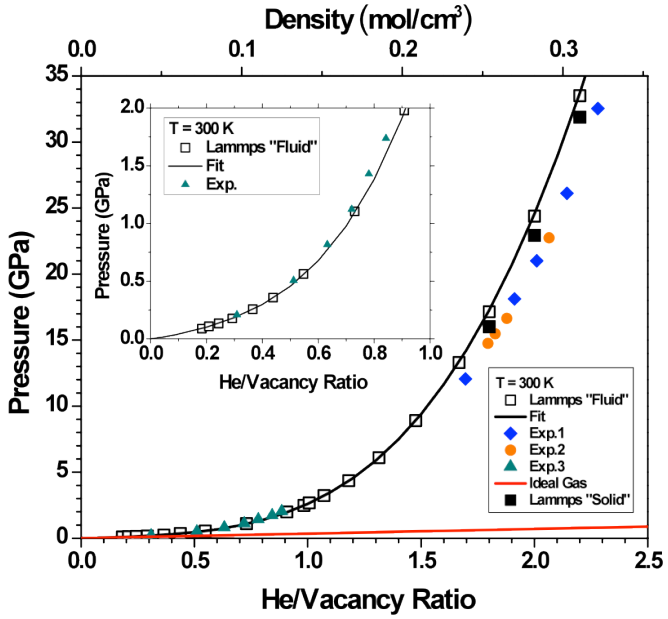
The helium found inside cavities within the steel lattice matrix is known to exist in both a solid and a liquid phase [30][31]. An accurate description of the He inside these bubbles is vital in reaching a quantitative understanding of radiation damage in these steels. The ternary potential used in this research is employed to give an estimate for an equation of state for the He inside the bubbles. This is an important step in checking the potential we are using for physical accuracy.

The equation of state is a relation between the pressure, density, and temperature of the He in our potential. Density, in our case, is measured relative to the Fe bcc lattice. So, one He atom in one Fe vacancy is taken as 1He/Vacancy (ρ=1). In order to get this equation of state, He atoms are arranged in a bcc structure with the same lattice spacing as Fe as to start with a density of one. The simulation starts by holding the temperature fixed at 0K and either compressing or expanding the volume by a specified amount. Once the desired density is established, the density is held fixed and the temperature rises from 0K to 1000K and then lowers back to 0K at the same rate while the corresponding pressure is recorded (Fig. 17). This gives thermodynamic data to construct an equation of state. Fig. 18 gives a pressure versus density plot determined by the LAMMPS simulations and compares that with experimental data. The difference in the “Fluid” and “Solid” curves with the experimental data is due to the finite size effects of the sample and defects that are frozen during cooling for the “Fluid” and “Solid” data [18].



The pressure versus temperature for various densities are labeled in units of He/Vac. The sudden jumps in the heating(lower) curves represent a phase change from solid to fluid. Upon cooling(upper curves), significant hysteresis is apparent due to high rates of heating/cooling. If the rates of heating and cooling were slow enough and no defects developed in the solid, the heating and cooling curves would be the same.

Fig. 17. He EOS thermodynamic data [18].



This is an example of the thermodynamic information simulated in LAMMPS at 300K compared to experimental data. The inset focuses on the low-density region. Ideal gas behavior is shown for comparison. The "Solid" and "Fluid" data refers to limitations inherent in the simulation discussed in the text. Experimental data is given in [29] (blue diamonds),[30] (red circles), and [31] (green triangles)

Fig. 18. Simulation data compared to experiment [18].

The equation of state sought is of the form

$$P(\rho,T) = a_0(\rho) + a_1(\rho) \times T + a_2(\rho) \times T^2$$

Eqn. 9

The data from the simulation is used to find the coefficients for each polynomial $a_n(\rho)$.

These polynomials are

$$a_0 = -5.98E-01\times\rho^5 + 3.29E+00\times\rho^4 - 1.76E+00\times\rho^3 + 3.76E-01\times\rho^2 - 1.0E-02\times\rho$$
$$a_1 = 1.02E-03\times\rho^5 - 4.24E-03\times\rho^4 + 4.66E-03\times\rho^3 + 2.19E-03\times\rho^2 + 1.07E-03\times\rho$$
$$a_2 = -6.49E-07\times\rho^5 + 2.39E+06\times\rho^4 - 2.09E+06\times\rho^3 - 9.56E-07\times\rho^2 + 1.20E-07\times\rho$$

Eqn. 10

This equation describes a surface in Pressure-Temperature-Density space (Fig. 19).

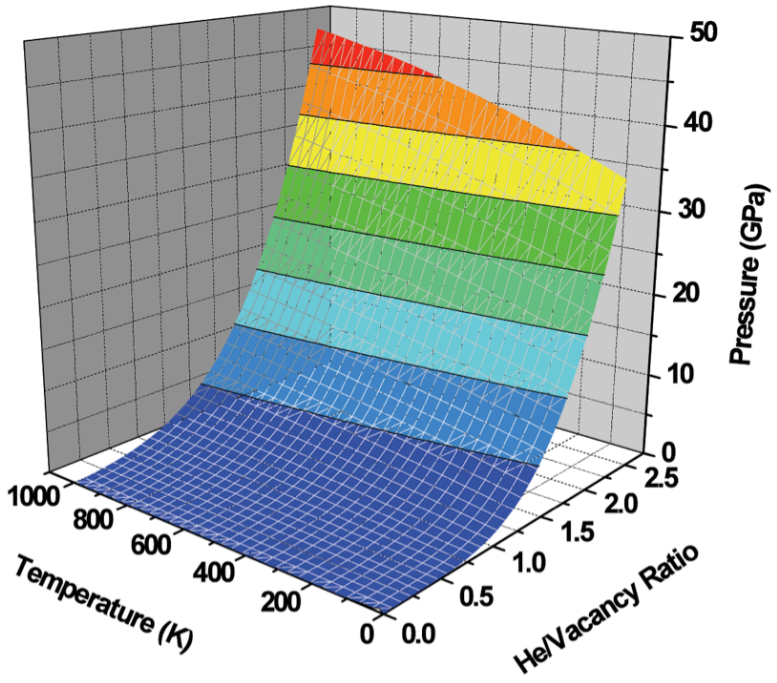


Fig. 19. Equation of state for helium in the fluid phase [18].

Formation Energies

The formation energies for Fe-interstitial, He-interstitial, He-substitutional, and vacancy were calculated using the LAMMPS *minimize* command. The energy of a pure α-Fe lattice was compared to the energy of the lattice when a defect is present. Table II lists the result for the formation energies.

Table II. Formation energies for point defects.

Formation Energies (eV) for Point Defects in an α-Fe Lattice			
He-interstitial	Fe-interstitial	He-substitutional	Vacancy
4.54	3.70	4.03	1.49

He-Interstitial

Point defects in systems with several thousands of atoms are commonly simulated using MD. The simulations for the He interstitials are analyzed by finding the mean-squared displacement and then plotting that data to get a linear fit for the slope of a line. This slope is used to find the diffusion constant for that specific temperature. After, an Arrhenius plot is made for the range of temperatures (300K-1000K with a 100K increment; Fig. 20 as an example). This Arrhenius plot yields all the valuable information about diffusivity (Table III). Fig. 21 gives the plot of the activation energies versus at%Cr. This graph shows an anomaly around 9%atCr.

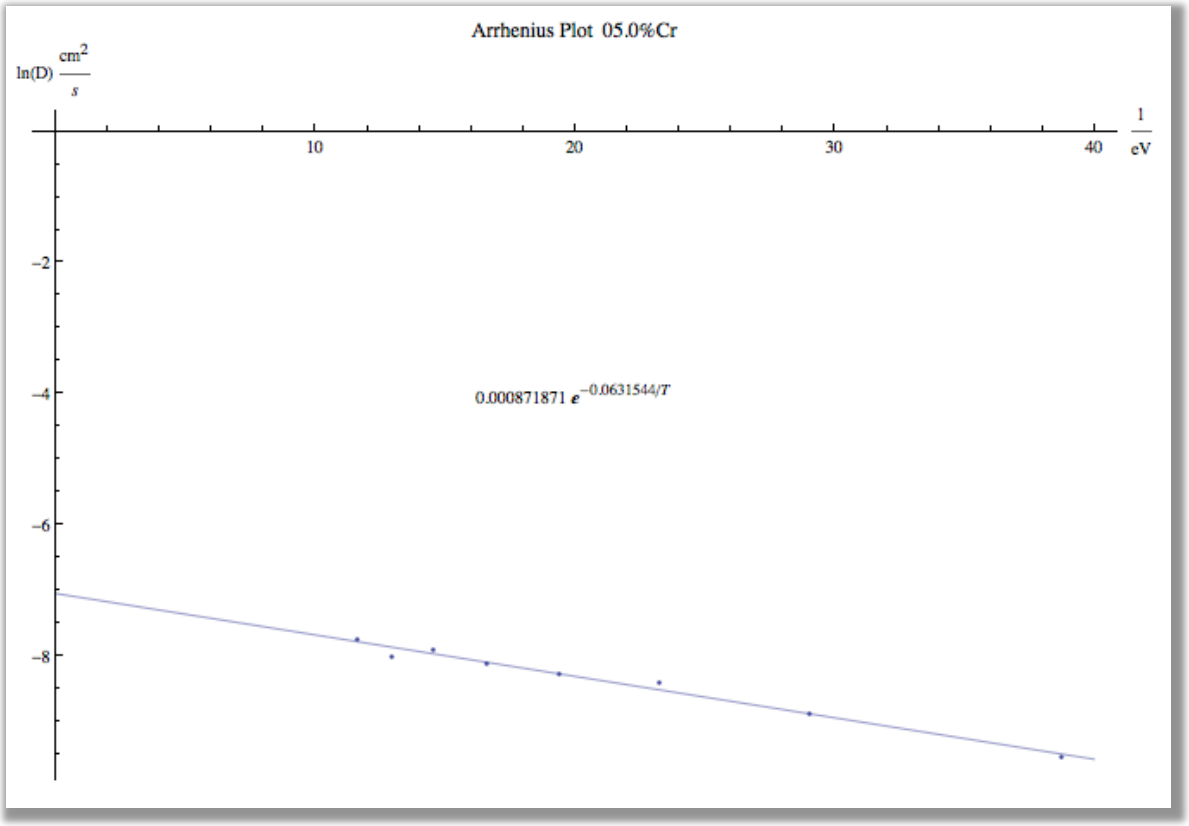


Fig. 20. Arrhenius plot for a He interstitial in an Fe-5.0at%Cr lattice.

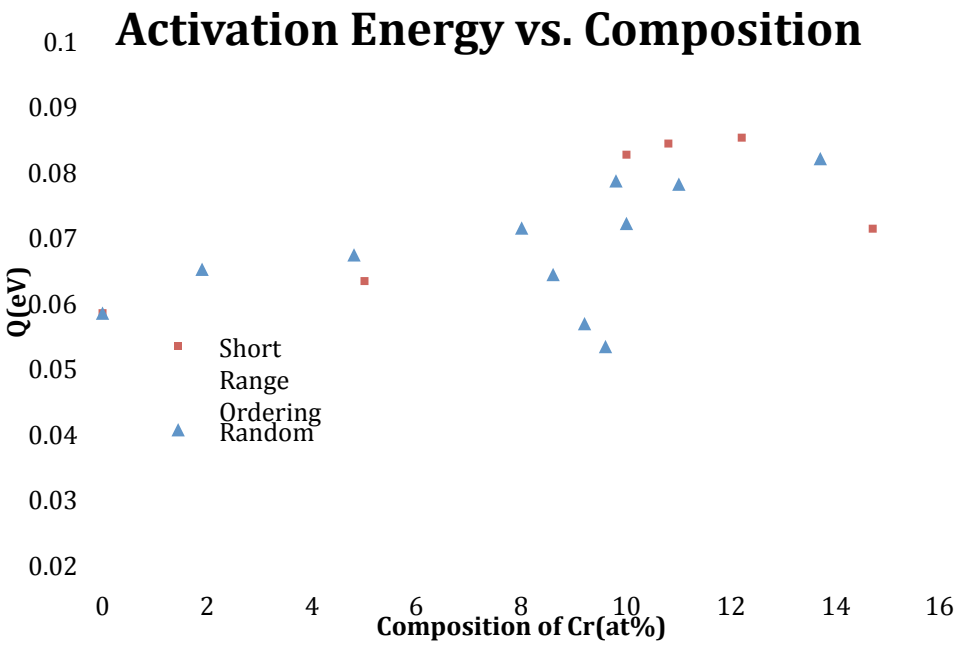


Fig. 21. Activation energy(Q) versus at%Cr.

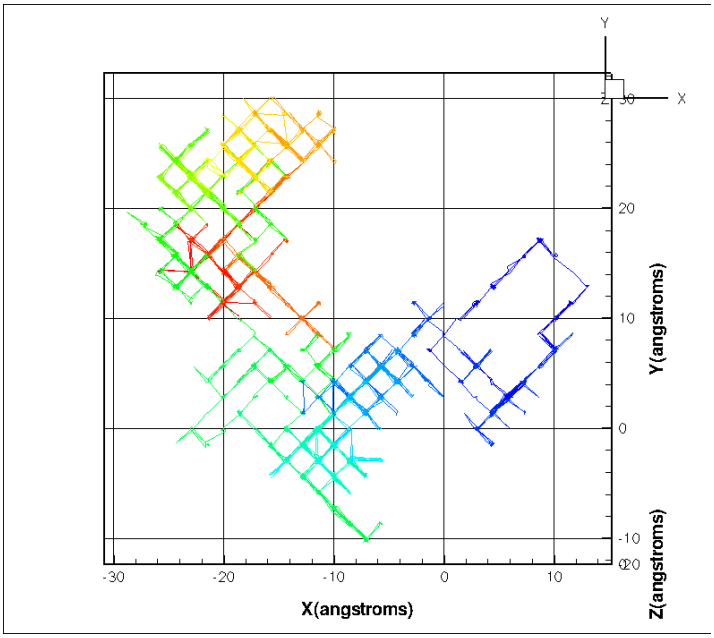
Table III. Rate constant data for He in varying compositions of Fe-Cr.

Arrhenius Equation for He in Fe-a%Cr $D = D_0 e^{-\frac{Q}{k_B T}}$		
Cr composition (at%)	**Activation EnergyQ(eV)**	**Pre Factor D_0(cm²/s)**
0	0.058	0.00095
Short Range Ordering (SRO)		
5.0	0.063	0.00087
10.0	0.083	0.00100
10.8	0.084	0.00097
12.2	0.085	0.00094
14.7	0.071	0.00065
Random Ordering		
1.9	0.065	0.00096
4.8	0.067	0.00090
8.0	0.071	0.00091
9.2	0.057	0.00069
9.6	0.053	0.00056
10.0	0.072	0.00091
11.0	0.078	0.00094
13.7	0.082	0.00091

Fe-Interstitial

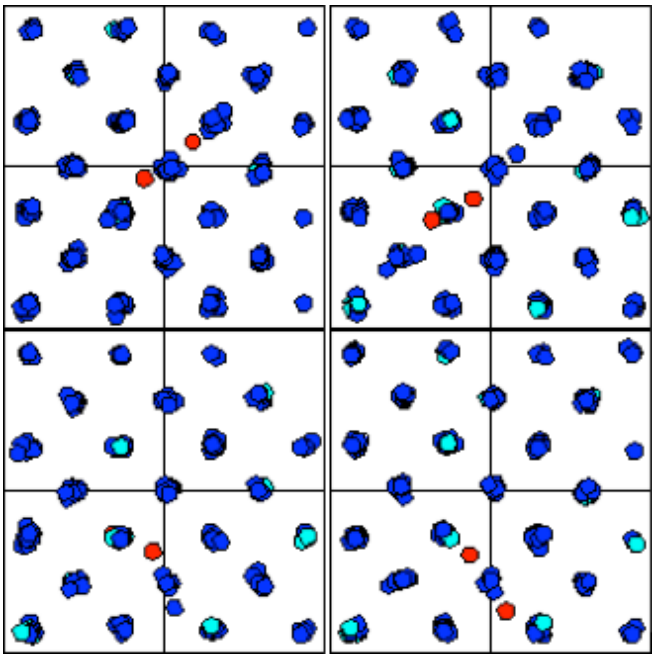
The same procedure used in the He-interstitial case is applied to the Fe-interstitial. As noted before, finding the dumbbell is much more difficult than locating a He atom (Appendix). After the dumbbell is located, the path of the center of mass is plotted in Fig. 22. The coordinates in this graph are unwrapped so that the periodic boundary conditions are taken into account and the absolute distances are shown. The dumbbell follows a *relatively* strict path along the <110> family of direction. The paths that are not along these directions could be because snapshots of the lattice were not taken frequently enough to capture the motion solely along the allowed directions or the dumbbell moved in a non-closed-packed direction for a short time. Figs. 23 and 24 show successive snapshots of an

interstitial dumbbell rotating and a plot of all dumbbells found during one simulation to illustrate that mixed dumbbells are preferred. Finally, Fig. 25 shows an Arrhenius plot for an Fe-interstitial, and Table IV gives a summary of the Arrhenius data for the Fe-interstitial.



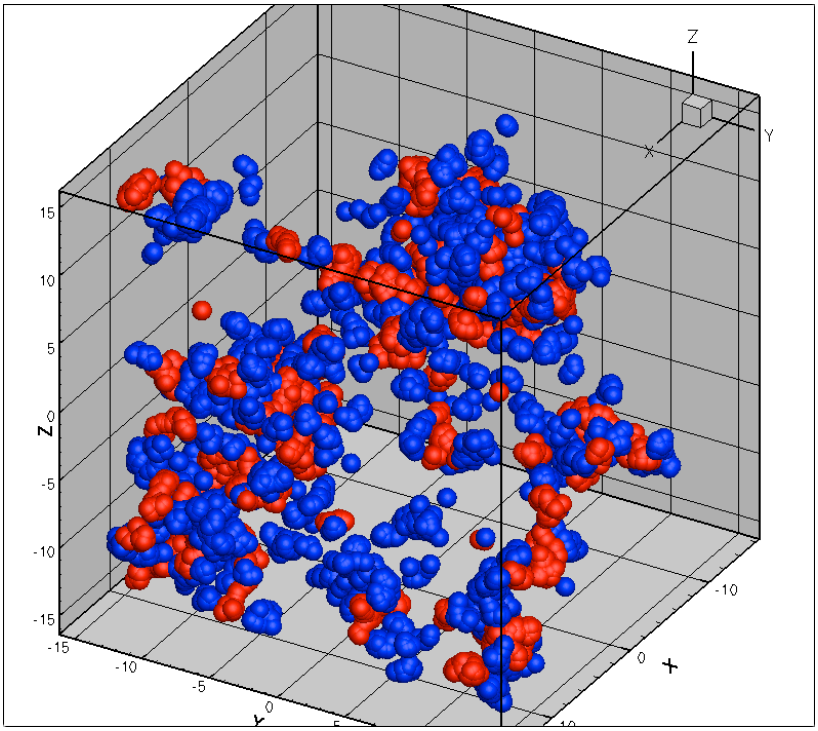
The path of the center of mass of the interstitial dumbbell in an Fe-10at%Cr lattice at 700K. The lattice has some short-range ordering. The bluer parts are the beginning of the path and redder part is the end of the path. Notice that the dumbbell almost exclusively travels along the <110> directions.

Fig. 22. Center of mass path for an interstitial dumbbell.



Four successive frames of an interstitial dumbbell (red) in an Fe (dark blue) 10at% Cr (light blue) lattice with SRO. Sequence is left to right, top to bottom. The discrepancy in the apparent amount of Cr between the first and second frames arises due to periodic boundaries.

Fig. 23.Interstitial dumbbell rotation.



This is a three dimensional picture of every dumbbell during a 1ns run. (Fe atoms are blue, Cr atoms are red) This illustrates the preference for the dumbbell being in a mixed Fe-Cr state rather than the pure Fe-Fe state. The amount of Cr is just 10at% but clearly more than 10% of the dumbbells are mixed.

Fig. 24. Interstitial dumbbell composition.

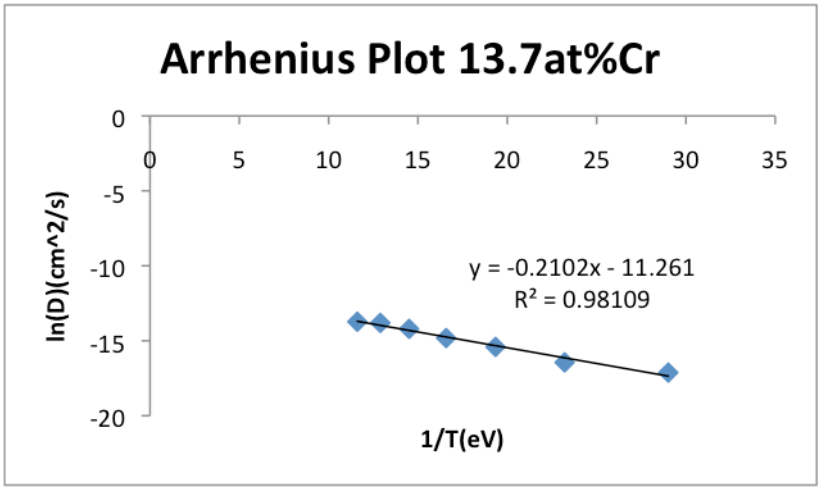


Fig. 25. Arrhenius plot for a Fe interstitial in an Fe-13.7at%Cr lattice.

Table IV. Arrhenius Fe interstitial data in varying compositions of Fe-Cr.

Arrhenius Equation for an Fe Interstitial in Fe-a%Cr $D = D_0 e^{-\frac{Q}{k_B T}}$		
Cr composition (at%)	**Activation Energy Q(eV)**	**Pre Factor D_0(cm^2/s)**
0	0.28	0.000104
Short Range Ordering (SRO)		
5.0	0.26	0.000052
10.0	0.21	0.000016
Random Ordering		
4.8	0.26	0.000042
9.6	0.21	0.000015
13.7	0.21	0.000013

CONCLUSION

The He equation of state data matches well with the experimental data. This will allow for more detailed studies and OKMC models to be developed. Additionally, the agreement within the experiment is a good check for accuracy for the new ternary potential being used. The results for the Fe interstitial activation energy are in good agreement with accepted values (Table V). The He-interstitial activation energy obtained in this research is somewhat different than values found in [32](Table VI). However, this research uses a different potential from [32]. Tables 3&4 and Fig. 21 indicate that there is not a strong correlation between migration energies of interstitials and lattice ordering.

Table V. Comparison of Fe interstitial energies.

Activation Energy for Fe Interstitial in an Fe BCC Lattice, Q(eV)			
This work	Resistivity Recovery [33]	*ab-initio* [34]	Different EAM potential [34]
.28	.27	.30	.31

Table VI. Comparison of He interstitial energies.

Activation Energy for He Interstitial in an Fe Lattice and Fe10at%Cr Lattice, Q(eV)			
Pure Fe Lattice		Fe10at%Cr	
This work	[32]	This work	[32]
.058	.078	.072	.078

The data gathered from this research can be used to further study the Fe-Cr surrogate system. By studying and cataloging the rates of point defect migration, this research will aid in developing rate theory models for OKMC simulations by providing the rates of migration for He-interstitials and Fe-interstitials. The He EOS data will also aid in making accurate, larger-scale models of the effects of He-bubbles in the Fe-Cr system.

Altogether, this information will give insight into how real Fe-Cr based steels behave in reactor conditions and may lead to better steels for the next generation of reactors.

FUTURE WORK

This Fe-Cr-He potential can be explored in many other ways. Once point-defect migration data is fully gathered, clusters of point defects and groups of vacancies combined with helium can be studied. The study of vacancy migration and the He bubble migration can be achieved through the use of LAMMPS' "prd" command. Work has indeed already begun on exploring some of these effects on the atomistic level using MD.

New potentials can be developed in order to encompass more alloying elements found in real steels. The potential used in this research only considers Fe, Cr, and He interactions. A more complex potential could take into account elements such as Ni or Mo in order to gain a more accurate picture of how steel behaves.

Even with accelerated dynamics methods such as PRD, the simulations in this research are still limited to MD timescales (at most a few μs). The next step is to use a rate theory approach to these objects such as KMC. Rate theory will allow for better assessment of general behavior and lead us closer to an accurate description of the way steels behave during irradiation.

APPENDEX A

SAMPLE LAMMPS INPUT SCRIPTS

Fe Interstitial

This is an example script used for running Fe interstitial simulations within a pure Fe lattice.

```
###self-interstitial point-defect simulation
###runs several temperatures sequentially
units         metal

atom_style     atomic
boundary       p p p
newton         on

read_data      /users/jwh0118/Self_Interstitial/0%Cr/Fe0Cr-Fe_SI.dat

pair_style     eam/cd
pair_coeff     * * /users/jwh0118/fecrhe-CD-EAM-Lammps-v1.fcn Fe Cr He

mass          1  55.8470
mass          2  51.9961
mass          3   4.0026

neighbor       1.0 bin
neigh_modify    every 1 delay 10 check yes

thermo         1000

variable       t index 300 400 500 600 700 800 900 1000

label          loop

if            $t != 300 then "undump 1"

variable       p equal v_t-100        ##previous temperature

shell          mkdir $tK

velocity       all create $p 87289 rot yes dist gaussian

log           $tK/$t.log

fix           integrate all nve
```

```
fix          T_control all temp/rescale 5 $p $t 10. 0.75

timestep       .001

# equilibrate
run          8000

unfix         T_control
fix          integrate all nve

reset_timestep  0

dump          1 all custom 100 $tK/frames.dat type mass x y z

run          1000000

next         t

jump          in.SI loop
```

He Interstitial

This is part of the script used in the He interstitial simulations. This script is intended to be called from a larger Unix submission script.

```
units                   metal
atom_style              atomic
boundary                p p p
newton                          on
read_data               $atomFil

group                   helium type 3
pair_style              eam/cd
pair_coeff              * * $potFil Fe Cr He

velocity                all create $temperature 4928459 rot yes dist gaussian

fix                     integrate all nve
fix                     T_control all temp/rescale 5 $temperature $temperature 10. 0.75
fix                press_ctrl all press/berendsen xyz 0 0 100.0

thermo_modify          flush yes
thermo              20
```

```
thermo_style        custom step temp pe etotal press lx ly lz

timestep            0.0001

run                 100000

unfix               T_control
unfix               press_ctrl
fix                 diffCoords helium coord/original
fix                 integrate all nve
compute             dis helium displace/atom diffCoords

dump                1 helium custom 50 $HeFile c_dis[12] c_dis[13] c_dis[14] c_dis[15]

run             2000000
```

Sample Atom File

This is just the heading of an atom file used for the “read-data” command.

```
He interstitial within an Fe10.8at%Cr lattice with SRO

-14.338 14.338 xlo xhi
-14.3377 14.3377 ylo yhi
-14.3478 14.3478 zlo zhi
2001 atoms
3 atom types

Atoms

1 2 -14.2961 14.3137 -14.3025
2 1 -11.6175 -14.312 -14.3001
3 1 -8.75495 -14.1872 -14.3108
4 1 -5.92232 14.286 -14.332
5 1 -2.98819 -14.2912 -14.3178
6 1 -0.221085 14.3037 14.343
7 1 2.90952 14.2107 14.2198
8 1 5.8067 14.2376 -14.3389
9 2 8.59681 -14.3106 -14.3251
10 1 11.6741 14.3282 14.2405
11 1 -14.2695 -11.611 14.294
12 1 -11.4255 -11.3983 -14.3047
13 1 -8.44187 -11.3968 14.3092
14 1 -5.75463 -11.4549 -14.3225
….
```

APPENDEX B

POST-PROCESSING CODE

He Interstitial

The following is the post-processing code used to generate the mean-squared displacement data for He interstitials written by Dr. Srinivasan in C.

```
/*************************************************************************
********
 *           He Mean Squared Displacement from LAMMPS output
*

*************************************************************************
*******
 *         Usage  : "HeMSD.x -I [Infile] -O [outfile] -v"
*

*************************************************************************
*******
 * Description: The program reads He LAMMPS He position file and
calculates  *
 *              Mean Squared Displacement of He.
*

*************************************************************************
*******/
#include <stdarg.h>/* variable arg handling */
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>/* size_t               */
#include <string.h>
#include <math.h>
/*-----------------------------------------------------------------
-------*/
externint strcasecmp(constchar *, constchar *);
/*-----------------------------------------------------------------
-------*/
#define      BUF_SIZE     1024
#define      MAX_A        2
/*-----------------------------------------------------------------
-------*/
constint    FRAME_STEP = 50;
constint    TOTAL_STEP = 3000000;   /* May need to INCREASE this */
constdouble PERCENTAGE = 0.2;
/*-----------------------------------------------------------------
-------*/
typedefstruct cm_data{
int    tstep;
```

```
double x[MAX_A], y[MAX_A], z[MAX_A], r[MAX_A];
} CMdataT;
/*-------------------------------------------------------------------
-------*/
CMdataT *FrameVecP = NULL;
/*-------------------------------------------------------------------
-------*/
/* Global Variables here */
char        IFname[BUF_SIZE];
char        OFname[BUF_SIZE];
/*-------------------------------------------------------------------
-------*/
void Error(char *msg, ...)
{
  va_list args;
char    buf1[BUF_SIZE];

/* Check if error message fits in buf[]; If the array bounds are not
     checked there may be overflow and coredump.   */
if( (strlen(msg) ) >=  BUF_SIZE )
    strcat( buf1," Error Buffer OverFlow. Can't EVEN Print Correct
Msg\0" );

/* Flush all buffers */
  fflush( stderr );

/* handle variable arguments */
  va_start(args, msg);
  vsprintf(buf1, msg, args);
  va_end(args);

/* Print the message */
  fprintf(stderr, "\n***********\n%s\n***********\n", buf1 );
  fflush( stderr );

  exit( -1 );
} /* void Error(char *msg, ...) */
/*-------------------------------------------------------------------
------*/
/*************************
 * main(): Main function *
 *************************/
int main(int argc, char *argv[])
{
char    inp_linE[BUF_SIZE];
  FILE    *ifp    = NULL;           /* input  file ptr
*/
  FILE    *ofp    = NULL;           /* output file ptr
*/
```

```
int     totsteps, stepsize, maxitrns;
int     nAtoms, c, i, j;
int     ifFlag, ofFlag, inFlag, isFlag;
int     verbose = 0;                /* Default: NOT verbose mode
*/
int     nFrames = 0;
int     tStep   = 0;
int     istop   = 0;
double  average = 0.0;

externchar *optarg;               /* For getopt()
*/
externint   optind;                /* For getopt()
*/

  ifFlag   = 0;
  ofFlag   = 0;
  inFlag   = 0;
  isFlag   = 0;
  totsteps = 0;

/* Number of frames: This determined MAX ARRAY SIZE */
  nFrames = TOTAL_STEP/FRAME_STEP;

/* Allocate memory for frames */
  FrameVecP =  ((CMdataT*) calloc(nFrames, sizeof(CMdataT)));
if( FrameVecP == NULL ) Error("$$$ main(): calloc FAILED\n");

/* Handle command line args */
while( (c = getopt( argc, argv, "I:O:N:S:v" )) != EOF )
switch (c) {
case'v':                          /* Verbose mode
*/
      verbose  = 1;
break;
case'I':                          /* Input  File Name
*/
/* In file name */
      sprintf( IFname,"%s", optarg );
      ifFlag++;                       /* Input file read
*/
break;
case'O':                          /* Output File Name
*/
      sprintf( OFname,"%s", optarg );
      ofFlag++;                       /* output file read
*/
break;
case'N':                          /* NSTEPS used in runs
*/
```

```
      totsteps = atoi(optarg);
      inFlag++;                         /* Nsteps value read
*/
break;
case'S':                           /* STEP SIZE
*/
      stepsize = atoi(optarg);
      isFlag++;                         /* STEP SIZE    read
*/
break;
default:
      (void) fprintf(stderr, "## ERROR in USAGE\n");
      Error("1) main():USAGE: '%s -I infile -O outFil -N totsteps -S
stepsize -v'\n",
        argv[16]);
    } /* while{switch()} */

if( ifFlag   != 1  )    Error("main(): Input  file  NOT read
correctly\n");
if( ofFlag   != 1  )    Error("main(): Output file  NOT read
correctly\n");
if( inFlag   != 1  )    Error("main(): Input Nsteps NOT read
correctly\n");
if( isFlag   != 1  )    Error("main(): Input stepsize NOT read
correctly\n");

if(verbose) fprintf(stderr, "OPENING FILE %s TO READ\n", IFname);
    (void) fprintf(stderr, "Input/Output File= (%s, %s)\n", IFname,
OFname);



/* OPEN Input file */
if( !(ifp = fopen(IFname, "r") ) )
    Error("2) main(): Unable to Open InFile: %s\n", IFname );

/* Loop and Read till the end of the file (EOF) */
  i = 0;
/* for(; !feof(ifp); )   {
     while(!feof(ifp))   {  */
  maxitrns = totsteps/stepsize;
for(i=0; i < maxitrns; i++) {
double x, y, z, r;

/* CHECK if ARRAY SIZE SUFFICIENT */
if( i == nFrames )
      Error("3a) main(): INCREASE FrameVecP[] array size to > %d\n",
nFrames);

/* "ITEM: TIMESTEP": read and discard */
```

```
    fgets(inp_linE, sizeof(inp_linE), ifp );

/* "TIMESTEP#": read and save */
    fgets(inp_linE, sizeof(inp_linE), ifp );
    sscanf(inp_linE, "%d", &tStep );
    FrameVecP[i].tstep = tStep;

/* "ITEM: NUMBER OF ATOMS": read and discard */
    fgets(inp_linE, sizeof(inp_linE), ifp );

/* "NATOMS#": read and save */
    nAtoms = 0;
    fgets(inp_linE, sizeof(inp_linE), ifp );
    sscanf(inp_linE, "%d", &nAtoms );

if(nAtoms > MAX_A)
      Error("3b) main(%d): InterstitialArray handles %d atoms; has %d
now\n",
        i, MAX_A, nAtoms);

/* "ITEM: "BOX BOUNDS": read and discard */
    fgets(inp_linE, sizeof(inp_linE), ifp );

/* Values of xlo/xhi: read and discard */
    fgets(inp_linE, sizeof(inp_linE), ifp );

/* Values of ylo/yhi: read and discard */
    fgets(inp_linE, sizeof(inp_linE), ifp );

/* Values of zlo/zhi: read and discard */
    fgets(inp_linE, sizeof(inp_linE), ifp );

/* "ITEM: ATOMS c_dis[12] c_dis[13] c_dis[14] c_dis[15]": read/discard
*/
    fgets(inp_linE, sizeof(inp_linE), ifp );

/* Scan in interstitial coordinates */
for(j=0; j < nAtoms; j++) {
      fgets(inp_linE, sizeof(inp_linE), ifp );
      sscanf(inp_linE, "%lf %lf %lf %lf", &x, &y, &z, &r );
      FrameVecP[i].x[j] = x;
      FrameVecP[i].y[j] = y;
      FrameVecP[i].z[j] = z;
      FrameVecP[i].r[j] = r;

/*---- DEBUG ----*/
/* fprintf(stdout,
     "main(): Itrn(%05d)\tStep(%07d)\tnAtoms(%03d)\tx(%.4lf)\n",
     i, tStep, nAtoms, x); */
/*---- DEBUG ----*/
```

```
    } /* for(j) */
/* INCREMENT RECORD NUMBER */
/* i++; */
  }  /* for() */

/* Close opened input file */
  fclose(ifp);

/* Write out output configuration */
if(verbose) fprintf(stderr, "OPENING FILE %s TO WRITE\n", OFname);

/* OPEN Output file */
if( !(ofp = fopen(OFname, "w") ) )
    Error("4) main(): Unable to Open OutFile: %s\n", OFname );

  average = 0.0;
  istop   = (int) (((double) maxitrns) * PERCENTAGE);

/* ASK JEFF: (1) logic of the loop and (2) what is istop */
for(i = 0; i < istop; i++) {
int ave_num;

    ave_num =   0;
    average = 0.0;

for(j = 0; j+i < maxitrns; j++){
double  dx,  dy,  dz,  dr;
double dx2, dy2, dz2, dr2;

      dx  = FrameVecP[j+i].x[16]-FrameVecP[j].x[16];
      dx2 = dx*dx;

      dy  = FrameVecP[j+i].y[16]-FrameVecP[j].y[16];
      dy2 = dy*dy;

      dz  = FrameVecP[j+i].z[16]-FrameVecP[j].z[16];
      dz2 = dz*dz;

      average += (dx2+dy2+dz2);
      ave_num++;
    } /* for(j) */

if(ave_num != 0) average /= ((int) ave_num);

    fprintf(ofp, "%d  %lf\n", i*FRAME_STEP, average);
  } /* for(i) */

/* Close opened output file */
  fclose(ofp);
```

```
return(EXIT_SUCCESS);
}  /* main() */
```

Fe Interstitial

The next post-processing code is written in C++ and is used to identify the Fe dumbbell and calculate the mean-squared displacement written by the author.

```
//finds an interstitial dumbbell and creates a tecplot file, a file
with the unwraped coords and displacement, and an msd file with D
//this program needs a lammps dump file with x y z type mass csp coord
ack pe
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <vector>
#include <math.h>

usingnamespace std;

constint AT_ARR_LNGT = 2;//for dumbbell calculating
constint FRAME_STEP = 100;//for msd
constfloat PERCENTAGE = .2;//for msd

ifstream infile;
ofstream outfile, outfile2, outfile3;
string trash;
longint tstep,numatoms,t0;
double xlo,xhi,ylo,yhi,zlo,zhi,x,y,z,xi,yi,zi;

struct atom{
    double x,y,z,mass,csp,pe;
    int type,coord,ack,xln,yln,zln;//xln = x coordinate lattice
number, calculated by correct_int
};
struct coord{
    longint tstep;
    double x,y,z,r,msd;
};

vector<coord> cm_vec;
vector<coord> unwrap_vec;
vector<coord> msd_vec;

bool share_same_ints(atom&,atom&);
```

```
int nint(double);
double d_abs(double);
coord correct_int(atom&,double,double,double);
coord x_crt_int(atom&,double,double,double);
coord y_crt_int(atom&,double,double,double);
coord z_crt_int(atom&,double,double,double);
coord find_cm(atom&, atom&, double&, double&, double&, double&,
double&, double&);

int main(int argc,char *argv[]){ //needs an input file in the for of a
lammps dump file and creates 3 output files

    infile.open(argv[12],ios::in);
    outfile.open("unwraped_data.dat",ios::out);
    outfile2.open("reducedZONES.dat",ios::out);
    outfile3.open("msd.dat",ios::out);

    if( !infile.is_open() ) return -1;

//  outfile<<"t-step x y z r r_squared\n";
    outfile2<<"VARIABLES = X Y Z TYPE PE\n";
//  outfile3<<"t msd\n";

    atom dumbbell_atom[AT_ARR_LNGT];
    atom prev_dumbbell_atom[AT_ARR_LNGT];

    getline(infile,trash);//ITEM: TIMESTEP
    while(!infile.eof()){

     getline(infile,trash);//tstep
     stringstream(trash)>>tstep;
     getline(infile,trash);//ITEM: NUMBER OF ATOMS
     getline(infile,trash);//numatoms
               stringstream(trash)>>numatoms;
     getline(infile,trash);//ITEM: BOX BOUNDS
     getline(infile,trash);//xlo xhi
     stringstream(trash)>>xlo>>xhi;
     double X_BOX = xhi - xlo;
     double X_MAX = X_BOX / 2;
     double Ax = X_BOX / 10;
     getline(infile,trash);//ylo yhi
     stringstream(trash)>>ylo>>yhi;
     double Y_BOX = yhi - ylo;
     double Y_MAX = Y_BOX / 2;
     double Ay = Y_BOX / 10;
     getline(infile,trash);//zlo zhi
     stringstream(trash)>>zlo>>zhi;
     double Z_BOX = zhi - zlo;
     double Z_MAX = Z_BOX / 2;
     double Az = Z_BOX / 10;
```

```
      double MAX_PREV = 2*(Ax+Ay+Az)/3;
      getline(infile,trash);
      if(cm_vec.size()<1) t0=tstep;
      atom atoms[numatoms];
      for(longint i = 0; i < numatoms; i++ ){
            getline(infile,trash);

    stringstream(trash)>>atoms[i].x>>atoms[i].y>>atoms[i].z>>atoms[i].
type>>atoms[i].mass>>atoms[i].csp>>atoms[i].coord>>atoms[i].ack>>atoms
[i].pe;
            coord tempcrd = correct_int(atoms[i],Ax,Ay,Az);
            atoms[i].xln = (int)tempcrd.x;
            atoms[i].yln = (int)tempcrd.y;
            atoms[i].zln = (int)tempcrd.z;
            for(longint j = 0; j < i-1; j++){
    if(share_same_ints(atoms[i],atoms[j])){
                                        dumbbell_atom[16] = atoms[i];
                                        dumbbell_atom[12] = atoms[j];
                                  } //end if
                            } //end for
      } //end for
      outfile2<<"ZONE\n";
      for(int i = 0; i < AT_ARR_LNGT; i++){
            outfile2<<" "<<dumbbell_atom[i].x<<"
"<<dumbbell_atom[i].y<<" "<<dumbbell_atom[i].z<<"
"<<dumbbell_atom[i].type<<" "<<dumbbell_atom[i].pe<<"\n";
      } //end for
      coord temp = find_cm( dumbbell_atom[16], dumbbell_atom[12],
X_BOX, X_MAX, Y_BOX, Y_MAX, Z_BOX, Z_MAX);
      cm_vec.push_back(temp);
      int counter = cm_vec.size() - 1;
      if(counter == 0){
            x = cm_vec[counter].x;
            y = cm_vec[counter].y;
            z = cm_vec[counter].z;
            xi = cm_vec[counter].x;
            yi = cm_vec[counter].y;
            zi = cm_vec[counter].z;
      }
      else{
            if(cm_vec[counter-1].x - cm_vec[counter].x > X_MAX) x +=
X_BOX - cm_vec[counter-1].x + cm_vec[counter].x;
            elseif(cm_vec[counter].x - cm_vec[counter-1].x > X_MAX) x -
= X_BOX - cm_vec[counter].x + cm_vec[counter-1].x;
            else x += cm_vec[counter].x - cm_vec[counter-1].x;
            if(cm_vec[counter-1].y - cm_vec[counter].y > Y_MAX) y +=
Y_BOX - cm_vec[counter-1].y + cm_vec[counter].y;
elseif(cm_vec[counter].y - cm_vec[counter-1].y > Y_MAX) y -= Y_BOX -
cm_vec[counter].y + cm_vec[counter-1].y;
else y += cm_vec[counter].y - cm_vec[counter-1].y;
```

```
          if(cm_vec[counter-1].z - cm_vec[counter].z > Z_MAX) z +=
Z_BOX - cm_vec[counter-1].z + cm_vec[counter].z;
elseif(cm_vec[counter].z - cm_vec[counter-1].z > Z_MAX) z -= Z_BOX -
cm_vec[counter].z + cm_vec[counter-1].z;
else z += cm_vec[counter].z - cm_vec[counter-1].z;
      }
      coord tempcrd;
      tempcrd.tstep = tstep- t0;;
      tempcrd.x = x;
      tempcrd.y = y;
      tempcrd.z = z;
      tempcrd.r = sqrt(pow(x-xi,2)+pow(y-yi,2)+pow(z-zi,2));

      unwrap_vec.push_back(tempcrd);

      getline(infile,trash);//ITEM: TIMESTEP, this is here to set
eofbit if I reach the end of the file
    } //end while

    infile.close();
    outfile2.close();
    //average algorithm and print unwraped_data.dat
    double stop = unwrap_vec.size() * PERCENTAGE;
for(unsignedint i = 0; i < unwrap_vec.size(); i++){
      if(i < stop){
    unsignedint ave_num = 0;
    coord temp;
    for(unsignedint j = 0; j+i < unwrap_vec.size(); j++){
    temp.msd += pow(unwrap_vec[j+i].x-unwrap_vec[j].x,2) +
pow(unwrap_vec[j+i].y-unwrap_vec[j].y,2) + pow(unwrap_vec[j+i].z-
unwrap_vec[j].z,2);
      ave_num++;
    } //end for
    if(ave_num != 0) temp.msd /= ave_num;
      temp.tstep = i*FRAME_STEP;
    msd_vec.push_back(temp);
      } //end if
                outfile<<unwrap_vec[i].tstep<<" "<<unwrap_vec[i].x<<"
"<<unwrap_vec[i].y<<" "<<unwrap_vec[i].z<<" "<<unwrap_vec[i].r<<"
"<<pow(unwrap_vec[i].r,2)<<"\n";
        } //end for
outfile.close();

    double D = 0;
    double ave_num = 0;
    for(int i = 0; i < msd_vec.size(); i++){
      double mini_ave = 0;
      int mave_num = 0;
      for(int j = 0; j < msd_vec.size(); j++){
           if(i!=j){
```

```
                if(abs(msd_vec[j].tstep - msd_vec[i].tstep) ==
i*FRAME_STEP){
                    if(j>i) mini_ave += (msd_vec[j].msd -
msd_vec[i].msd)/(i*FRAME_STEP);
                    else  mini_ave += (msd_vec[i].msd -
msd_vec[j].msd)/(i*FRAME_STEP);
                    mave_num++;
                } //end if
            } //end if
     } //end for
     if(mave_num != 0) mini_ave /= mave_num;
     D += mini_ave;
     ave_num++;
    } //end for
    if(ave_num != 0) D /= ave_num;

    D /= 6;     //from Einstein's equation of diffusion

    outfile3<<"D = "<<D<<" (length units)^2 / (time step * frame
step)\n";
    for(int i = 0; i < msd_vec.size(); i++){
     outfile3<<msd_vec[i].tstep<<" "<<msd_vec[i].msd<<"\n";
    }

    outfile3.close();
    return0;
} //end main

bool share_same_ints(atom& a,atom& b){

    if(a.xln == -10) a.xln *= -1;
    if(a.yln == -10) a.yln *= -1;
    if(a.zln == -10) a.zln *= -1;
    if(b.xln == -10) b.xln *= -1;
    if(b.yln == -10) b.yln *= -1;
    if(b.zln == -10) b.zln *= -1;

    if(a.xln != b.xln || a.yln != b.yln || a.zln != b.zln)
returnfalse;

    returntrue;
}

double d_abs(double x){
    if(x <0) return -1*x;
    elsereturn x;
}

int nint(double x){   //nearest integer, modf returns fractional part
of x and stores integer part in 'integer'
```

```
    double integer;
    if(x>0){
     if(modf(x, &integer) <.5) return (int) integer;
     elsereturn (int) integer + 1;
    }
    else{
     if(modf(x, &integer) > -.5) return (int) integer;
elsereturn (int) integer - 1;
    }
}

coord correct_int(atom& a, double ax, double ay, double az){
    coord temp;
    double xint, yint, zint;
    double x = 2*a.x/ax;
    double y = 2*a.y/ay;
    double z = 2*a.z/az;
    float xdis, ydis, zdis;     //displacements from normalized
lattice site

    if(d_abs(modf(x, &xint)) <.5) xdis = d_abs(x) - abs(nint(x));
    else xdis = abs(nint(x)) - d_abs(x);
    if(d_abs(modf(y, &yint)) <.5) ydis = d_abs(y) - abs(nint(y));
    else ydis = abs(nint(y)) - d_abs(y);
    if(d_abs(modf(z, &zint)) <.5) zdis = d_abs(z) - abs(nint(z));
    else zdis = abs(nint(z)) - d_abs(z);

    if(xdis < ydis){  //choose direction with smallest displacement to
start
     if(xdis < zdis) temp = x_crt_int(a, x, y, z);
     else temp = z_crt_int(a, x, y, z);
    } //end if
    else{
     if(ydis < zdis) temp = y_crt_int(a, x, y, z);
     else temp = z_crt_int(a, x, y, z);
    } //end else

    return temp;
}

coord x_crt_int(atom& a,double x,double y,double z){
    coord temp;
    if(nint(x)%2 == 0){
if(nint(z)%2 == 0){
                        temp.z = nint(z);
                } //end if
else{
if(z >0){
if(nint(z)>z) temp.z = nint(z) - 1;
else temp.z = nint(z) + 1;
```

```
                        } //end if
else{
if(nint(z)<z) temp.z = nint(z) + 1;
else temp.z = nint(z) - 1;
                        } //end else
                } //end else
if(nint(y)%2 == 0){
                        temp.y = nint(y);
                } //end if
else{
if(y >0){
if(nint(y)>y) temp.y = nint(y) - 1;
else temp.y = nint(y) + 1;
                        } //end if
else{
if(nint(y)<y) temp.y = nint(y) + 1;
else temp.y = nint(y) - 1;
                        } //end else
                } //end else
                temp.x = nint(x);
        } //end if
else{
if(nint(z)%2 == 0){
if(z >0){
if(nint(z)>z) temp.z = nint(z) - 1;
else temp.z = nint(z) + 1;
                        } //end if
else{
if(nint(z)<z) temp.z = nint(z) + 1;
else temp.z = nint(z) - 1;
                        } //end else
                } //end if
else{
                        temp.z = nint(z);
                } //end else
if(nint(y)%2 == 0){
if(y >0){
if(nint(y)>y) temp.y = nint(y) - 1;
else temp.y = nint(y) + 1;
                        } //end if
else{
if(nint(y)<y) temp.y = nint(y) + 1;
else temp.y = nint(y) - 1;
                        } //end else
                } //end if
else{
                        temp.y = nint(y);
                } //end else
                temp.x = nint(x);
        } //end else
```

```
return temp;
}
coord y_crt_int(atom& a,double x,double y,double z){
    coord temp;
    if(nint(y)%2 == 0){
if(nint(x)%2 == 0){
                        temp.x = nint(x);
                } //end if
else{
if(x >0){
if(nint(x)>x) temp.x = nint(x) - 1;
else temp.x = nint(x) + 1;
                        } //end if
else{
if(nint(x)<x) temp.x = nint(x) + 1;
else temp.x = nint(x) - 1;
                        } //end else
                } //end else
if(nint(z)%2 == 0){
                        temp.z = nint(z);
                } //end if
else{
if(z >0){
if(nint(z)>z) temp.z = nint(z) - 1;
else temp.z = nint(z) + 1;
                        } //end if
else{
if(nint(z)<z) temp.z = nint(z) + 1;
else temp.z = nint(z) - 1;
                        } //end else
                } //end else
                temp.y = nint(y);
        } //end if
else{
if(nint(x)%2 == 0){
if(x >0){
if(nint(x)>x) temp.x = nint(x) - 1;
else temp.x = nint(x) + 1;
                        } //end if
else{
if(nint(x)<x) temp.x = nint(x) + 1;
else temp.x = nint(x) - 1;
                        } //end else
                } //end if
else{
                        temp.x = nint(x);
                } //end else
if(nint(z)%2 == 0){
if(z >0){
if(nint(z)>z) temp.z = nint(z) - 1;
```

```
else temp.z = nint(z) + 1;
                        } //end if
else{
if(nint(z)<z) temp.z = nint(z) + 1;
else temp.z = nint(z) - 1;
                        } //end else
                } //end if
else{
                        temp.z = nint(z);
                } //end else
                temp.y = nint(y);
        } //end else
return temp;
}
coord z_crt_int(atom& a,double x,double y,double z){
    coord temp;
    if(nint(z)%2 == 0){
if(nint(x)%2 == 0){
                        temp.x = nint(x);
                } //end if
else{
if(x >0){
if(nint(x)>x) temp.x = nint(x) - 1;
else temp.x = nint(x) + 1;
                        } //end if
else{
if(nint(x)<x) temp.x = nint(x) + 1;
else temp.x = nint(x) - 1;
                        } //end else
                } //end else
if(nint(y)%2 == 0){
                        temp.y = nint(y);
                } //end if
else{
if(y >0){
if(nint(y)>y) temp.y = nint(y) - 1;
else temp.y = nint(y) + 1;
                        } //end if
else{
if(nint(y)<y) temp.y = nint(y) + 1;
else temp.y = nint(y) - 1;
                        } //end else
                } //end else
                temp.z = nint(z);
        } //end if
else{
if(nint(x)%2 == 0){
if(x >0){
if(nint(x)>x) temp.x = nint(x) - 1;
else temp.x = nint(x) + 1;
```

```
                        } //end if
else{
if(nint(x)<x) temp.x = nint(x) + 1;
else temp.x = nint(x) - 1;
                        } //end else
                } //end if
else{
                        temp.x = nint(x);
                } //end else
if(nint(y)%2 == 0){
if(y >0){
if(nint(y)>y) temp.y = nint(y) - 1;
else temp.y = nint(y) + 1;
                        } //end if
else{
if(nint(y)<y) temp.y = nint(y) + 1;
else temp.y = nint(y) - 1;
                        } //end else
                } //end if
else{
                        temp.y = nint(y);
                } //end else
                temp.z = nint(z);
        } //end else
return temp;
}

coord find_cm(atom& a, atom& b, double& xb, double& xm, double& yb,
double& ym, double& zb, double& zm){
    coord cm;
    //should multiply by a.mass and b.mass and divide by sum
    if(a.x-b.x > xm) cm.x =
(a.x*a.mass+(b.x+xb)*b.mass)/(a.mass+b.mass);
    elseif(b.x-a.x > xm) cm.x =
(b.x*b.mass+(a.x+xb)*a.mass)/(a.mass+b.mass);
else cm.x = (b.x*b.mass+a.x*a.mass)/(a.mass+b.mass);
    if(a.y-b.y > ym) cm.y =
(a.y*a.mass+(b.y+yb)*b.mass)/(a.mass+b.mass);
    elseif(b.y-a.y > ym) cm.y =
(b.y*b.mass+(a.y+yb)*a.mass)/(a.mass+b.mass);
else cm.y = (b.y*b.mass+a.y*a.mass)/(a.mass+b.mass);
if(a.z-b.z > zm) cm.z = (a.z*a.mass+(b.z+zb)*b.mass)/(a.mass+b.mass);
    elseif(b.z-a.z > zm) cm.z =
(b.z*b.mass+(a.z+zb)*a.mass)/(a.mass+b.mass);
else cm.z = (b.z*b.mass+a.z*a.mass)/(a.mass+b.mass);

    return cm;
}
```

REFERENCES

[1] *United States Subcommittee on Generation IV Technology Planning on a Technology Roadmap for Generation IV Nuclear Energy Systems.* Report to Nuclear Energy Research Advisory Committee. Washington: Technical Roadmap Report, 2003.

[2] Allen, T.R., J.I. Cole, C.L. Trybus, and D.L. Porter. "The effects of long-time irradiation and thermal aging on 304 stainless steel." *Journal of Nuclear Materials* 282: 171-9 (2000).

[3] Ustinovshikov, Y., B. Pushkarev, and I. Igumnov. "Fe-rich portion of the Fe-Cr phase diagram: electron microscopy study." *Journal of Materials Science* 37: 2031-42 (2002).

[4] *Iron-Chromium (Fe-Cr) Phase Diagram.* 2009. http://www.calphad.com/iron-chromium.html (accessed 10 April 2010).

[5] Porter, D.A., and K.E. Easterling. *Phase Transformations in Metals and Alloys.* 2nd ed. Cheltenham: Stanley Thornes, 2001.

[6] Wakai, E., et al. "Swelling behavior of F82H steel irradiated by triple/dual ion beams." *Journal of Nuclear Materials* 318: 267-73 (2003).

[7] Trinkaus, H., and B.N. Singh. “Helium accumulation in metals during irradiation – where do we stand?” *Journal of Nuclear Materials*, 323: 229-42 (2003).

[8] Rothaut, J., H. Schroeder, and H. Ullmaier. "The growth of helium bubbles in stainless steel at high temperatures." *Philosophical Magazine A* 47: 781-95 (1983).

[9] Schroeder, H. "High temperature helium embrittlement in autenitic stainless steels - Correlations between microstructure and mechanical properties." *Journal of Nuclear Materials* 155-157: 1032-37 (1988).

[10] Jager, W., et al. “Density and pressure of helium in small bubbles in metals.” *Journal of Nuclear Materials* 111 & 112: 674-80 (1982).

[11] Fréchard, S., M. Walls, M. Kociak, J.P. Chevalier, J. Henry, and D. Gorse. "Study by EELS of helium bubbles in a martensitic steel." *Journal of Nuclear Materials* 393: 102-7 (2009).

[12] Stoller, R.E. "The influence of helium on microstructral evolution: Implications for DT fusion reactors." *Journal of Nuclear Materials* 174: 289-310 (1990).

[13] Schroeder, H., W. Kesternich, and H. Ullmaier. "Helium effects on the creep and fatigue resistance of austenitic stainless steels at high temperatures." *Nuclear Engineering and Design/Fusion* 2: 65-95 (1985).

[14] Kuramoto, E., N. Tsukuda, Y. Aono, M. Takenaka, and K. Kitajima. “Vacancy migration and void swelling in a stainless steel.” *Annal Reports of the Research Reactor Institute, Kyoto University*: 137-40 1984.

[15] Mansur, L.K. “Theory and experimental background on dimensional changes in irradiated alloys.” *Journal of Nuclear Materials* 216: 97-123 (1994).

[16] Terentyev, D., and N. Castin. "Mobility of small clusters of self-interstitial atoms in dilute Fe–Cr alloy studied by means of atomistic calculations." *Computational Materials Science* 46: 1178-86 (2009).

[17] Ullmaier, H., and E. Camus. "Low temperature mechanical properties of steels containing high concentrations of helium ." *Journal of Nuclear Materials* 251: 262-8 (1997).

[18] Caro, A., A.Stukowski, J.Hetherly, M.Caro, S.Srivilliputhur, L.Zepeda-Ruiz, P. Erhart, B. Sadigh. "Computational model of helium bubbles in FeCr alloys." [in progress]

[19] Daw, M.S., Baskes, M.I. “Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals.” *Phys Rev B*: 6443-53 1984.

[20] Caro, A., D.A. Crowson, and M. Caro. “Classical many-body potential for concentrated alloys and the inversion of order in iron-chromium alloys.” *Physical Review Letters* 95: 1-4 (2005).

[21] Stukowski, A., B. Sadigh, P. Erhart, and A. Caro. "Efficient implementation of the concentration-dependent embedded atom method for molecular-dynamics and Monte-Carlo simulations." *Modelling and Simulation in Materials Science and Engineering* 17: 1-13 (2009).

[22] Voter, A.F. “Hyperdynamics: Accelerated molecular dynamics of infrequent events.” *Physical Review Letters* 78: 3908-11 (1997).

[23] Voter, A.F. "Parallel replica method for dynamics of infrequent events." *Physical Review B* 57: 985-8 (1998).

[24] Voter, A.F., and M.R. Sørensen. "Accelerating atomisitic simulations of defect dynamics: Hyperdynamics, parallel replica dynamics, and temperature-accelerated dynamics." *Materials Research Society Symposium* 538: 427-39 (1999).

[25] Stoller, R.E., S.I. Golubov, C. Domain, and C.S. Becquart. “Mean field rate theory and object kinetic Monte Carlo: A comparison of kinetic models.” *Journal of Nuclear Materials*: 77-90 2008.

[26] Caturla, M.J., N. Soneda, T. Diaz de la Rubia, and M. Fluss. “Kinetic Monte Carlo simulations applied to irradiated materials: The effect of cascade damage in defect nucleation and growth.” *Journal of Nuclear Materials* 351: 78-87 (2006).

[27] Sandia National Laboratory. *pair_style eam command.*1-15-2010. http://lammps.sandia.gov (accessed 19 Feburary 2010).

[28] Le Roux, S., and V. Petkov. *Model Box Periodic Boundary Conditions - P.B.C.* 2009. http://isaacs.sourceforge.net/phys/pbc.html (accessed 31 March 2010).

[29] Loubeyre, P., R. Le Toullec, J. P. Pinceaux, H. K. Mao, J. Hu, and R. J. Hemley. "Equation of state and phase diagram of solid ^{4}He from single-crystal X-ray diffraction over a large *P-T* domain" *Physical Review Letters* 71: 2272-5 (1993).

[30] Mao, H. K., et al. "High-pressure phase diagram and equation of state of solid helium from single-crystal X-ray diffraction to 23.3 GPa." *Physical Review Letters* 60: 2649 (1998).

[31] Young, D. A., A. K. McMahan, and M. Ross. "Equation of state and melting curve of helium to very high pressure" *Physical Review* 24: 5119 (1981).

[32] Terentyev, D., N. Juslin, K. Nordlund, and N. Sandberg. "Fast three dimensional migration of He clusters in bcc Fe and Fe–Cr alloys." *Journal of Applied Physics* 105: 103509-1-12 (2009).

[33] Takaki, S., J. Fuss, H. Kugler, U. Dedek, and H. Schultz. "The resistivity recovery of high purity and carbon doped iron following low temperature electron irradiation." *Radiation Effects* 79: 87-122 (1983).

[34] Willaime, F., C.C. Fu, M.C. Marinica, and J. Dalla Torre. "Stability and mobility of self-interstitials and small interstitial clusters in α-iron: Ab initio and empirical potential calculations." *Nuclear Instruments and Methods in Physics Research B* 228: 92-9 (2005).