2017

Interactive Web-based Visualization of Atomic Position-time Series Data

Simron Thapa
Louisiana State University and Agricultural and Mechanical College, sthapa5@lsu.edu

INTERACTIVE WEB-BASED VISUALIZATION OF ATOMIC POSITION-TIME SERIES DATA

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Masters in Science in Computer Science

in

The Division of Computer Science and Engineering, School of EECS

by
Simron Thapa
B.E., Tribhuvan University, 2013
August 2017

Acknowledgments

I would like to express my sincere gratitude to my advisor Dr. Bijaya B. Karki for his guidance and encouragement in every aspect of my education at LSU. I would like to acknowledge the generousity of Dr. Doris L. Carver and Dr. Kisung Lee for serving on my MS thesis committee.

I would like to thank Dr. Dipta Ghosh and Dr. Suraj K. Bajgain for their support and thoughtful inputs. I would also like to thank all my friends for making my stay at LSU a memory of a life time. I greatly value their friendship. A special thank you to my room-mate turned friend turned family Karuna Kharel. Moreover, my collegue Saanu Raja Maharjan, who has been guiding me through all the technicality of the research.

Finally, I would like to thank my parents Ganesh Thapa and Sharmila Kumari Thapa for their unconditional love, support, and guidance to pursue my dreams and succeed in my career path. Also, my little brother Samyam Thapa for always being by my side giving me strength and motivation to keep moving foreward.

Table of Contents

List of Tables

List of Figures

List of Nomenclature

API	Application Programming Interface
BALL	Biochemical Algorithms Library
CSS	Cascading Style Sheet
DOM	Document Object Model
ECMA Script	European Computer Manufacturers Association Script
ES	Elastic Search
FPMD	First-Principles Molecular Dynamics
GET	Method used to submit data which will be visible in the page address field
GLSL	OpenGL Shader Language
GNU	Recursive acronym for GNU's Not Unix!
GUI	Graphical User Interface
HTML5	Hypertext Markup Language version 5
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MD	Molecular Dynamics
MVC	Model View Controller
MVP	ProjectionMatrix times ViewMatrix times ModelMatrix
OpenGL ES	Open Graphics Library for Embedded System
POST	Method used to submit data which will not be visible in the page address field
UV	"U" and "V" denote the axes of the 2D texture
VMD	Visual Molecular Dynamics
WebGL	Web Graphics Library

Abstract

Extracting and interpreting the information contained in large sets of time-varying three dimensional positional data for the constituent atoms of simulated material system is a challenging task. This thesis work reports our initial implementation of a web-based visualization system and its use-case study. The system allows the users to perform the desired visualization task on a web browser for the position-time series data extracted from the local or remote hosts. It involves a pre-processing step for data reduction, which involves skipping uninteresting parts of the data uniformly (at full atomic configuration level) or non-uniformly (at atomic species level or individual atom level). Atomic configuration at a given time step (snapshot) is rendered using the ball-stick representation and can be animated by rendering successive configurations. The entire atomic dynamics can be captured as the trajectories by rendering the atomic positions at all time steps together as points. The trajectories can be manipulated at both species and atomic levels so that we can focus on one or more trajectories of interest. They can be color-coded according to the additional information including the time elapsed and the distance traveled. The instantaneous atomic structure and the complete trajectories can be superimposed to help assess the 3D geometries and extents of the selected trajectories. The implementation was done using WebGL and Three.js for graphical rendering, HTML5 and Javascript for GUI, and Elasticsearch and JSON for data storage and retrieval within the Grails Framework. We have demonstrated the usefulness of our visualization system by analyzing the simulated position-time series for proton-bearing forsterite (Mg_2SiO_4) system – an abundant mineral of Earths upper mantle. Visualization reveals that protons (hydrogen ions) incorporated as interstitials are much more mobile than protons substituting the host Mg and Si cation sites. The proton diffusion appears to be anisotropic with high mobility along the x-direction, showing limited discrete jumps in other two directions. Our work at the present represents a simplistic (direct) web-based rendering of large atomic data sets. While the atomic structure can be animated at an interactive rate, the trajectory processing is slow, taking several minutes. We anticipate to further improve the system and use it in gaining useful structural and dynamical information from more materials simulation data.

Chapter 1
Introduction

Molecular dynamics (MD) simulations [1] generate immense amounts of positional data for the constituent atoms or molecules of the simulated real material system. The dataset is represented in the form of atomic configurations, corresponding to several snapshots. It is a three-dimentional, time-varying dataset. Visualization of these scattered MD data can greatly help in gaining meaningful insights into the structural and dynamical behavior of the materials system. The product of the total number of atoms in a single snapshot (i.e., atomic configuration) and the total number of snapshots (or time steps) determines the total size of the data. Different approaches have been used to manipulate the data and have a rapid navigation through the data. The two approaches commonly followed are animation and particle trajectories [5, 7].

Animation [6, 7] helps us visualize the system's configuration at each time step. It goes through all successive steps fast enough to show a smooth animation of the mobile atoms or molecules. The three-dimentional coordinates of individual atoms are generally represented as spheres (balls). We can show interatomic bonds/connections by sticks for better representation of atomic arrangements. However, we cannot develop an overall picture of the dynamical system through animation because of the disconnected information between different time steps. This leads us to another approach, i.e., particle trajectories [6, 7] which allow a complete representation of atomic position-time series data as points, thus enabling us to visualize the entire simulation information. The particle (atomic) trajectories can be further coded based on time elapsed, distance from the center, or total distance travelled. Here, the color of each pixel representing each position can be varied based on the colormap selected. Furthermore, we can superimpose the atomic structure with the atomic trajectories in such a way that maximum information can be retrieved. For instance, a particular species or atom can have its full trajectory rendered while remaining species or atoms can just be rendered as spheres at any time step.

In this work, we develop a user friendly web-based implementation of such atomic visualization system which can visualize the atomic position-time series data in the form of animation and trajectories. The material system used in this study is forsterite (Mg_2SiO_4) which is considered to be one of the most abundant minerals of Earth's upper mantle [22]. The forsterite is an orthorhombic structure consisting of Si-O tetrahedra with Mg atoms in the background. The simulated system of our concern contains 16 Mg_2SiO_4 formula units in the perfect crystal so it has total 112 atoms (16 magnesium, 32 silicon, 64 oxygen). For visualization analysis, we consider proton (hydrogen ion) defects in Mg_2SiO_4 incorporated as interstitials or at Mg/Si-vacant sites and explore the movement patterns of these protons.

1.1 Related Work

Spatio-temporal data sets are usually generated by numerical simulations like molecular dynamics. The data size can vary considerably consisting of hundered atoms to several thousands, to several millions, and also consisting of time steps from a few thousands to hundreds of thousands, to millions. Interactive visualization of such data is challenging and has been widely studied. Here, we review some existing visualization systems of relevance to our work.

1.1.1 Visual Molecular Dynamics (VMD)

VMD belongs to the most advanced programs and is distributed free of charge and open-sourced under UIUC (University of Illinois Urbana-Champaign) Open Source License [16]. Due to numerous visualization modes as well as built-in parsing plug-ins and collection of structure manipulation and visualization tools, VMD can process a wide variety of formats. It is written in C++ and utilizes Python plugins [16]. However, it does not provide much flexibility to the user to manipulate their positional data set based on their analysis requirement. These manipulations could result into multiple data formats. Also, it is a desktop-based application and needs to be installed on the local computer.

1.1.2 BALL/BALLView

BALL [15] (Biochemical Algorithms Library) is an application framework implemented in C++ which consists of various algorithms, data structures and classes and is useful for developing biochemical applications specifically designed for Rapid Software Prototyping. Other remarkable features includes file import/export, analysis, modification, visualization, python scripting interface, etc. BALLView is a stand-alone molecule visualization application built using QT GUI toolkit and BALL framework. However, visualization and analysis of system dynamics of large data sets and treating each species and atoms within the species individually is difficult to achieve using BALL.

1.1.3 Jmol/JSmol

Jmol [14] is significantly different from most of the aforementioned applications because it is written in Java, making it not only cross-platform by default but also suitable for web deployment via Java applets. It is a lightweight program having virtually no features besides molecule visualization. A fall-back solution (JSmol), implemented using JavaScript-only (HTML5/WebGL) version, can be used on computers with no Java. However, it has poor performance in Internet Explorer and Opera, and JavaScript is much more slower than Java [14]. The desktop application, Java applet, JavaScript alternative and a module that can be integrated into any Java application (JmolViewer) are free and open-source under GNU LGPL (lessser general public license) [8]. It doesnot provide molecular trajectory visualization and manipulation settings.

1.1.4 AtomViz

AtomViz [5] is a visualization application (developed at Louisiana State University) intended to address the rendering and analysis needs of atomic data generated from first principles molecular dynamics simulations. It is a platform dependent software built in C/C++ and works only on windows operating system. The format of the data sets we use in this study is similar to that AtomViz uses. Previously, we tried working on the AtomViz system inorder to make it more stable, usable and user friendly. Because of its complex nature and limitations, we prefer to develop

a new web-based application which is platform independent and allows interactive visualization of more diverse data sets.

1.2 Statement of Problem

This work focuses on the implementation of web-based interactive atomistic visualization of the atomic positional dataset from molecular dynamics simulations. Data loading of moderately large-sized datasets in a light weight web application is a challenge. Rendering atomic trajectories in a flexible way to analyze their geometries and extents in the 3D space is desirable. Also, superimposing the atomic structures (or configurations) at a time step allows us to examine the trajectories in relation to the background structure. We want to explore the dynamical behavior of protonic defects in Mg_2SiO_4 (forsterite) system for understanding the underlying mechanism of proton mediated electrical conduction in this key mineral.

1.3 Thesis Organization

The organization of the remaining of the dissertation is as follows. In Chapter 2, we explain the concept of web based visualization, the WebGL graphics pipeline and the languages and techniques used in the system implementation. In Chapter 3, we talk about position-time series data generated from molecular dynamics simulations and the forsterite (Mg_2SiO_4) system. In Chapter 4, we describe the methods followed for data manipulation and data reduction. In Chapter 5, we present the concepts used for data rendering, their structural modeling and dynamical properties, and superimposition strategies. We also present the time analysis of the system. In Chapter 6, we present the visualization-based analysis of Mg_2SiO_4 forsterite system containing proton defects, and demonstrate the applicability of our web-based visualization system.

Chapter 2
Web-based Implementation

WebGL and HTML5 are the major languages chosen for developing the project. Different techniques used for data storage, data manipulation, data rendering, and creation of GUI are briefly dicussed along with the high level system architecture.

2.1 WebGL and Three.js

WebGL [19] is a web standard for low-level 3D graphics API (Application Programming Interface) which is based on OpenGL ES (Open Graphics Library for Embedded System) and is exposed to ECMAScript (Eurpean Computer Manufacturers Association Script) via the HTML5 (Hypertext Markup Language version 5) canvas element. It is a shader-based API using GLSL (OpenGL Shader Language) which brings plugin-free 3D to web and is supported by major browser vendors like: Google (Chrome), Apple (Safari), Mozilla (Firefox), and Microsoft (Edge). It is a cross-platform and royalty-free API. The Figure 2.1, shows the graphics pipe-line of WebGL.

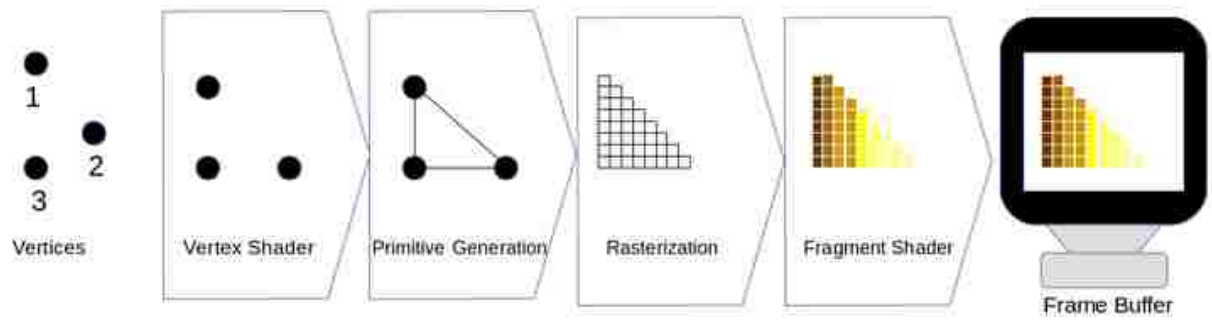

Figure 2.1: WebGL graphics pipe-line

Shader language code [10] is the control code in any WebGL application which is written in JavaScript and communicates with the GPU (Graphics processing unit). Firstly, in GPU programming we describe the geometry, normals, UVs (axes of the 2D texture) and set up the parameters

that will go into shading calculations [19]. Secondly, we program the vertex shader and apply them independently to each vertex. The objects are then moved to their positions relative to the camera and other objects [3]. Model transformation translate, rotate, and scale model to its position in ”the world”. View transformation move object into its position relative to camera [3]. Projection transformation perspective or orthographic projection is used to give depth. Usually all are combined into a single MVP (Projection, View, Model) matrix [19]. Rasterization maps points on model surface to pixels of the screen to create fragments [4]. Fragment shaders are also programmable and they use fragment data to calculate pixel color [19]. Multiple fragments might be considered to calculate the color of a pixel. They choose the closest fragment for opaque objects and blend several fragments for transparent objects. Rendering is the final application of colors to pixels of the screen. It also includes some logic such as blending and depth testing.

Three.js [12] is a free open source JavaScript library [21] designed for general 3D graphics programming. On top of abstracting from the low-level details of WebGL, it provides limited fall-back rendering options (regular HTML5 canvas) for platforms without WebGL support. The library organizes the rendering data and processes in object oriented way into scenes consisting of meshes (a combination of geometry and material), cameras and lights. Moreover, it provides many useful presets (materials, cameras, geometries, visual effects), particle system, animation support, data export/import and useful 3D math functions.

In our application, WebGL and three.js are used for visualization purpose, specifically for creating different shaders, materials, lighting, camera and rendering them to the scene. In the scene, we have 3D positional data represented as spheres, points, and lines.

2.2 HTML5

WebGL is not a part of HTML5 (HyperText Markup Language version 5) standard [2], but strongly relies upon it. HTML5 introduces new $\langle canvas \rangle$ element which offers two contexts: 2D providing methods and properties to draw and manipulate raster graphics, and 3D exposing WebGL API. In our application, HTML5 is basically used to create the menu options for user interaction with the system. It enables the users to manipulate the animation and trajectories of the atoms rendered. It allows the entire scene to be rendered on its canvas element.

2.3 JavaScript and JQuery

JavaScript [12] in both WebGL and Three.js is used to write the control code of the program. It is used to initialize the WebGL, create arrays to hold the data of the geometry, create buffer objects by passing the arrays as parameters, create, compile and link the shader programs, create attributes, enable them and associate them with the buffer objects, associate uniforms, create transformation matrices and many more.

JQuery [9] is free, open source and the most popular JavaScript library. It simplifies client-side JavaScript application development in several ways: advanced DOM elements selection and manipulation, improved event handling and ajax support [18], animating, etc. It is majorly used in the applicaion for sending requests to fetch data and user-interface manipulation.

2.4 Elasticsearch and JSON

Java API in our application is used to make a call from a client-side to a server-side and the data is received from the server through a HTTP protocol. It treats server objects as resources that can be created and destroyed. Elasticsearch [13] is used as a backend of our system for quick data storage, analysis and search in real time. Elasticsearch is a highly scalable open-source which is generally used as the underlying engine/technology that powers applications that have complex search features and requirements.

One of the use-cases used for elasticsearching is that we have analytical need and want to quickly analyze and visualize a lot of data. In this case, we use Elasticsearch to store our data

and then the data is pulled with appropriate queries for further processing. For instance, getting the time elapsed and trajectory length can be evaluated from the 3D scattered data representing snapshots of atomistic configurations.

The three-dimentional scattered data is stored in the form of JSON [11] which is an ubiquitous internet data interchange format. In a single data fetch, a HTTP request is sent and a single snapshot of the data is retrieved which is shown below:

```
//in each data fetch
[
  {
    "x": 0.97201533,
    "y": 0.99846838,
    "z": 0.01218933,
    "atomCode": "Mg",
    "color": "99CC00",
    "size": 12,
    "i": 0
  },
  ... //it goes till (i == size of snapshot-1)
]
```

Now, if we store the parsed JSON data in some variable, its values are easily accessible by a simple dot operator.

```
//loop goes from j = 0 to j < sizeOf snapshot
var x = data[j].x;
var y = data[j].y;
var z = data[j].z;
```

2.5 Grails Framework

Grails [20] is an open source platform, a full stack framework that uses groovy for developing java enterprise applications. It is model-view-controller (MVC) platform that is used for developing the applications which are used for handling everything from the view layer down into the persistence layer [17]. Model or data layer contains all the components that represent and manage data stored in our application. View or presentation layer consists of components that are responsible for showing to the user the state of data models, and also the actions that they can execute. Controller or control layer contains the components that receive orders from the user, manage the business logic over the data model and decide which view should be displayed. The idea behind using this architecture is that we can switch the underlying technology of any layer without affecting the other layers. For instance, as our application grew, we switched our model layer from temporary RAM storage to Elasticsearch index. Also, we added AJAX and HTML5 for building interactive application. And in these cases, all the business logic stored in the controller was not modified in anyway.

Upon each HTTP request, Grails identifies which controller it should call based on the definition set in `conf/UrlMappings.groovy`. This makes debugging and drilling down the code much easier. Following format is the default configuration of defining URIs: `[controller]/[action]/[id]`. For instance, in our system: `atom/trajectorySettings` where:

- `controller` is the first part of the name of our controller after removing the ”Controller” suffix. In our case, `AtomController` becomes `atom`.
- `action` is the method to be executed. In our case, the method is `trajectorySettings`
- `id` is the identifier of domain-class instance.

2.6 System Architecture

The system is best represented by a merge of cliet-server and MVC (model-view-controller) diagram as shown in Figure 2.2. In the client side, we have web browser. The model and controller are written in Java while the view is written in HTML5, CSS, JavaScript and JQuery. In

the server side we have model, controller, and elasticsearch service linked to the disk storage where ES index is stored.

Firstly, the raw three-dimentional positional data from the uploaded file is indexed in a document with a POST request in the elasticsearch engine. The document is then saved in the disk of the computer system in JSON format. Then, at each data fetch, a GET request is sent from client side to the document and the ES service gets the data in JSON format from the disk (one configuration block per fetch). The data passes through the controller to the view where it is rendered in the required manner.

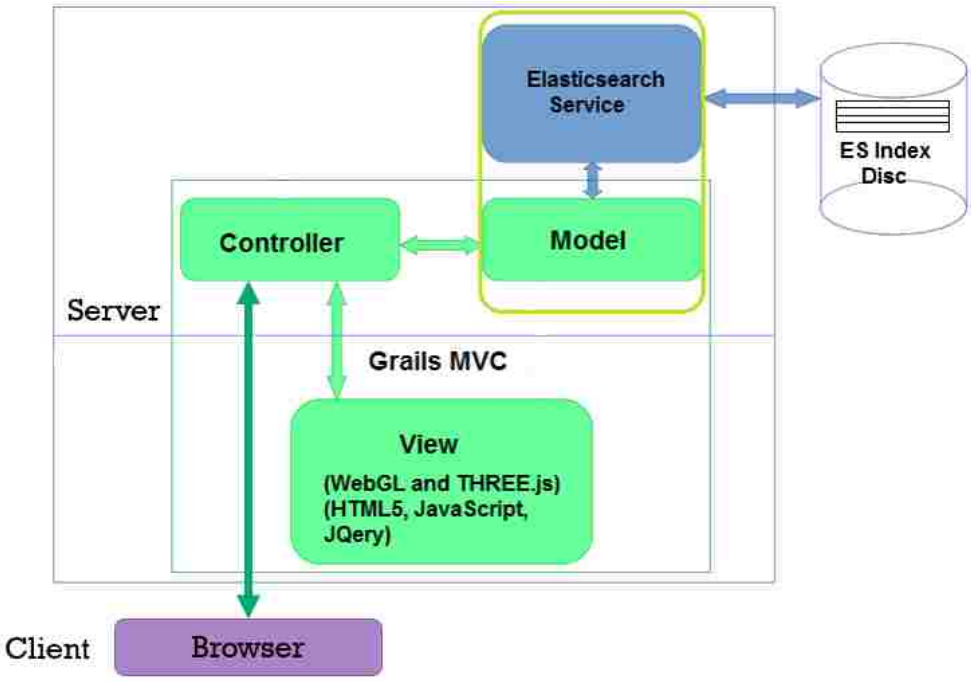

Figure 2.2: Client-server architectural diagram

Chapter 3
Molecular Dynamics Simulation Data

3.1 Background

The purpose of building the visualization system is to analyze the movements of the atoms of several species at certain temperature and pressure condition. Any atom moves from one position to another due to some force. Basically, there are two types of such forces: attractive force and repulsive force. If any two atoms are too close, they repel each other. If they are within some minimum and maximum cut-off distances, they attract each other. The force becomes zero when the two atoms are at the optimal seperation. The Newton's first law of motion is used to calculate the atomic positions at different time instants which, in turn, form the atomic trajectories. For an atom i, we have:

$$\vec{F_i} = m_i \cdot \vec{a}_i \tag{3.1}$$

$$\sum_{j \neq i, i=1}^{N} \vec{f}_{ij} = m_i \frac{d^2 \vec{r_i}}{dt^2} \tag{3.2}$$

$$dt = \Delta t \tag{3.3}$$

where m_i is the mass, $\vec{a}_i$ is the acceleration, $\vec{F_i}$ is the force on an atom i due to all other atoms j, and N is the total number of atoms. The equation is solved numerically. In our case, Δt (time step) $= 1$ fs (femtosecond) $= 10^{-15}$ seconds, and $\vec{r_i}(t) =$ positional coordinates (x,y,z) of i^{th} atom at time t.

The atoms move with the forces exerted on them due to all other atoms. Simulation is done by giving different values to temerature, pressure, time steps, etc. This resulting position-time series dataset is then used as the input of our visualization application. The complete position-

time series can be expressed as:

$$D = \{P(j\Delta t) \,|0 \leq j \leq N_{step}\} \tag{3.4}$$

where N_{step} is the number of snapshots (time steps) or configuration defined as:

$$P(t) = \{p^i_\alpha(x, y, z, t) \,|i = 1 \ldots s\} \tag{3.5}$$

where p^i_α is the position of i^{th} atom for α species at time $t = j\Delta t$ [5].

The simulated position datasets vary from several MBs to one GB for the data analyzed in the study. Loading and storing these data in a lighthanded JavaScript and HTML5 application has time and memory constraints. These data may contain many parts which are not significant, and visualizing them generates no new information. So, data need to be manipulated such that the important information is sustained while the data size is reduced considerably.

3.2 Forsterite (Mg_2SiO_4) System

Forsterite (Mg_2SiO_4) is the atomic system considered in our work. A perfect forsterite system shown in Figure 3.1 consists of 16 formula units so there are 32 Mg, 16 Si and 64 O atoms in the simulation supercell [22]. Hydrogen defect (which is also referred to as proton since it carries $+1e$ charge) is incorporated in the forsterite via interstitial or cation-vacancy sites.

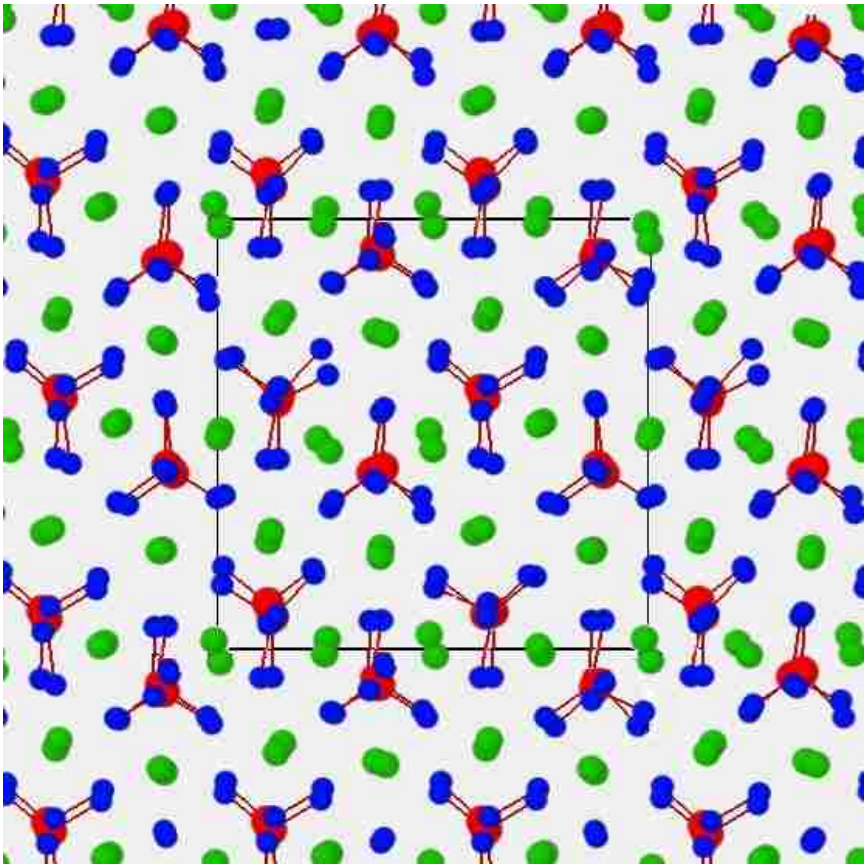

Figure 3.1: The perfect forsterite system at the ambient pressure and 2000 K. Green (medium), red (large) and blue (small) spheres represent the Mg, Si and O atoms, respectively. Each Si atom is bonded with four O atoms forming tetrahedron.

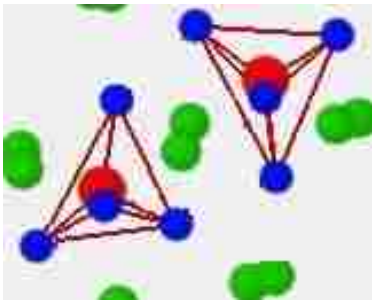

Figure 3.2: The structure of perfect forsterite crystal showing Si-O tetrahedra.

Initially, two types of vacancy defect sites were created by removing relevent ion from the simulation cell. The magnesium vacancy was created by removing the Mg core leaving its two valence electrons in the supercell and silicon vacancy defect was created by removing the Si core

leaving its four valence electrons in the supercell. The net charges in the Mg and Si vacancies are thus 2– and 4–, respectively. Then, the protonic defects were inserted either of these cationic sites. 2 protons ($2H^+$) were added to the Mg vacant site and 4 protons ($4H^+$) were added to the Si vacant site at temperature 2200 K. Similarly, the protonic defects were inserted at the interstitial sites where one hydrogen or two hydrogens were added at 2000 and 2200 K. Finally, the dynamics of these defective forsterite systems were simulated for a couple of hundreds of picoseconds (i.e., a couple of thousands of time steps), which produced the position-time series data used as input for our visualization system.

The overall mobility of atomic species can be quantified by their diffusivity values obtained from the simulated position-time series data. Table 3.1 shows only hydrogen diffusivity results for different data sets considered. Here, the filename signifies the number of hydrogen (proton), the type of defective system, and temperature of the simulation. For instance, H_I-2000 stands for one interstitial-proton system simulated at 2000 K. Similarly, $2H_I$-2200 represents the case of two interstitial protons simulated at 2200 K. $2H_{Mg}$-2200 is a system where 2 protons were added to the Mg vacant site, and, $4H_{Si}$-2200 is a system where 4 protons were added to the Si vacant site. In the Table 3.1, we see that with the increase in temperature from 2000 to 2200 K, the proton diffusivity decreases. This is abnormal behavior since the atoms are expected to diffuse faster as temperature increases. Also the diffusion rates differ significantly between different proton incorporation mechanisms. Visualization can help us find the underlying physical mechanism.

Table 3.1: Information about simulation datasets

Dataset	Original Total Time Steps	Reduced Time Steps with 10 Skipped Steps	Run Time (ps)	Hydrogen Diffusivity (10^{-9} m^2/s)
H_I-2000	126050	12505	126	12
H_I-2200	144500	14450	144.5	8
$2H_I$-2000	117000	11700	117	50
$2H_I$-2200	155270	15527	155	30
$2H_{Mg}$-2200	232080	23208	230	5
$4H_{Si}$-2200	210070	21007	210	0.5

3.3 Data Format

The primary contents of the raw position-time series data are the normalized x, y, z coordinates of the atoms as shown in Figure 3.3. The first few lines consist of metadata, representing information such as the total rime steps, the number of atoms, the simulation box dimensions (L_x, L_y and L_x), and the volume of simulation cell, the temperature condition and the simulated system name. However, the metadata always may or may not be in the same format, and sometimes they might not be present at all. So, the only content important is the 3D coordinate set.

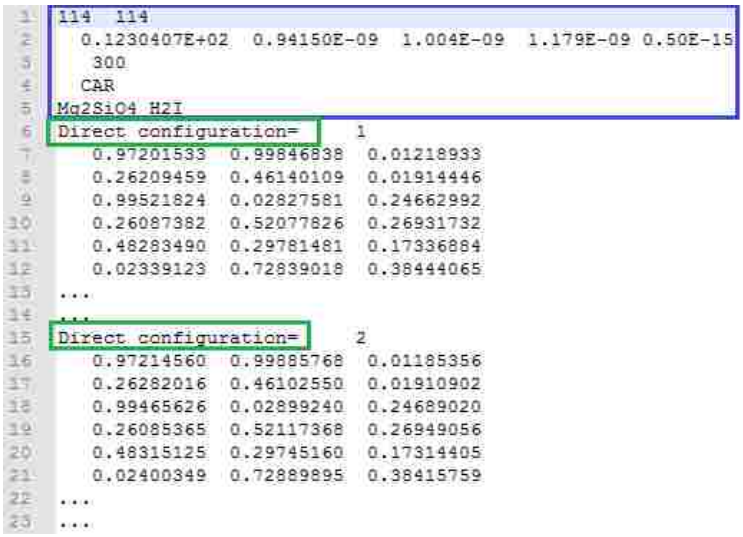

Figure 3.3: Layout of raw positions at multiple time snapshots. It has 5 lines of metadata and "Direct Configuration" denotes the beginning of a new snapshot (time step) or block.

Generally, "Direct configuration" line marks the beginning of a new snapshot (time step or block), and the following rows or lines consist of position coordinates of all atoms for that snapshot. However, in other cases the marking is just done by a different text or a blank line or not done at all. Hence, a dynamic file reading which can read files with varying contents of metadata and snapshot markings is required. Our current system has this feature.

Furthermore, the user provides additional information: species name and their ordered number in each snapshot, values for the simulation cell dimentions as a 3×3 matrix (lattice vectors),

and step manipulation values. The Figure 3.4 shows the user-interface for providing the information. The information provided can be downloaded as species file, as shown in Figure 3.5. This file can be reused for similar systems or different systems with minor edits.

Figure 3.4 shows the interstitial forsterate system with one proton. There are 32 magesium, 16 silicon, and 64 oxygen in the system. This information is submitted, and once the species file is generated, it has all the information about the species color, their starting and ending indices, chemical symbol and size, as shown in Figure 3.5.

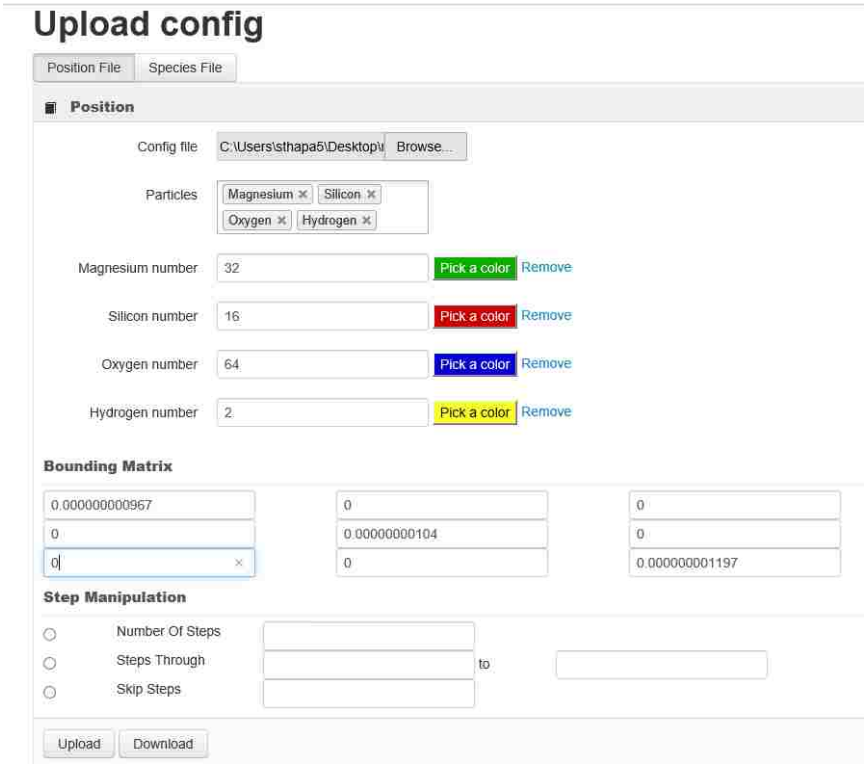

Figure 3.4: Interface where the user provides the species information.

```
!config
atoms:
   1: !atom
      colorCode: 06CC00
      end: 31
      particle: Mg
      size: 15
   2: !atom
      colorCode: FC0000
      end: 47
      particle: Si
      size: 33
      start: 32
   3: !atom
      colorCode: 000BFF
      end: 111
      particle: O
      size: 6
      start: 48
   4: !atom
      colorCode: FFFB00
      end: 113
      particle: H
      size: 1
      start: 112
end: -1
increment: 1
lastIndex: 113
matrix:
-  0.0000000009415
   0
   0
-  0
   0.000000000104
   0
-  0
   0
   0.0000000001179
start: 1
```


Figure 3.5: File format showing the species information. It is editable and reusable.

Chapter 4
Data Manipulation

4.1 Data Manipulator Application

We have created a seperate application for data manipulation (Figure 4.1). It is a simple java desktop application built on Gradle which is an open source build automation system. It builds upon the concepts of Apache Ant and Apache Maven and introduces a groovy-based domain-specific language (DSL) for declaring the project configuration. The application takes the position file and species information file as input and remodels the data based upon user's selections. Details are explained in the later sections. We have used two approaches for data reduction: uniform and non-uniform skipping.

Figure 4.1: Architecture of data manipulator application showing all components including input and output.

4.1.1 Uniform (Full-Block) Skipping

In uniform skipping we skip multiple snapshots (configuration blocks) repeatedly. The user inputs the number of snapshots they want to skip over the entire 3D time-varying data file. For instance, the user input for steps to skip is 10 and the data file has 10,000 configuration blocks or snapshots. Then, a new reduced file is created such that it writes the first snapshot, skips the next 10 snapshots (2^{nd}-11^{st}), again writes the 12^{nd} snapshot and skips the next 10 blocks and so on till

the end of file. This way we reduce the size of the file dramatically thus making data analysis and visualization faster. The scheme is only useful if all the species do not loose any important information in the skipped steps.

If n is the total number of time steps (configuration blocks) in the system and s is the number of steps to be skipped, then the total number steps in the resultant file is given by the following equation:

$$\text{Number of reduced steps, } m = n/s \tag{4.1}$$

If k is the size of atomic configuration block (that is, the total number of atoms in the simulated system), then the total number of positions decreases from nk to mk.

For the $2H_I$-2000 dataset, $n = 117000$, $k = 114$ and $s = 10$, then, $m = 117000/10 = 11700$. Initially, the total position is $nk = 117000 \times 114 = 13338000$ and finally, the reduced total becomes $mk = 11700 \times 114 = 1333800$. The total number of atomic positions decreases from 13 million to about 1.3 million.

The file size used in the study varies from few tens of megabytes to one gigabyte. Loading such big file causes $java.lang.OutOfMemoryError$ exception, i.e., a run-time memory leak. Usually, this error is thrown when there is insufficient space to allocate an object in the java heap. Table 4.1 shows the comparative size and time analysis of two of the datasets used. Both the data files, have data reduction factor of ten. The loading time decreases fifteen times.

Table 4.1: Uniform skipping analysis: showing the change in file size and loading time

System Considered: 2 proton incorporation at interstitial site of forsterite (Mg_2SiO_4)						
Dataset	Actual data size	Actual loading time	Loading time with ES	Uniform skipping steps	Final data size	Loading time after skipping
$2H_I$-2000	511 MB	25 min	5 min	10	46 MB	1.5 min
$2H_I$-2200	679 MB	30 min	7 min	10	62 MB	2 min

4.1.2 Non-Uniform (Partial-Block) Skipping

Non-uniform skipping is the second approach for data manipulation. In many positional data sets the positional information of some species or some atoms are valuable while that of some other species do not contribute much to visualization and analysis. In other words, some species or atoms in particular are highly mobile and cover large distance while others are confined to small area. The uniform skipping does not help in these scenarios because we skip data of both the mobile and immobile species or atoms uniformly. Non-uniform skipping makes it possible to skip the less-significant positional data selectively, thereby keeping only the significant ones. This can be done in per-species level or per-atomic level within each configuration block.

In the species level skipping, the system lists out the atomic species (in our case: Mg, Si, O and H) and the user can select the species to skip. The application then creates a new file with the complete positional data of first snapshot and that of the unskipped species for the remaining many snapshots (until the next full snapshot is encountered). The resultant file is then exported. The file is greately reduced in terms of size with negligible information loss.

In the atomic level skipping, the user has the flexibility of skipping any range of atomic positional data, ranging from 1 to k where k is the total size of a snapshot (configuration block). The chosen range is removed from all the snapshots except the initial and selected snapshots that are kept unchanged. Thus, several particular immobile atomic positions can be removed irrespective of their entire species.

Skipping is not done at the configuration block level so the total number of configurations (time steps) remain the same. The number of the steps where skipping is done at atomic or species level is denoted by s, n is the total original time steps, $m = n/s$ is the reduced number of steps where no skipping is applied (full configuration block), j is the number of atoms that are removed in every configuration block where skipping is applied.

In the case of the species-level skipping, j is the total number of atoms of one or more species selected for skipping. The total number of atomic positions is reduced from nk to $mk + (n - m)(k - j)$. If the skipping is applied to all configurations except one selected configuration

(which is used for rendering the background atomic structure), m = 1 and $k - j$ = 1 to 4 protons if only atomic positions of proton defects are kept by removing all other species.

For the $2H_I$-2000 dataset, $n = 117000$, $k = 114$. Let, m = 1 and $k - j$ = 4 for four protons in the system. Initially, the total position is $nk = 117000 \times 114 = 13338000$ and finally, the reduced total becomes $mk + (n - m)(k - j) = 469100$. The total number of atomic positions decrease from 13 million to about 470 thousand. For the case when $m = n/s = 10000$, the total number of reduced protons is $mk + (n - m)(k - j) = 10000 \times 114 + (117000 - 10000) \times 4 = 1568000$ or about 1.57 million.

We have to consider the following points for data manipulation:

- In uniform skip, the loading time and file size depend upon the steps skipped while in species level non-uniform skip, they depend upon the species skipped and finally in the atomic level non-uniform skip, they depend upon the number of atoms skipped.

- The complete analysis of all the aforementioned datasets is done in Chapter 6. From the analysis, we see that only hydrogen atom shows considerable movement while all other atoms have negligible displacement. In such cases, non-uniform skipping will be more helpful where we can keep the positions of only the mobile atoms and skip the remaining.

- In the Table 4.2, the uniform skipped step is 10 and the non-uniform skipped species are Si, Mg, and O. Here, we can see that the size and time are greately reduced.

Table 4.2: Non-Uniform skipping analysis: showing the change in file size and loading time

System Considered: 2 proton incorporation at interstitial site of forsterite (Mg_2SiO_4)						
Dataset	Actual data size	Actual loading time	Uniformly skipped data size	Uniform Loading time	Non-uniformly skipped data size	Non-uniform Loading time
$2H_I$-2000	511 MB	25 min	46 MB	1.5 min	12 MB	0.2 min
$2H_I$-2200	679 MB	30 min	62 MB	2.0 min	16 MB	0.3 min

Chapter 5
Data Rendering

The atomic position datasets must be explored interactively in time and space in order to extract meaningful information. Interactive visualization helps in exploring the data by mapping the data into computer images.

5.1 Material System's Structure

The structure of a three-dimentional positional dataset is rendered as spheres centered at the provided positions of constituent atoms. The size of the sphere can be made proportional to the effective ionic radius. For example, hydrogen is showed by the smallest sphere.

5.1.1 Ball Representation

The sphere has three attributes: position, size, and color. The user can update the color and size of the sphere with a single click and the changes are seen on the canvas. A snapshot consists of positions of all the species in the material system at a certain time step. When we loop through the snapshots, we get the animation of all atoms thereby representing the visualization of the atomic movements during the simulation (Figure 5.1).

```
geometry = new THREE.BufferGeometry();
positions = new Float32Array(particles * 3 * 26);
colors = new Float32Array(particles * 3 * 26);
var sizes = new Float32Array(particles * 3 * 26);
 geometry.addAttribute('position', new
    THREE.BufferAttribute(positions, 3));
geometry.addAttribute('color', new THREE.BufferAttribute(colors,
   3));
geometry.addAttribute('size', new THREE.BufferAttribute(sizes, 3));
geometry.computeBoundingSphere();
```

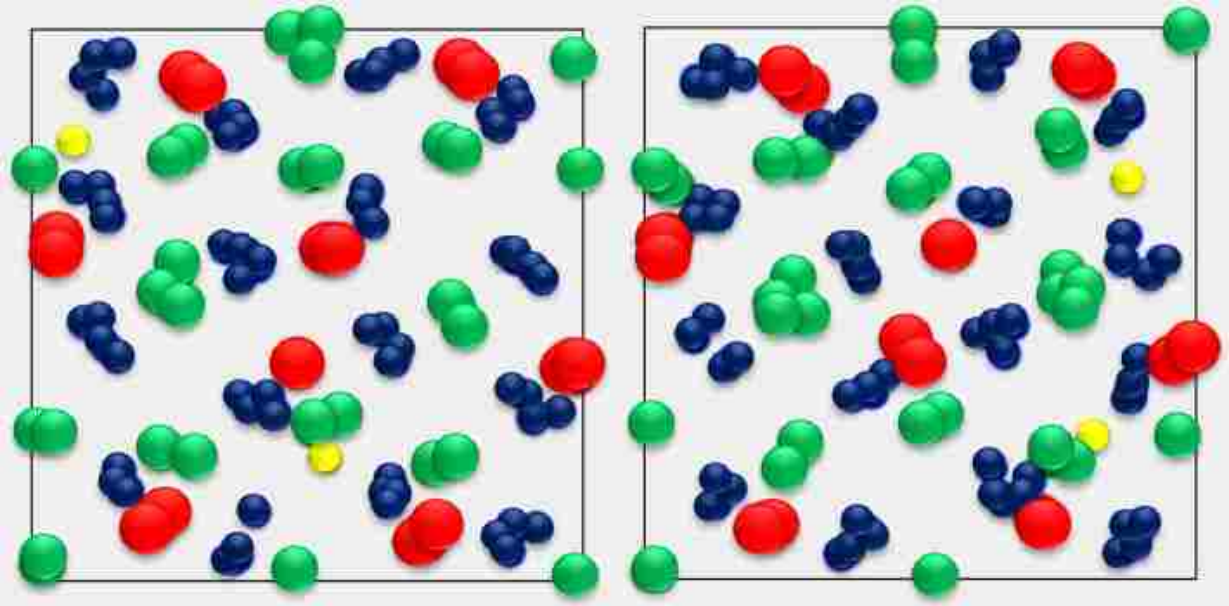

Figure 5.1: Initial (left) and 100th (right) snapshots showing atomic structure of 2H$_I$-forsterite system. The atomic species are shown by spheres with their respective species colors (Mg: green, Si: red, O: blue, and H: yellow). There are 32 Mg, 16 Si, 64 O and 2H in the system.

We can manipulate the size and color of atoms at both species and structural level by providing the species name and index number, respectively. We can also manipulate the steps by updating the start and end steps as needed, and also by increasing the step increment value whose default value is set to unity. We can zoom in and out the resultant visualization display and can also set the view from the x, y and z-directions.

5.1.2 Ball and Stick Representation

Any two atoms can come closer to each other only till a certain minimum distance which is guided by the strong repulsive force that gets dominant at smaller distances [5]. A cutoff-min distance is the minimum allowable distance between any two atoms and a cutoff-max is the maximum distance within which two atoms are associated (e.g., bonded) with each other. These distances vary according to the types of atoms considered. The distribution of other atoms around each atom lying within a cutoff space window defines the coordination environment [5]. In a multi-species system, the user selects any pair of species, and bonds (not necessarily chemical bonds) are extracted and displayed between these species based on the cut-off distance window

provided. We also count the number of bonds (or connections) which represents the coordination number.

In our visualization system, the user provides the minimum and maximum cutoff between any two species (the same or different) and the species themselves. In Figure 5.2, we set the cutoff minimum and maximum as 0.0 and 0.18, respectively. Ball and stick representations define the local tetrahedra coordination environment between Si and O.

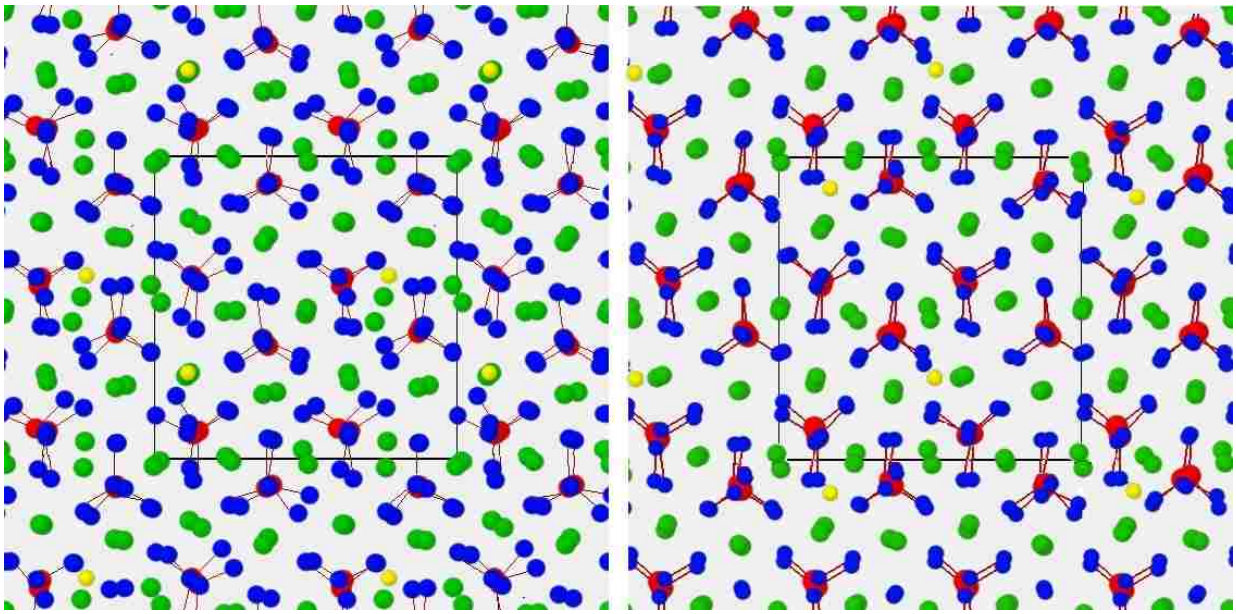

Figure 5.2: Coordination of Si with respect to O in the $2H_I$-forsterite system at initial and 100^{th} snapshots.

5.1.3 Periodic Boundary Conditions

Periodic boundary conditions [5] means that a simulation cell is being replicated throughout the space, i.e., in all three major dimentions in such a way that an infinite (arbitarily extended) system is being formed. In other words, the dynamics of the atomic motions in the central cell is replicated in all other cells. The number of atoms in the simulation cell remains constant. When an atom leaves the simulation cell, it enters the one the periodic images. This means that the atom enters the simulation region from the opposite face as shown in Figure 5.3.

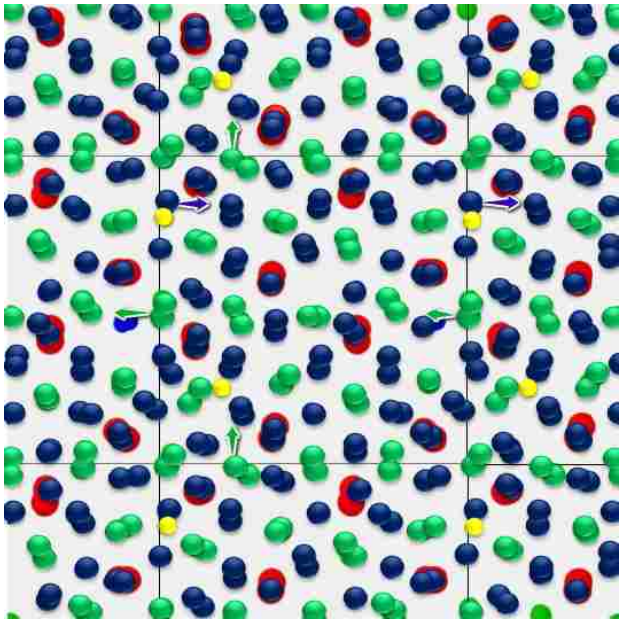

Figure 5.3: Schematic illustration of periodic boundary conditions in the simulation cell.

5.2 Material System's Dynamics

In order to understand how and to what extent the constituent atoms in a given material system move, we consider two approaches: 1) animating atomic positions, and 2) displaying complete trajectories of the atoms.

The set of positions an atom occupies over the time defines the trajectory of that atom. In a solid system the trajectories of different atoms do not overlap because each atom remains within a well-defined region around itself whereas in a liquid system the trajectories are dispersed and overlapping because atoms move far from their mean positions. Forsterite system is a defect-bearing solid system. The defect atoms (like hydrogen) and nearby host atoms can move significantly showing extended trajectories. Trajectories can be of two types: folded trajectories and unfolded trajectories.

We draw trajectories in two ways: drawing trajectories step by step in an animated matter; and alternatively, storing all the positional data of the entire configurations and displaying the trajectory at once at the end. The animated trajectory rendering is preferable for smaller datasets (few tens of MBs), because one can view the dynamical behavior at every time step. However,

it might be time consuming for larger datasets for which one-time rendering is preferable. The additional features of the incremental rendering include the options of changing the color of individual trajectories such that tracking the motion of individual atom is possible, applying different color codes in the trajectories for getting the information of their total length, and the time elapsed. Here, the color codes merge with the individual color of the trajectories such that we can differentiate the atoms as well as get the desired information simulatenously.

5.2.1 Folded Trajectories

As discussed earlier, when an atom leaves the simulation cell, it enters the cell from the opposite face. Thus, the atom leaves the cell. This behavior is shown by the folded trajectories which are always confined within the simulation cell. The entire path of the atom is shown as a series of points which are discontinuous as they leave and enter from opposite wall. Figure 5.4 (left) shows folded trajectories of the system $16(Mg_2SiO_4) + H_2$ for its two hydrogen atoms, represented by yellow and purple colors. Mg, Si and O trajectories are denoted by green, red, and blue colors, respectively. We can see that protons travel long distances while Mg, Si and O atoms are confined in small areas. The trajectories do not get out of the box, thereby representing the periodic boundary conditions imposed in the simulations.

5.2.2 Unfolded Trajectories

Trajectories that are unfolded become continuous curves extending in the space both inside and outside the simulation cell. They show the actual nature of motion of any atom more effectively. The input simulation data sets represent the folded data, which are converted to the unfolded data by the following code.

```
//in each data fetch i.e. each snapshot
$(data).each(function (i, row) {
       //getting the folded data
         var x = row.x;
         var y = row.y;
```

```
        var z = row.z;
        if (isUnfolded(row, i) && _DATA != undefined) {
          //calculating the difference
           var x1 = x - _DATA[i].dX;
           var y1 = y - _DATA[i].dY;
           var z1 = z - _DATA[i].dZ;
           if (x1 < -0.5) {
              x = _DATA[i].dX + (x1 + 1);
           } else if (x1 > 0.5) {
              x = _DATA[i].dX + (x1 - 1);
           }
           if (y1 < -0.5) {
              y = _DATA[i].dY + (y1 + 1);
           } else if (y1 > 0.5) {
              y = _DATA[i].dY + (y1 - 1);
           }
           if (z1 < -0.5) {
              z = _DATA[i].dZ + (z1 + 1);
           } else if (z1 > 0.5) {
              z = _DATA[i].dZ + (z1 - 1);
           }
           //storing the new unfolded data
           data[i].dX = x;
           data[i].dY = y;
           data[i].dZ = z;
        }
}
```

Figure 5.4 (right) shows unfolded trajectories of the forsterite system with two protonic defects. We see that the hydrogen atoms colored yellow and purple travel more distances than other Mg, Si and O atoms colored green and red and blue, respectively. However, trajectories of magnesium and oxygen also travel outside the box but only cover small distances.

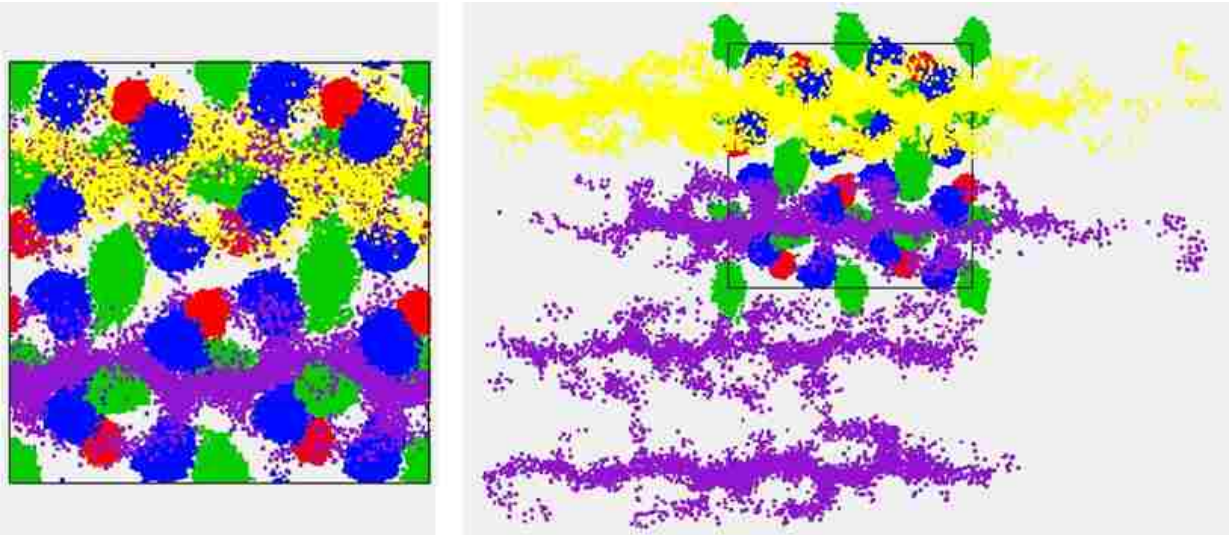

Figure 5.4: Folded trajctory (left) and unfolded trajectory (right) of the atoms in the $2H_i$-forsterate system drawn as series of points with their respective species colors (Mg: green, Si: red, O: blue, and H: yellow and purple).

5.3 Color Coding of Information

A variety of information about the structural and dynamical behavior of the simulated material system can be coded along the particle-trajectories. For instance, the color of the pixel representing each position can be varied according to distance travelled, elapsed time by using appropriate color maps. Figure 5.5 shows the color coded trajectories for displaying the atomic distances calculated with respect to the time elapsed.

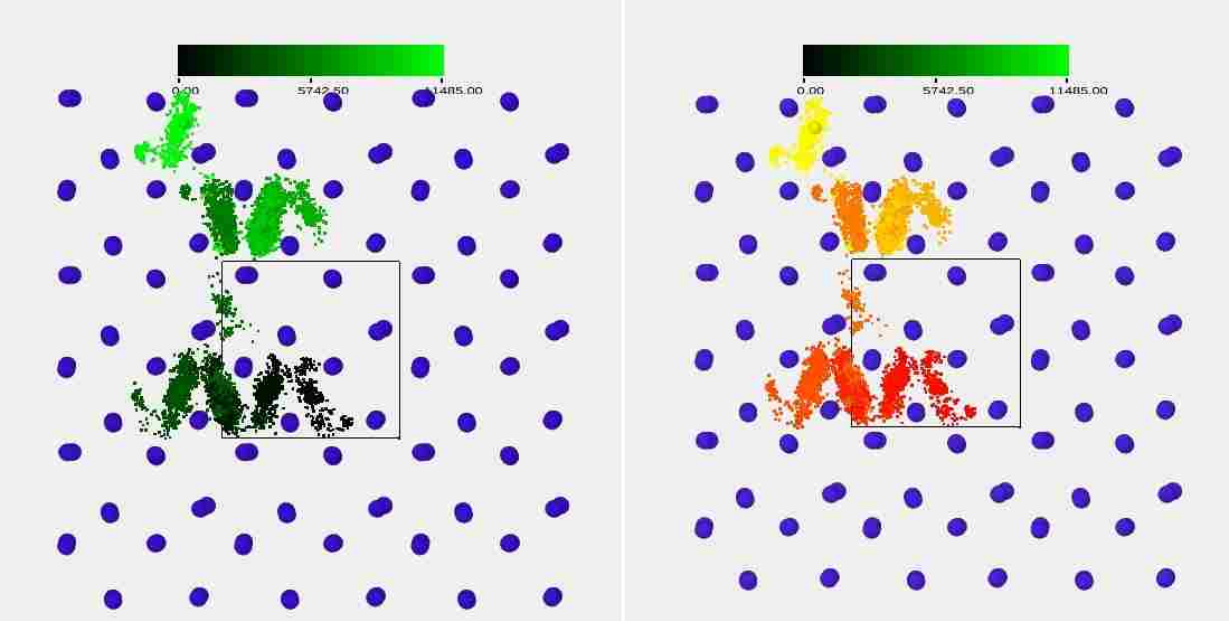

Figure 5.5: Color coded trajectories of all atoms showing the distance information with a black-to-green color map (left). The color-map is blended with the species color (Red) in the right figure.

5.4 Superimposition

While trajectories do not discard any information, the uncorrelated and highly entangled trajectories are difficult to interpret. Further enhancements are required for the information gain from the visualization. Features like color encoding and changing the color of individual trajectories can be done simultaneously. Superimposition is the combination of atomic structuress and trajectories. There are three options for the superimposition: default, user-defined and view-selection superimposition.

5.4.1 Default Superimposition

In default superimposition, the current position of the atoms are rendered as spheres while all positions (from the time steps) along the trajectory are rendered as points. It is easier to track the location and path followed by each atomic trajectory. Figure 5.6 shows the default superimposition where the small spheres are the contemporary positions of the atoms.

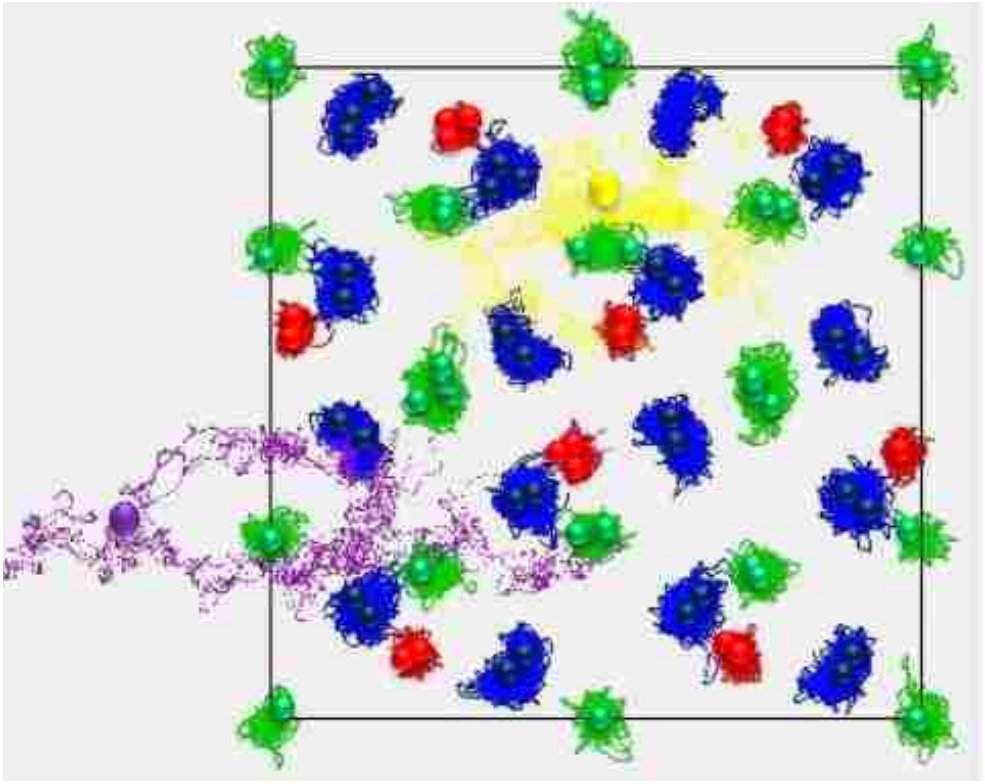

Figure 5.6: Default superimposition: the current position of the species is denoted by the sphere followed by the tail like trajectories denoting the path followed. (Mg: green, Si: red, O: blue, and H: yellow and purple).

5.4.2 User-Defined Superimposition

In the user-defined superimposition, the user provides the index number, which is the configuration number, and its unique position within the configuration. The corresponding coordinates are then rendered as spheres. The user can select multiple configuration numbers for the same or different trajectories individually. They have the flexibility of selecting size and color of the spheres, such that it is easier to locate the positions. This allows us to visualize the positions of the selected atom at different snapshots (Figure 5.7). In the figure, we see that there are three trajectories with unique configuration number or index targetted. They have specific color and size.

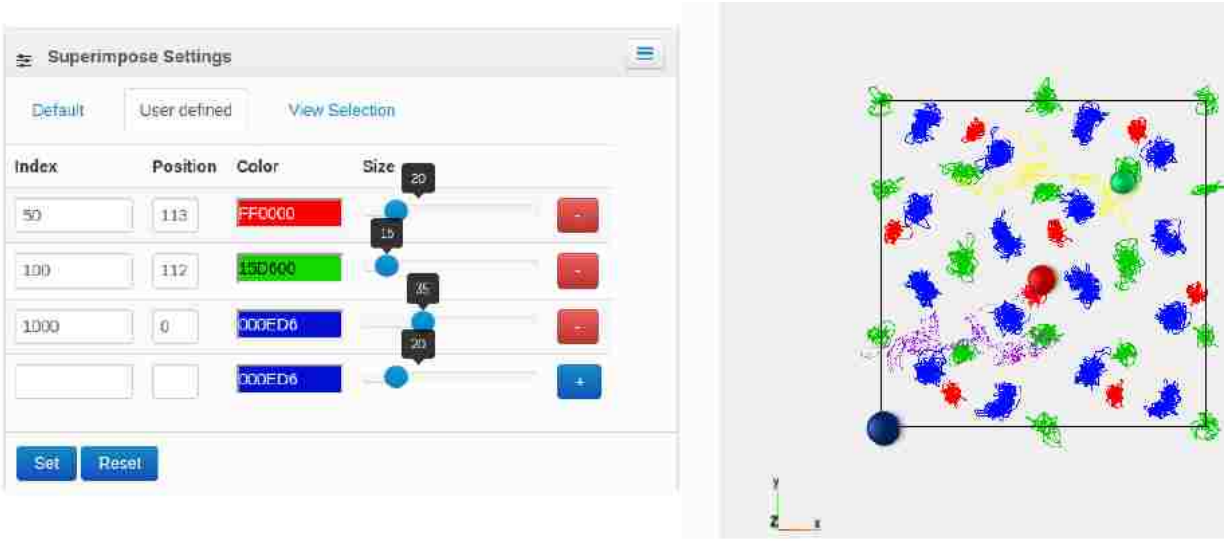

Figure 5.7: User-defined superimposition: the index and position of the atom is chosen in order to render that specific position as sphere (left) and the rest is rendered as trajectories.

5.4.3 View-Selection Superimposition

In this type of superimposition, the users are able to view animation of the selected atoms and trajectories of the remaining atoms in the same canvas. The animation takes place using the folded data while the trajectories are drawn with the unfolded data. There is a continuous conversion from folded to unfolded data, which in turn increases the disk access time and processing time. Furthermore, the species or individual atoms selected to be rendered as sphere can be replicated in all the major axes and they follow periodic boundary conditions. View selection superimposition is further divided into two types: species level (Figure 5.8) and atomic level (Figure 5.9).

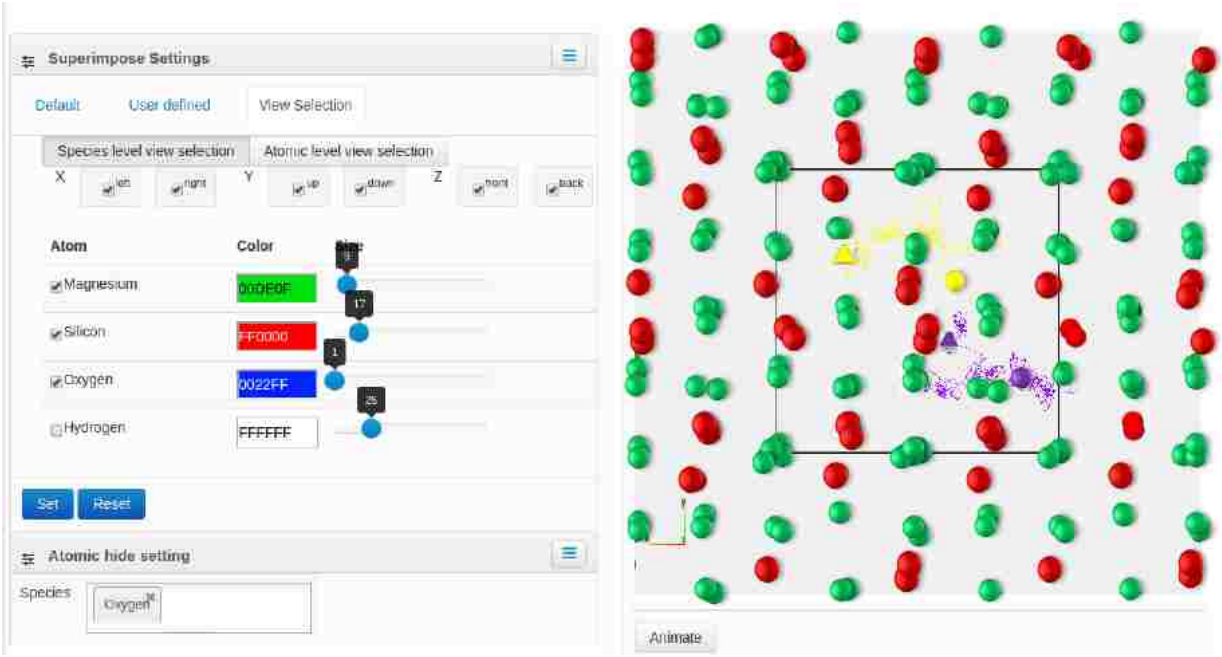

Figure 5.8: Species level view selection superimposition: Mg and Si are selected to be rendered as spheres, while H is rendered as trajectory.

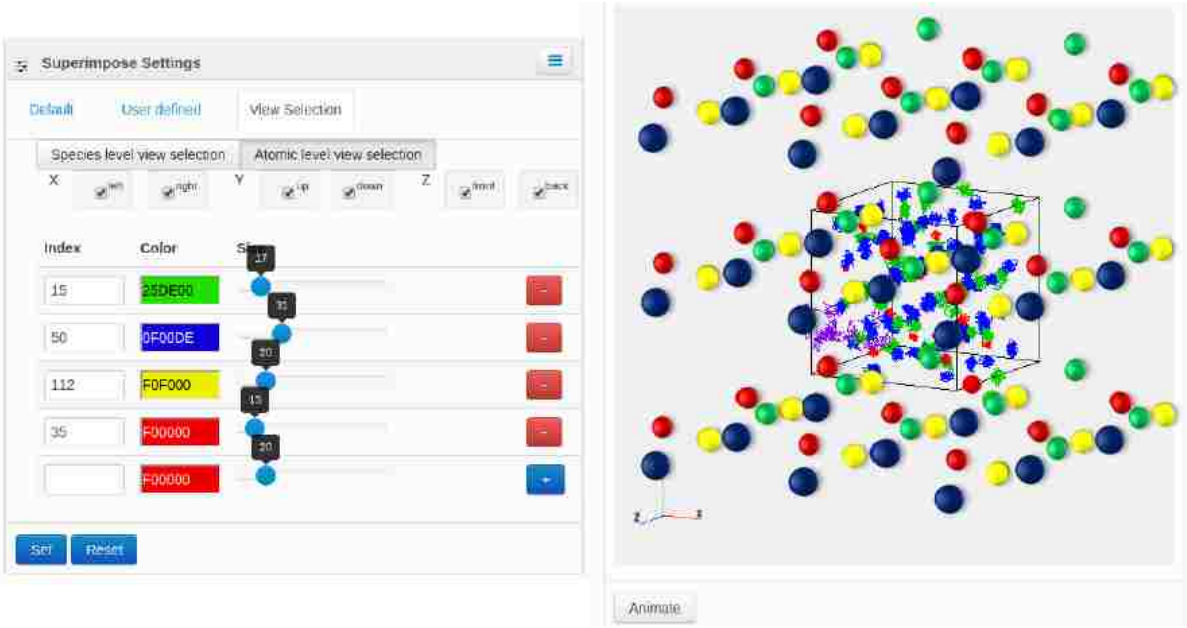

Figure 5.9: Atomic level view selection superimposition: three perticular atomic indices are selected to be rendered as spheres and the remaining ones are rendered as trajectories.

In Figure 5.8, we see that all Mg, Si and O atoms are rendered as spheres, while the H atoms are rendered as trajectories. We have also selected replication along the y- and z- axis. Similarly, in Figure 5.9, the user has selected three atoms or indices to be rendered as spheres and also selected replication in all three axes. The initial and final positions of the trajectories are rendered.

5.5 System's Time Analysis

Our system follows a cycle (Figure 5.10) where it fetches data block by block (configuration by configuration) with the HTTP GET request to the ES server. The response is received in a JSON format which is further parsed to a list. The list consists of the 3D positional coordinates of each atom in that snapshot. These data are saved in a vertex buffer and added to the scene. The scene is finally rendered and the process continues for the next block till the last block (or any user-defined termination block). Now, each step takes a finite processing time. Clock-functions are set to evaluate the data fetching and rendering time for finite range of configurations in different defective fosterite systems. We could find the total time taken for the entire dataset as well as per configuration (to fetch or to render). During the entire process, a constant CPU utilization of 12% was observed.

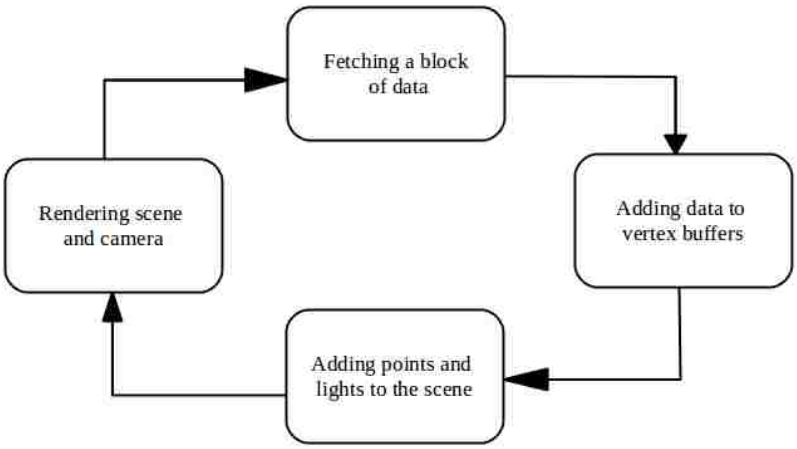

Figure 5.10: Data fetching, buffering and rendering cycle which continues till the last configuration (block) or till a user defined termination step.

5.5.1 Data Fetch Analysis

Table 5.1 shows that the time taken to fetch the multiples of 4000 configurations for the $2H_{Mg}$–forsterite system having the positional data file of size 91 MB. The rate of fetching each block in every bundle of 4000 configurations (tile steps) is almost constant. In every data fetch, a packet of about 11 KB was fetched at the average speed of 9.3 ms.

Table 5.1: Data fetching timing for the $2H_{Mg}$ -forsterite system.

Total configurations (steps)	Time (sec)	Time per step (sec)
4000	39	0.0098
8000	75	0.0094
12000	109	0.0091
16000	144	0.0090
20000	179	0.0090

5.5.2 Analysis of Animated Structures

The average rendering time (which also includes the data fetching time) per configuration (time step) is 0.06 sec, which gives an interactive frame rate. Table 5.2 shows that the time taken to animate atoms in multiples of 4000 configurations. The rate of rendering each block (configuration) in every bundle of 4000 configurations is almost constant. This analysis was performed for $2H_I$ and $2H_{Mg}$–forsterite systems having the positional data file of size 62 and 91 MB, respectively.

Table 5.2: Data rendering timing for the $2H_I$ and $2H_{Mg}$ -forsterite systems.

Total configurations (steps)	Time (sec)	Time per step (sec)
System: $2H_I$-forsterite system		
4000	240	0.060
8000	488	0.061
12000	726	0.061
System: $2H_{Mg}$-forsterite system		
4000	244	0.061
8000	482	0.060
12000	724	0.060
16000	969	0.061
20000	1202	0.060

5.5.3 Analysis of Trajectories

Trajectory rendering is done in two ways as illustrated in Figure 5.11. In the static trajectory, the data blocks are fetched, buffered, and added to the scene till the last step (or any user-defined termination step), one step per cycle. The rendering of the complete trajectories is done once at the end. In animated trajectory, rendering is done multiple times within a single cycle (every time a data block is fetched and added to the buffer). Such animation is suitable for smaller dataset where the user can view and manipulate the properties of the trajectories in real time and visualize the progressive motion at each time-step.

The total time taken to render the complete trajectory for $2H_{Mg}$ -forsterite system was found to be 2.93 hours (Table 5.3) and 0.15 hours (Table 5.4) with the animated and static rendering, respectively. Thus, the trajectory rendering time is reduced by a factor of 20 from the animated to static rendering. Table 5.3 shows that the total rendering time of animated trajectories increases linearly with the number of configurations (n). This is because multiple renderings are done at each time step in animated trajectories. In other words, in first configuration, rendering is done one time and in second configuration rendering is done two times (current buffer + previous buffer), and so on. But in static trajectory, single rendering is done at a constant average time of 0.03 sec (Table 5.4).

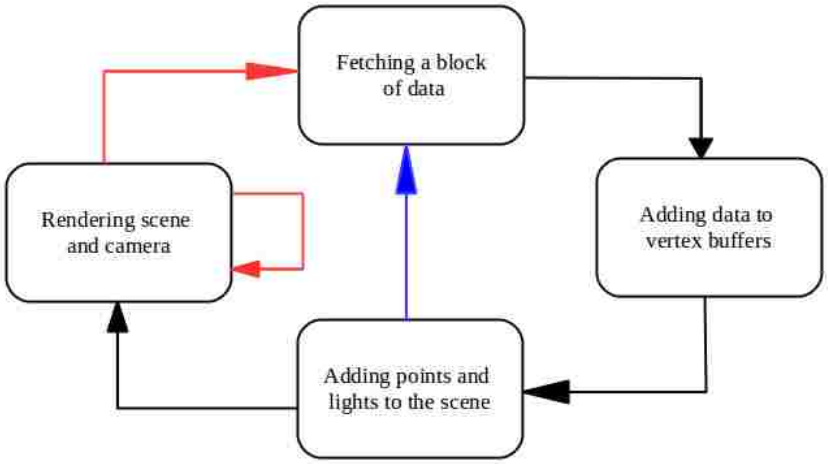

Figure 5.11: Data fetching, buffering and rendering cycle (black arrow for both animated and static trajectories; blue arrow for static trajectory only; and red arrow for animated trajectory only).

Table 5.3: Animated trajectory rendering timing for the $2H_I$ and $2H_{Mg}$ -forsterite systems.

Total configurations (steps)	Time (sec)	Time per step (sec)
System: $2H_I$-forsterite system		
4000	660	0.165
8000	2091	0.261
12000	4491	0.374
System: $2H_{Mg}$-forsterite system		
4000	420	0.105
8000	1620	0.200
12000	3900	0.325
16000	6960	0.435
20000	10560	0.528

Table 5.4: Static trajectory rendering timing for the $2H_{Mg}$-forsterite system.

Total configurations (steps)	Time (sec)	Time per step (sec)
System: $2H_{Mg}$-forsterite system		
4000	106	0.027
8000	212	0.027
12000	321	0.027
16000	435	0.027
20000	549	0.027
System: $2H_{Mg}$-forsterite system		
40000	1243	0.031
80000	2501	0.031
120000	3819	0.032
160000	5054	0.032
200000	6260	0.031

Chapter 6
Visualization Application Case Studies

Visualization-based analysis of forsterite (Mg_2SiO_4) system with one and two protonic defects at the interstitial site is done at 2000 K and 2200 K using the first-principles MD simulation datasets: H_I-2000, H_I-2200, $2H_I$-2000, and $2H_I$-2200. Table 3.1 shows that with temperature increase from 2000 to 2200 K, the hydrogen (proton) diffusivity value decreases. This is an abnormal case because the atoms are expected to diffuse faster at higher temperature. There may be different scenarios. First, the mobility of proton (hydrogen ion) is high but the net distance covered is small, which may be due to the back and forth motions that cancel eachother. Second, the atoms do not move at the same rate throughout the simulation process. Sometimes they move fast and sometimes they move slow. Here, we consider finite simulation runs (e.g., a couple of hundred picoseconds) so there is a possibility of capturing slow dynamic regime.

The atomic movement in the forsterite system with the cation vaccancy protons is analyzed at 2200 K. Table 3.1 shows that the hydrogen diffusivity in vacant magnesium site ($2H_{Mg}$-2200) with two protons is larger than that in vacant silicon site with four protons ($4H_{Si}$-2200). It is important to understand how these protons (individually or collectively) move in and out of their site of incorporation.

Our purpose is to understand the dynamical behavior of the protons by examining the geometries and extents of their trajectories. We make quantitative and qualitative analysis of the proton movements. In quantitative analysis, we calculate the net distance covered by the protons between their original and final positions. Qualitative analysis involves the examination of the movement patterns along the three different directions.

6.1 Quantitative Analysis

The 3D distance formula is used to calculate the distance between the initial and final positions of corresponding atoms. However, our input spatio-temporal data are normalized and are in the range [0, 1]. These position coordinates should be multiplied by the simulation cell dimen-

sions (lattice vectors) for correct distance calculation. These cell dimensions (L_x, L_y, and L_z) are recorded in the species file along with other information. The distance formula is:

$$d = \sqrt{(x_i - x_f)^2 \cdot L_x^2 + (y_i - y_f)^2 \cdot L_y^2 + (z_i - z_f)^2 \cdot L_z^2} \tag{6.1}$$

where (x_i, y_i, z_i) is the initial position and (x_f, y_f, z_f) is the final position for the atom under consideration. Note that the positions are unfolded (i.e., the periodic boundary conditions are removed).

Table 6.1 shows the net distances covered by the the interstitial and vacant cationic protons. We have $L_x = 9.415$ $\mathring{A}$, $L_y = 10.04$ $\mathring{A}$ and $L_z = 11.79$ $\mathring{A}$, where 1 $\mathring{A}$ = 10^{-10} m. We can see that the single interstitial proton covers less distance at 2200 K than at 2000 K. For two interstitial protons, H1 travels much more at 2200 K compared to 2000 K. The second proton H2 covers similar distances at both temperatures. At the Mg-vacant site, H2 travels much more than H1. At the Si-vacant site, two protons H2 and H3 cover significant distances. It appears that only H2 is caplable of escaping the vacant sites in both the cases. Our system can calculate the net distances the atoms have covered at any time instant with respect to their initial positions. Furthermore, visualization will helps us find the details of atomic movements to explain the net distance results.

Table 6.1: Net distance calculations for various protonic defects in the simulated forsterite system.

Filename	Proton label	Initial position	Final position	d ($\mathring{A}$)
H_I-2000	H	(0.4168, 0.3902, 0.9893)	(-0.9913, 1.7373, 1.2428)	19.2
H_I-2200	H	(0.5210, 0.5527, 0.4724)	(-0.3005, -0.3642, 0.7589)	12.5
$2H_I$-2000	H1	(0.7623, 0.7122, 0.7768)	(0.6086, 0.7791, 0.7692)	1.6
	H2	(0.4168, 0.3902, 0.9893)	(0.3614, 0.6808, 0.3182)	8.5
$2H_I$-2200	H1	(0.0749, 0.7967, 0.0681)	(0.6695, 0.3033, 0.8037)	11.5
	H2	(0.5341, 0.2170, 0.7536)	(0.3347, 0.7263, 0.2854)	7.8
$2H_{Mg}$-2200	H1	(0.0848, 0.9110, 0.0262)	(0.1239, 0.9513, 0.2614)	2.8
	H2	(0.8926, 0.0743, 0.9772)	(0.0546, 0.1437, 0.2770)	11.9
$4H_{Si}$-2200	H1	(0.4660, 0.5249, 0.3759)	(0.45367, 0.3883, 0.2994)	1.7
	H2	(0.3280, 0.1971, 0.5054)	(1.4824, 0.54208, 1.4221)	15.7
	H3	(0.4754, 0.3796, 0.2869)	(0.4304, 0.2036, 0.7801)	6.1
	H4	(0.4967, 0.3149, 0.4323)	(0.4355, 0.3102, 0.4864)	0.9

6.2 Qualitative Analysis

Our qualitative analysis focuses on the movement patterns followed by the proton defects at the interstitial and cationic Mg and Si vacant sites. We visualize the atomic positional data by merging animation (i.e., rendering atomic configuration at each time step going through successive time steps) with trajectories (i.e., allowing complete representation by rendering positions of all atoms at all-time steps as points). Additional information can be also coded along the particle trajectories. Then, we view the results from all three major axes (x-, y-, and, z-direction) and analyze their patterns. The number of simulation cell dimension distances travelled and the number of back and forth motions of the protons along each direction are also extracted.

6.2.1 Dynamics of One Interstitial Proton

The H_I-forsterite system $16(Mg_2SiO_4)$ + H_1 was simulated at two temperatures: 2000 and 2200 K. We call them as system 1 and system 2. Our general expectation would be to observe more protonic movement in system 2 than in system 1 because of its high temperature. However, Table 6.1 implies that the case is opposite. Figures 6.1 and 6.2 help us finding the underlying reasons.

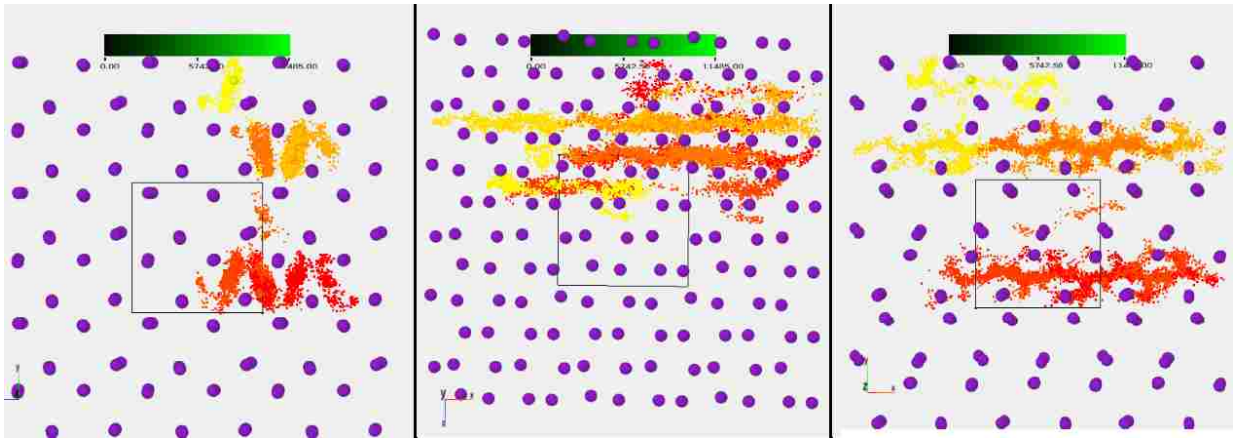

Figure 6.1: Views of H_I-forsterite system from the $x-$, $y-$, and $z-$ directions at 2000 K. Time elapsed is encoded with black-to-green colormap. Si atoms are shown by the purple spheres (Mg and O atoms are not shown for the shake of clarity).

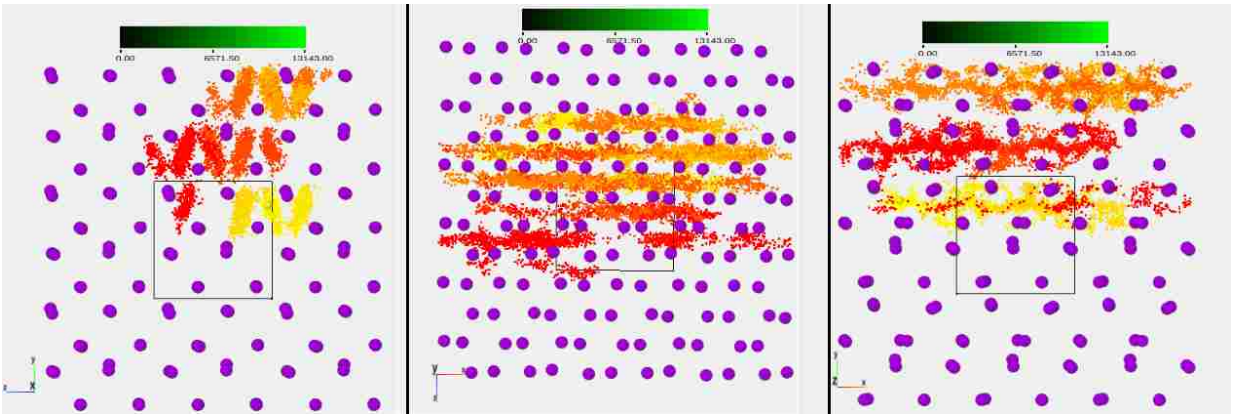

Figure 6.2: Views of H_I-forsterite system from the $x-$, $y-$, and $z-$ directions at 2200 K. Time elapsed is encoded with black-to-green colormap. Si atoms are shown by the purple spheres (Mg and O atoms are not shown for the shake of clarity).

In Figures 6.1 and 6.2, we see a color coded trajectory of H1 atom in system 1 and system 2, respectively viewed from the $x-$, $y-$ and $z-$ directions. The varying color of the pixels represents each position based on the time elapsed while travelling through time step 0 to time step 11500 (system 1) or 13143 (system 2). The initial color of the hydrogen trajectory is red, and the red color merges with the black to green color band, thus converting to yellow towards the end. We can extract the qualitative information by visualizing system 1 and 2 as follows:

- Almost linear continuous motion occurs along the x-direction. Denser layers in system 2 than in system 1 illustrates that there are more back and forth motions of the protons during the course of simulation.

- Large hops between different layers occur along the y-direction covering four layers in system 1 and three layers in system 2. Thus, the net distance travelled along the y-direction is larger than in system 1.

- Clear zig-zag motion occurs along the z-direction. The net motion along the z-direction is small.

- Several reverse motions can be seen, as shown by the mixture of bluish and greenish pixel colors, i.e., no gradual change in color observed. The protons move back and forth thus

cancelling the net distance travelled. Such motions are more frequent in system 2 than in system 1. This explains the quantitative difference.

6.2.2 Dynamics of Two Interstitial Protons

Let us consider $2H_I$-2000 and $2H_I$-2200 as system 3 and system 4, each having two protons: H1 and H2. Figures 6.3 and 6.4 help us explaining the difference in the net protonic movements at the two temperatures.

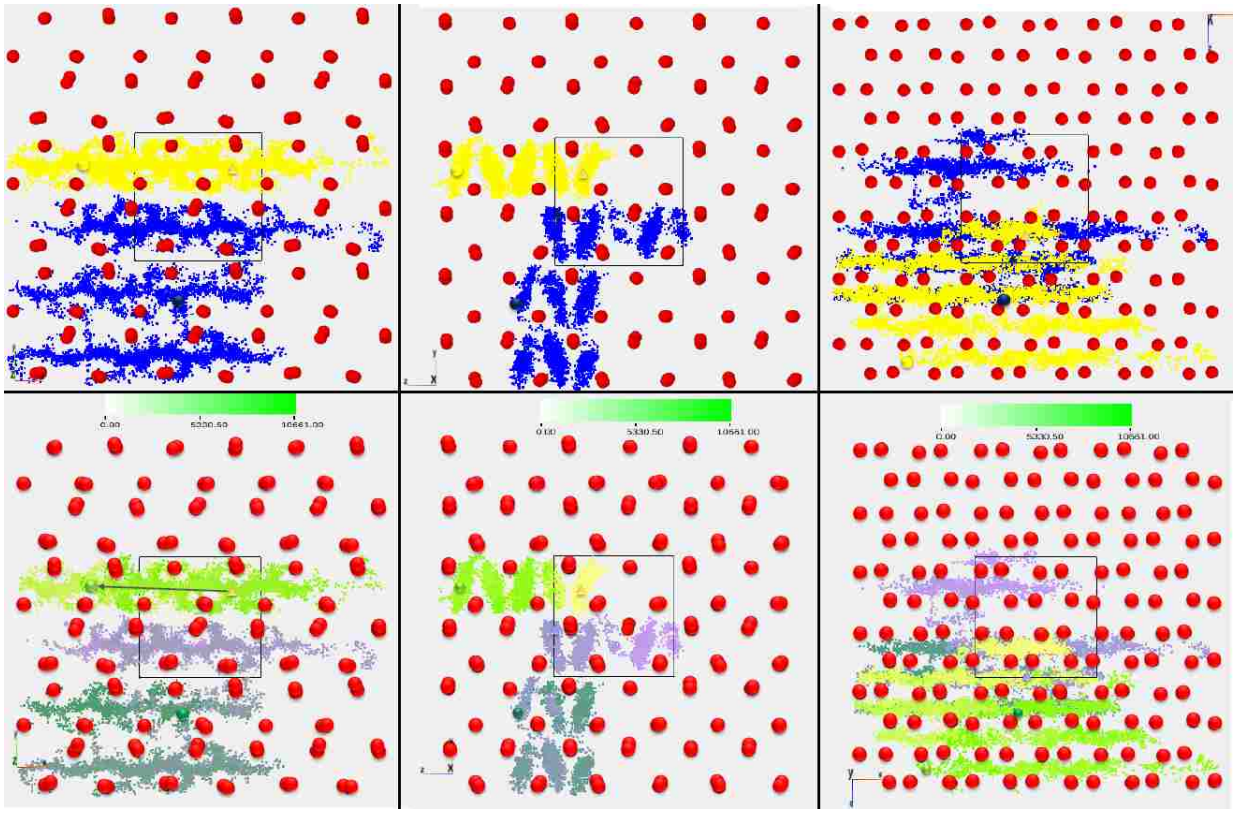

Figure 6.3: Views of $2H_I$-forsterite system from the $x-$, $y-$, and $z-$ directions at 2000 K. The upper three images represent the trajectories based on species color (H1: yellow, and H2 blue) and the lower three images show the time elapsed information encoded with a white-to-green colormap. Si atoms are denoted by the red spheres.

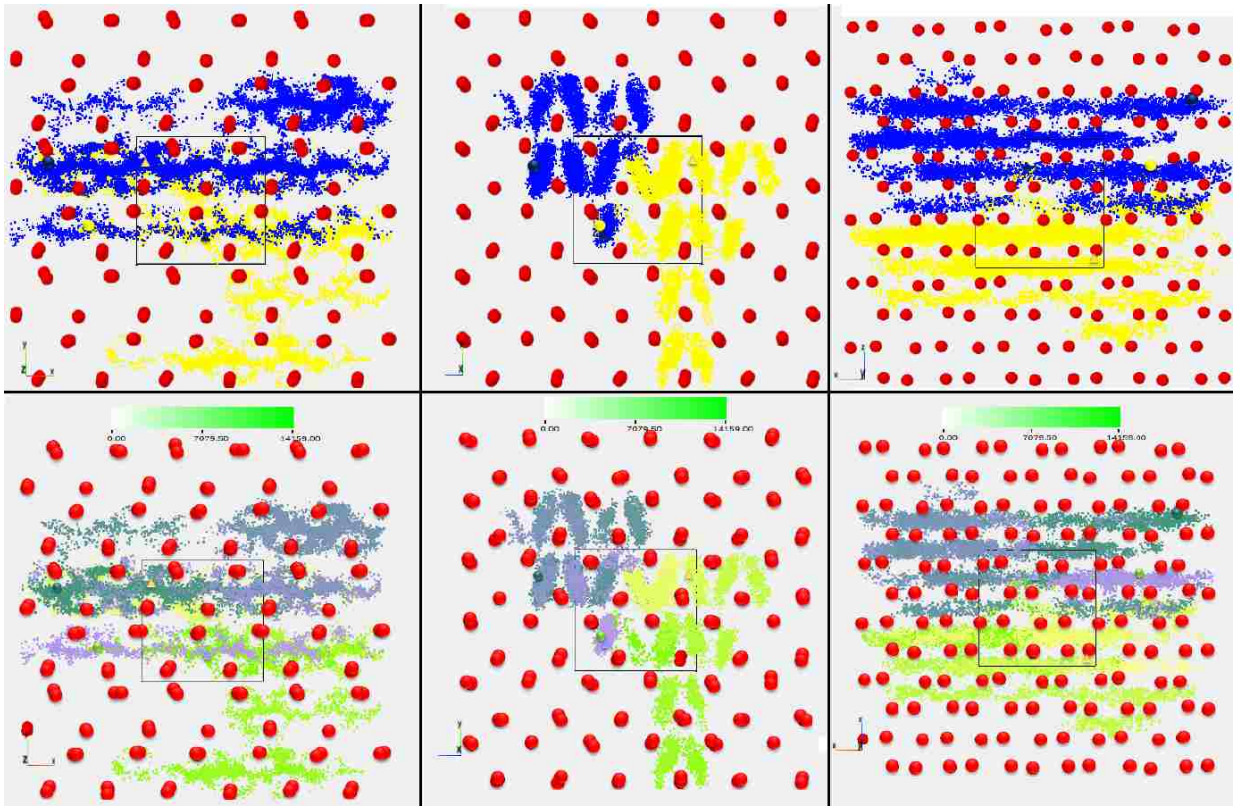

Figure 6.4: Views of $2H_I$-forsterite system from the $x-$, $y-$, and $z-$ directions at 2200 K. Trajectories based on species color (H1: yellow, and H2: blue) (upper three images) and time elapsed (lower three images) are rendered.

The qualitative information we can obtain from system 3 and system 4 is listed below:

- There is a difference in the motion of H1 and H2. H1 is less mobile than H2 in system 3 whereas H1 is more mobile than H2 in system 4. This supports the quantitative difference shown in Table 6.1.

- In system 3, the motion of H1 is mostly along the $x-$ and $z-$ directions while H2 moves in all three directions. In system 4 both hydrogens (H1 and H2) cover almost the same area and have similar pattern of motion in all three directions.

- In system 3, there is linear motion with small hops from one layer to another along the $x-$ direction. H1 has a single layer movement while H2 has multi-layer movement. The denser parts illustrate that there are more repetitions of path (reverse motion of the protons)

during the course of simulation. In system 4, the pattern followed is the same with H1 and H2 having net movements of almost one simulation distance.

- In system 3, the motion along the $y-$ direction is negligible for H1, while the motion is half a simulation distance for H2. The total motion of H2 is almost three times the total motion of H1 along the $y-$ direction. In system 4, the motion along y direction is negligible for both protons. The initial and final positions are almost co–linear. H2 travels in 3 distinct layers while H1 travels in 4 distinct layers.

- In system 3, the net z-directional motion is seen more in H1 than in H2. Both the protons move more than one block (one simulation cell) along z-axis. The motion is zig-zag. In system 4, the pattern of motion is same. The gross motion is more than one block (one simulation cell), however, the net motion between the initial and final position is half a blong along the z-direction.

6.2.3 Dynamics of Two Protons Incorporated at Vacant Mg Site

We visualized the dynamical behavior of two protons that replace one of the Mg atoms in the Mg_2SiO_4 forsterite at 2200 K. These protons are denoted as H1 and H2 with blue and yellow colors initially assigned. These initial colors merge with the white-to-green colormap for time information, and turn gradually into bluish green and yellowish green, respectively. The initial and final positions of H1 are closer than those of H2 when viewed from the $x-$ and y-direction. Also, the protons spend a lot of time at a certain region (pocket) about the Mg-site, and somehow escapes from the pocket (one at a time). They again get trapped into similar pocket (a periodic image of the original Mg-vacant site).

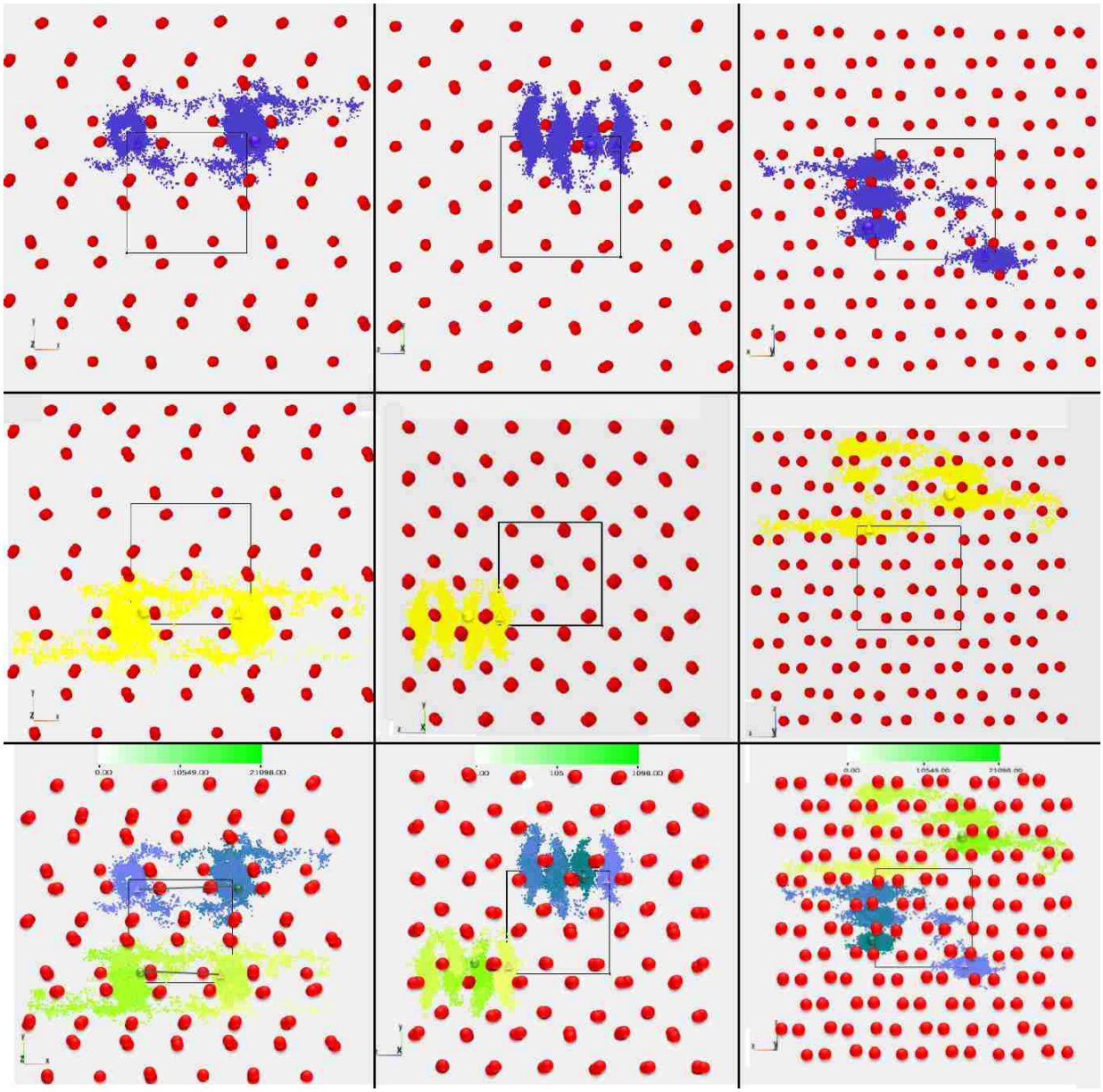

Figure 6.5: Trajectories of protons incorporated at vacant Mg site viewed along the $x-$, $y-$, and z-directions at 2200 K. The top three images show the blue colored trajectories of H1 atom, the middle three images show the yellow trajectories of H2 atom. The bottom three images show the time encoded trajectories of both the atoms.

The qualitative information we can obtain from the system is listed below:

- The probability of finding the protons is high near the vacant Mg-site. One of the protons eventually escapes from the pocket to the interstitial region. The other proton also escapes from the pocket when the first one re-enters the posket.

- The net movements of protons in the x-direction are almost equal to one simulation length. However, their total movements are more than 2 simulation lengths. The reverse motion cancel each other.

- A lot of back and forth motion occurs in the z-direction and the net distance is small. The protons never jump one simulation cell length.

- The protons move a simulation length with back and forth motion along the y-direction. The net motion remains small.

6.2.4 Dynamics of Four Protons Incorporated at Vacant Si Site

We visualized the movements of four protons that replace one of the Si atoms in the forsterite system at 2200 K. These protons are denoted as H1, H2, H3, and H4 with their initial colors shown in Figure 6.6.

Figure 6.6: Initial colors selected for four protons.

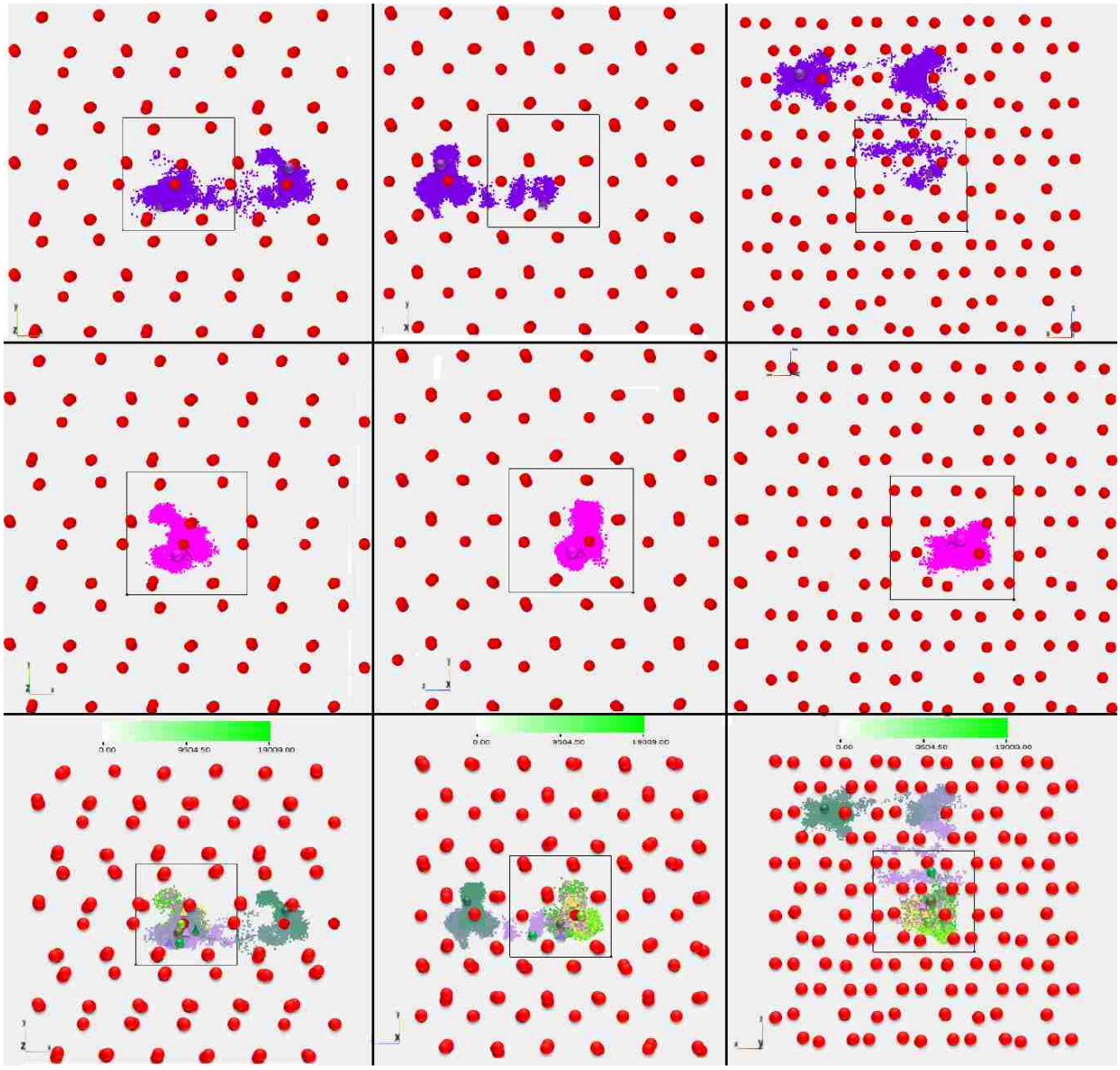

Figure 6.7: Trajectories of protons incorporated at vacant Si site viewed along the $x-$, $y-$, and $z-$directions. The top three images show the purple colored trajectory of most mobile H2 atom, the middle three images show the pink colored trajectory of least mobile H4 atom. The bottom three images show the time encoded trajectories of all four atoms.

As shown in Figure 6.7, one proton (H2) make the movement of more than one simulation length. The other three protons (H1, H3 and H4) show the short back-and-forth movements and remain confined in the local neighborhood around the vacant Si site. Only H2 manages to escape from the vacant site to interstitial region, covering large distance which is much larger than the net distances covered by other three protons (Table 6.1).

The qualitative information we can obtain from the system is listed below:

- The probability of finding the protons near the vacant site is higher so they spend most of their time in this local neighborhood or pocket. At most, only one proton gets to escape from the vacant site.
- The net movement of H2 in the x-direction is slightly more than one simulation length but the net movements of H2, H3 and H4 along the x-direction are much smaller.
- The H2 atom which manages to escape from the pocket covers four or more full simulation cell lengths.
- The escaped proton (H2) travels less than a half simulation length along the y-direction. It covers one ore more full simulation length along the z-direction. A lot of back-and-forth motion occurs for these directions.
- The pattern of movement of the escaped protons in the non-vacant areas closely resembles that of the interstitial protons.

Chapter 7
Conclusions and Future Work

This thesis presents a web-based data visualization system for atomic position-time series obtained from materials simulations. It can process spatio-temporal datasets of different sizes by providing the flexibility to manipulate, render, and analyze the data. For large data set, the preprocessing step of data reduction is performed by identifying unimportant (less important) portions of the data. For instance, in the forsterite (Mg_2SiO_4) system considered in this study, protons (hydrogen ions) are highly mobile in comparison to all other atoms. Discarding large fractions of the positional data of the immobile atomic species reduces the data size considerably for further processing without any critical information loss. The data manipulator allows us to skip any range of atomic configurations uniformly or non-uniformly.

Our web-based visualization system (Figure 7.1) processes the data by creating a JSON structure, which can be queried quickly using Elasticsearch. Atomic configuration corresponding to a time step is visualized by rendering the atoms as spheres with an option of showing inter-atomic bonds/connections. The atomic structure can be animated by rendering all configurations sequentially. On other hand, the trajectories are obtained by rendering atoms as points for all configurations (time steps) together. The system allows us to manipulate the atomic trajectories at the individual level so that the dynamics of each particle can be examined in detail. The trajectories can also encode the additional information such as the time elapsed and the distance travelled using appropriate color maps. The atomic structure and trajectories are superimposed so that one can explore the movements of mobile atoms in relation to the three-dimensional arrangement formed by less mobile atoms.

We used our newly implemented web-based visualization system to explore the dynamical behavior of protons (hydrogen ions) in the forsterite (Mg_2SiO_4) system – an abundant mineral of Earths upper mantle. The simulation position-time data sets used are for the three types of

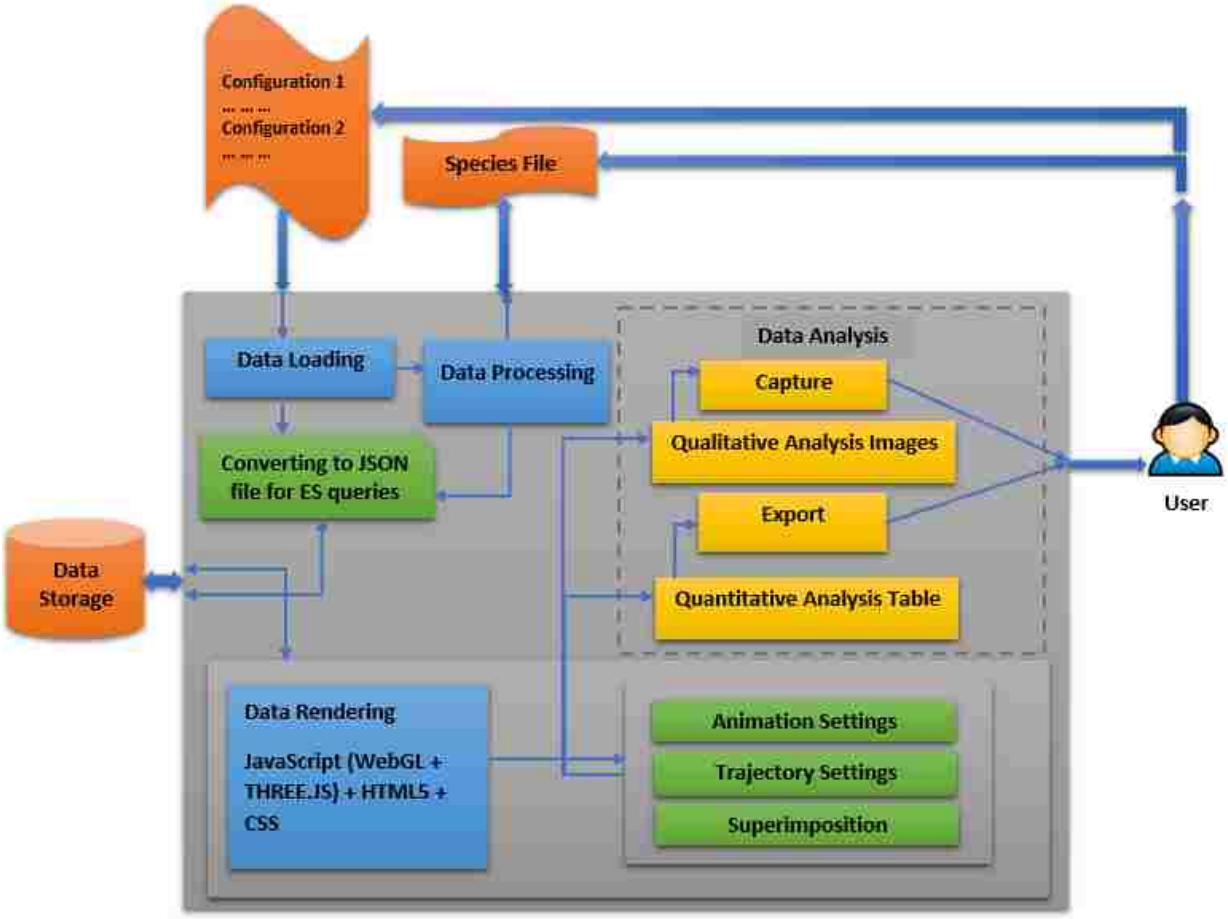

Figure 7.1: Architecture of the web-based visualization system showing all components including input and output.

protonic defects: a) one or two protons incorporated at interstitial site, b) two protons substituted Mg site, and c) four protons substituted Si site at temperatures 2000 and 2200 K. The detailed visualization reveals several interesting features:

- The interstitial protons are much more mobile than the cation-vacancy protons (those incorporated at Mg or Si vacant site).
- The protons show anisotropic movement with high mobility along the x-direction. The motion is limited and occurs in jumps along the y- and z-directions.
- Motion reverses its direction at full simulation box length-scale or even at longer distances. Because of such back and forth movements, the net distance covered by proton (considering the initial and final positions) can vary considerably within the finite durations of simulation. The net distance travelled at higher temperature sometimes can be smaller than that at lower temperature as we found in the case of the interstitial protons between 2200 and 2000 K.
- The Mg vacancy protons appear to be more mobile and show more extended trajectories than than the Si vacancy protons. In each case, only one proton manages to eventually escape from the vacant site while other proton(s) remain confined in the local neighborhood. This means that only one proton per vacant site is actually contributing to the proton diffusion/conduction process.

Our work represents a simplistic (direct) web-based rendering of the simulated position-time series data with the main focus on the analysis of the dynamical behavior of protonic defects in materials. While its usefulness has been demonstrated for a real material problem, the proposed system needs improvement on several fronts. The rendering quality should be improved so that the 3D geometries and extents of the trajectories and also 3D atomic structures can be assessed more effectively and interactively. Our time-efficiency analysis shows that the system can load the positon-time data of size up to 1 GB in few minutes. The atomic structure can be animated

smoothly with interactive frame rates achievable. However, the trajectory processing is slow. Depending on the size of the data, the total processing time varies from several seconds to several minutes, and even to an hour. We need to speed up the trajectory processing and to also enhance the capability to handle larger datasets than those considered in the present study. More useful information can be extracted and rendered on the fly to aid visualization. These improvements will help us in gaining useful structural and dynamical information from more materials simulation data which are widely generated.

References

[1] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids*, Oxford university press, 1989.

[2] G. Anthes, "HTML5 leads a web revolution,"*Communications of the ACM*, vol. 55, no. 7, 2012, pp. 1617.

[3] M. Anttonen, A. Salminen, T. Mikkonen, and A. Taivalsaari, "Transforming the web into a real application platform: new technologies, emerging trends and missing pieces,"*Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 2011, pp. 800807.

[4] A. Anyuru, *Professional WebGL programming: developing 3D graphics for the Web*, John Wiley & Sons, 2012.

[5] D. Bhattarai and B. B. Karki, "Atomistic visualization: spacetime multiresolution integration of data analysis and rendering,"*Journal of Molecular Graphics and Mod- elling*, vol. 27, no. 8, 2009, pp. 951968.

[6] B. Bohara and B. B. Karki, "Rendering particle trajectories with color-coded infor- mation for atomistic visualization,"*Proceedings of the IASTED International Con- ference on Visualization, Image and Image Processing (VIIP 2012)*, 2012, pp. 3542.

[7] B. Bohara and B. B. Karki, "Clutter reduction in rendering of particle (atom) tra- jectories with adaptive position merging,"*International Conference on Computer Graphics Theory and Applications (GRAPP 2014)*, 2014, pp. 265271.

[8] X. Carreras, I. Chao, L. Padr, and M. Padr, "FreeLing: an open-source suite of language analyzers.,"*LREC*, 2004, pp. 239242.

[9] J. Chaffer and K. Swedberg, *Learning jQuery*, Packt Publishing Ltd, 2011.

[10] J. Congote, A. Segura, L. Kabongo, A. Moreno, J. Posada, and O. Ruiz, "Interactive visualization of volumetric data with webgl in real-time,"*Proceedings of the 16th International Conference on 3D Web Technology*. ACM, 2011, pp. 137146.

[11] D. Crockford, "The application/json media type for javascript object notation (json),"2006.

[12] J. Dirksen, *Learning Three. jsthe JavaScript 3D Library for WebGL*, Packt Publish- ing Ltd, 2015.

[13] C. Gormley and Z. Tong, *Elasticsearch: The Definitive Guide*, "OReilly Media, Inc.", 2015.

[14] A. Herraez, "Biomolecules in the computer: Jmol to the rescue,"*Biochemistry and Molecular Biology Education*, vol. 34, no. 4, 2006, pp. 255261.

[15] A. Hildebrandt, A. K. Dehof, A. Rurainski, A. Bertsch, M. Schumann, N. C. Tou- ssaint, A. Moll, D. Stckel, S. Nickels, S. C. Mueller, et al., "BALL-biochemical algorithms library 1.3,"*BMC bioinformatics*, vol. 11, no. 1, 2010, p. 531.

[16] W. Humphrey, A. Dalke, and K. Schulten, "VMD: visual molecular dynamics,"*Jour- nal of molecular graphics*, 1996.

[17] C. M. Judd, J. F. Nusairat, and J. Shingler, *Beginning Groovy and Grails*, Springer, 2008.

[18] M. D. Network, "Ajax,"2006.

[19] T. Parisi, *WebGL: up and running,* "OReilly Media, Inc.", 2012.

[20] G. Rocher, P. Ledbrook, M. Palmer, J. Brown, L. Daley, B. Beckwith, and L. Hotari, "The Grails Framework-Reference Documentation,"*Environments,* vol. 3, 2009, p. 3.

[21] M. Sundstrm, "Developing a Dynamic Web Based 3D Visualization Tool,"2013.

[22] A. K. Verma and B. B. Karki, "Ab initio investigations of native and protonic point defects in Mg 2 SiO 4 polymorphs under high pressure,"*Earth and Planetary Science Letters*, vol. 285, no. 1, 2009, pp. 140149.

Vita

Simron Thapa was born on June, 1991, in Biratnagar, Nepal. She holds a Bachelor in Computer Engineering from Tribhuvan University, Nepal. Thereafter, she worked in Deerwalk, Nepal as a software engineer. In the fall of 2015, she was accepted to the graduate program in computer science at Louisiana State University. She worked as a research assistant to Dr. Bijaya B. Karki while working toward the masters degree in computer science. Her primary interests include machine learning and scientific visualization.