Fast Time-Reversible Algorithm for Rotational Motion and Its Applications to Molecular Simulation of Quasi-Liquid Layers on Ice-Ih

（高速な時間反転対称型回転アルゴリズムと、その氷の疑似液体層シミュレーションへの適用）

by
Yasuhiro Kajima

Submitted to
the Department of Scientific and Engineering Simulation
in partial fulfillment of the requirements for the degree of
Doctor of Engineering
at
Nagoya Institute of Technology

2014

Abstract

The importance of atomistic simulations which treat atoms and molecules explicitly is increasing in various fields such as nanotechnology, atmosphere, and development of devices. The computational power of supercomputers is rapidly increasing, and simultaneously, supercomputers are strongly expected to fully demonstrate their computational power to perform such atomistic simulation both on larger systems and for longer period than ever before. Recent supercomputers with many nodes and cores enable us to perform simulation on large systems by dividing objects. On the other hand, high speed intramolecular vibration of molecules becomes an obstacle to take longer time step, and there is no fundamental solution to overcome the obstacle up to now. However, we can take a longer time step in simulation by assuming each molecule as a rigid body.

This rigid body molecular dynamics (MD) simulation method ignores intramolecular vibration/deformation which is the inevitable nature of molecules. However, we can take even a 5 fs time step by considering molecules as rigid body; while, on the contrary, if we take into account the intramolecular motion, we can take a 1 fs at best in such cases as water, where molecules contain light atoms as hydrogen. The rigid body approximation method enables us to perform MD simulation with long time step for the cases where intramolecular vibration/deformation is negligible.

Therefore, many numerical algorithms to describe rotational motion of rigid body have been devised, however, I think that those algorithms are still insufficient for long time MD simulation. To see this, let me see some of those algorithms. The angular momentum Verlet (AMV) algorithm proposed by M. Hiyama et al. a few years ago is very simple and the computation cost is small, however, I find the total energy increases monotonously in many cases. The symplectic algorithm shows high stability, however, the algorithm is complicated and requires long computation time. Gear's predictor-corrector algorithm is very accurate for a short time step. However, it is not time-reversible, and becomes unstable or the total energy increases significantly in a long-time simulation run.

So, I firstly proposed an algorithm, named numerically exact time-reversible (NET) algorithm. The algorithm is time-reversible, stable, and simpler than others. However, it requires an iteration procedure to get the angular velocity, which results in arbitrariness in estimating accuracy. Therefore, I have proposed two kinds of simple iteration-free algorithms by advancing the NET algorithm, which I call Fast Time-reversible (FT) algorithms. The FT algorithms are probably the simplest (shortest in the length of codes) compared with other algorithms for rotational motion, however, their stability is no less than that of symplectic algorithm. They are ones of the simplest algorithms and in fact, the computation timings of the FT algorithms are the minima in existing algorithms in several CPU architectures. Hence I claim that the FT algorithms are the most suitable ones for the computation of rotational motion.

As an application of the FT algorithms, I have devised an parallelized computer code for rigid body molecular dynamics, by adopting MPI libraries and the fast-multipole method (FMM) for the computation of the Coulomb forces. Employing this code, I have performed large-scale simulations of quasi-liquid layers [i.e., liquidlike layer on the surface of ice at temperatures below the bulk melting point, (QLLs)] of ices-Ih on several supercomputers. The ices are without dislocation or with screw dislocation, and each of which is put in a vacuum box.

Here, to prepare configurations of hydrogen atoms in ice-Ih with dislocation, I adopted my original method. The largest dimension of the ices used in our simulation is 0.06 micrometer. I regard each H_2O molecule as a rigid body, and employ the TIP4P intermolecular interaction potential with an established reputation. Simulations are performed at three temperatures $T_{\rm m} - 23$, $T_{\rm m} - 13$, and $T_{\rm m} - 1$ K where $T_{\rm m}$ is the melting point of the TIP4P bulk-ice.

As the results of our simulation, I have found the following: (a) The QLLs are bumpy with the bump heights as large as a few inter-bilayer distances of ice (9 to 12 Å), and the widths of trenches surrounding the liquid bumps reach 100 Å. The liquid bumps fluctuate to form and break at a variety of places in a random manner. (b) At relatively lower temperatures,

the molecules in the second bilayer from the outside are partly melted and they melt easily when they are located under the trenches. Furthermore, by the change of the locations of liquid bumps over time, the melted areas newly covered by thick bumps recrystallize. (c) At a temperature slightly below the melting point, the second bilayer under the liquid bumps melts entirely to form a liquid sheet, and the bumps sit on it. (d) Microscopic properties (i.e., mean squared displacements and O-O distances) of the liquid bumps differ from those of the liquid sheet.

The present results (a) and (b) will offer a novel picture of surface melting of sub-micrometer-scale ices, and indicate that the recrystallization will make environmental hydrophilic (acidic) substances dissolve to be incorporated substantially in the layer even if the QLL is thin.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the present thesis, I present two results: One is the fast and time-reversible algorithms for rotational motion that are perhaps ones of the fastest compared with other existing ones, and the other is detailed analyses of structures of quasi-liquid layers (QLLs) on ices-Ih through molecular dynamics (MD) simulation performed with a computer program adopting one of the FT algorithms. I also present an algorithm that I have used to make configurations of water molecules of ices-Ih, since it is based on different method than that in preceding papers.

The first part of this thesis is devoted to developing algorithms for rotational motion, which I call Fast Time-reversible (FT) algorithms (Chapter 2), the second part to an algorithm for making initial configurations of ice-Ih (Chapter 3 and Appendix A), and the last part to the simulation of quasi-liquid layers of ices-Ih (Chapter 4). In addition, in Appendix B, we recall some notion of quaternion which will be helpful to understand the description of rotation used in preceding chapters, and by which I show some results concerning to FT algorithm. In appendix C, I show a modified version of FT algorithm, which has advantage as well as disadvantage compared with original FT algorithm.

Chapters are mutually related to each other, however they are logically independent. Thus I describe the motivation and aim of each chapter at its introduction. In this introduction, I give motivations throughout the thesis and a brief sketch of the three parts.

1.1 Motivation

It is well known that the surface of ice melts below the bulk melting temperature [1]. The properties of the melted layers are similar to liquid, however they are not quite the same as liquid. Thus we call the layers quasi-liquid layers (QLLs). The QLLs of ices are thought to play very important roles in many aspects, e.g., in the skating mechanics while neither pressure melting nor frictional heating can explain slipperiness [2], and in the acid snow formation by dissolving acidic substances in the QLL [3]. Hence many experimental and theoretical studies have been carried out to investigate the properties of the QLLs. Detailed theoretical studies of the QLL using the MD simulation method were reported in the literature [4–6]. However, relatively small slab systems (the numbers of molecules are at most 2,130) under the periodic boundary conditions were used in those simulations. There should be phenomena that can be observed only by large-scale simulation. For instance, it was reported recently through experiments that there are two different morphologies in QLLs with super-micrometer scales on an ice [7, 8], which cannot be reproduced by simulation with a small simulation box.

Therefore, what we have to do in the next step is to perform simulation of ices as large as possible. Employing the TIP4P intermolecular potential [9], I planned to create a computer program to perform such large-scale simulation. Then I found that algorithms for rotational motions were quite complicated (or unstable) compared to those of translational motion, and seemed to consume computational time. It is usually the computation of electrostatic field that takes most of computation time, nevertheless the improvement of the algorithm for rotational motion will save computational time fairly well and enable us to perform longer simulation. In addition, in the cases where intermolecular forces can be truncated at a limited range, the long computational time for the rotational motion will be a bottleneck in computing, and my algorithm may dramatically shorten the simulation period.

Thus, in this thesis, I begin with proposing novel algorithms for ro-

tational motion (FT algorithms). Second, I propose a novel method to prepare configurations of ice-Ih. Third, I seek possible dynamics in QLLs of ices through large-scale MD simulation. The simulation is performed using a computer code adopting the FT algorithm and Fast Multipole Method [10, 11], where I assume that water molecules are rigid and employ the TIP4P intermolecular interaction potential.

1.2 First Part - Algorithms for rotation

The first part of the thesis is devoted to explaining novel numerical algorithms for rotational motion called Fast Time-reversible (FT) algorithms (Chapter 2). Previous to the FT algorithms, I firstly devised an algorithm named Numerically Exactly Time-Reversible (NET) Algorithm [12], where we determine angular position q_{n+1} of step-$(n+1)$ of a molecule from its angular velocity ω_n and angular position q_n using Taylor expansion as usual. We denote the so obtained angular position q_{n+1} by $q_{n+1} = Q(q_n, \omega_n, \Delta t)$. (Here I omit writing the force acting on the molecule for simplicity.) Then the angular velocity ω_{n+1} is determined so that it satisfies time-reversibility condition: $q_n = Q(q_{n+1}, \omega_{n+1}, -\Delta t)$. In other words ω_{n+1} is determined so as to satisfy the following relation:

$$q_n = Q(Q(q_n, \omega_n, \Delta t), \omega_{n+1}, -\Delta t). \tag{1.1}$$

The equation (1.1) yields an algebraic equation with respect to ω_{n+1} whose degree is higher than five.

The NET algorithm is simpler than other time-reversible algorithms for rotational motion, however it requires solving this higher degree equation derived from (1.1). To solve the equation I employed an iteration method that results in requiring a little longer computation time.

Thus I have improved the NET algorithm to get rid of the iteration method by modifying the equation (1.1) within the error of the algorithm $(=O(\Delta t^3))$. This modification does not violate the time-reversibility at all, however it reduces the higher degree equation into simple linear equations. In this way, I have proposed two improved algorithms and I named the

algorithms as Fast Time-reversible (FT) algorithms [13]. In spite of the FT algorithms' implementation codes being very short, the stability of the total energy of FT algorithms is comparable to symplectic algorithm.

In Chapter 2, I firstly explain the idea of the NET algorithm and secondly present the FT algorithms.

In Appendix C, I present modified FT algorithm, which is an improvement of the FT algorithm in a sense, but requires little more computations.

1.3 Second Part - Preparations of ice-Ih

Second, I present the method for making configurations of ice-Ih (Chapter 3). The ices given by the method are used in our simulation (Chapter 4 and [14]).

The positions of oxygen atoms are unique, however those of hydrogen atoms are not determined in advance. All configurations of hydrogen atoms are allowed for ice-Ih if and only if they satisfy the Bernal-Fowler rule i.e., if and only if there exists a single H per O-O bond and O has just two adjacent H's [15].

There are known some methods to generate the configurations of hydrogen atoms of ice-Ih. Perhaps the most used methods give unit cells (large or small layers) satisfying the Bernal-Fowler rule. We paste together the cells and get an arbitrarily large ice-Ih (like construction toys, e.g., Lego). These methods are very useful but they can not be used for making ice-Ih with dislocation as they are. They can make two kinds of planes (even layers and odd layers) by fitting together sidewise, and the two planes can be stuck to each other. However, ices with crystallographic defects can not be obtained in this way. Since the structures of QLL are considered to be influenced by dislocation, it is desirable for future use to make an algorithm that can be used to make ices with dislocation.

Therefore, although I treat ices with dislocation only a little in this thesis, I have made an algorithm to determine the positions of hydrogen atoms of ice-Ih without/with dislocation. The configurations of hydrogen atoms are determined randomly (within the limits of Bernal-Fowler rule)

to realize nearly zero macroscopic Coulomb dipole.

In Chapter 3 and Appendix A, I explain the method by which I make the configurations of water molecules of ice-Ih. In Chapter 3, I restrict ourselves to the configurations of the first layer of ices, and the rest is left to Appendix A. However, the outlines of my method are represented in Chapter 3 except for some complicated cases. I do not show constructions of ices with dislocation in detail. I think the method shown in Part 2 can be easily applied to get configurations of ice-Ih with dislocation. Since the constructions of the algorithm presented in this part can be checked easily (by brute force), I only state the algorithm without detailed explanations.

1.4 Third Part - Simulation

Lastly, I present in Chapter 4 the method and results of my simulation [14]. I have performed my simulation with a computer code adopting one of the FT algorithms and the Fast Multipole Method (FMM) on parallel computers [11]. The FMM is a well-known technique that can reduce the calculation of forces and potentials into order N in order to make the time required for simulation small. The weak scaling of the program code is nearly constant. I employed usual velocity-Verlet algorithm for the calculations of the translational motion.

The initial configurations of H_2O molecules of ices are prepared by the algorithm explained in Chapter 3 and Appendix A. The ices are made in a hexagonal prism shape. Each of them is put in a vacuum box where the z-axis is set perpendicular to the basal (0001) surface of the ice. The present systems are all composed of 1,317,600 molecules. In the runs, the temperatures are controlled to $T = 205$, 215, and 227 K. Since the melting temperature T_m of TIP4P is about 228 K [16], the three temperatures correspond to $T_{\rm m} - 23$, $T_{\rm m} - 13$, and $T_{\rm m} - 1$ K.

The main results I have obtained in the simulation are: (a) The QLLs are bumpy with the bump heights as large as a few inter-bilayer distances of ice. The liquid bumps fluctuate to form and break at a variety of places in a random manner. (b) At relatively lower temperatures, the molecules

in the second bilayer from the outside are partly melted. The ice molecules in the second bilayer melt easily when they are located under the trenches surrounding the liquid bumps. Furthermore, by the change of the locations of liquid bumps over time, the melted areas newly covered by thick bumps recrystallize. (c) At a temperature slightly below the melting point, the second bilayer under the liquid bumps melts entirely to form a liquid sheet, and the bumps sit on the sheet. (d) Microscopic properties (mean squared displacements and O-O distances) of the liquid bumps differ from those of the liquid sheet.

Chapter 2

Algorithms for rotational motion

2.1 Introduction

Molecules are often handled as rigid bodies in molecular dynamics simulation. Despite the simplification, in the case of the TIP4P potential [9, 17] for H_2O molecules for instance, one can reproduce various physical properties of interest with reasonable accuracies such as the freezing and boiling conditions, the electric permittivity, and the interfacial energy of ice and water. TIP4P potential provides a qualitatively correct description of the phase diagram of ice [18] and is widely used for simulation of both water and ice [19, 20]. The simplification by ignoring the fast vibration of constituting atoms of a molecule is highly effective for taking a long time step to realize a long-time simulation [21]. Several time-integration algorithms for rotational motion of rigid body molecules have been deviced, e.g., (i) the Gear's predictor-corrector algorithm [21], (ii) the Matubayasi-Nakahara's algorithm [22], (iii) the symplectic algorithms [23–27], and (iv) the angular momentum Verlet (AMV) algorithm [28].

The algorithm (i) is very accurate for a short time step. However, it is not time-reversible, and becomes unstable or the total energy increases significantly in a long time simulation run. It is highly unstable for longer time steps. The algorithm (ii) is time-reversible and non symplectic. It shows high stability, however, is slightly complicated with its procedure composed of several parts that use auxiliary functions. The algorithms (iii)

have the feature of time-reversibility in addition to the symplecticness and show very high stability; a conserved quantity that is close to Hamiltonian exists. However, it is complicated compared with other algorithms. The algorithm (iv) is easy to understand and interesting since it is constructed in an analogous manner to the velocity-Verlet algorithm. Although it is not time reversible in the strict sense, it shows smaller fluctuation in the total energy than does the leap-flog algorithm [21] when it is applied to the system of tetrahedral molecules. However, I find the total energy increases monotonously during the simulation runs for some systems including a water droplet.

I think that the total energy increase of the AMV algorithm is caused by its lacking of time-reversibility. Therefore, I sought for simple algorithms that satisfy time-reversibility to reduce computational time for large scale simulation. For known time-reversible algorithms, the time-reversibility comes from its symmetric constructions, and I think that the constructions are the cause of their complexity. On the other hand, the time-reversibility of our algorithms comes from new constructions. In short, time-reversibility is not obtained as results, but, on the contrary, we assume its time-reversibility and derive angular velocity as the results of the time-reversibility as explained below. The algorithms obtained in this manner turned out to be ones of the shortest and fastest time-reversible algorithms as results of the novel construction. I named the algorithms Fast Time-reversible (FT) algorithms.

In §2.2, we recall the notion of velocity-Verlet algorithm and its time-reversibility. In §2.3, we derive the equations required for our FT algorithms. In §2.4, the FT algorithms will be applied to simulate a water droplet composed of 499 H_2O molecules to demonstrate their stability in the total energy and computation speeds. §2.5 is devoted to summary and concluding remarks. The FT algorithms will be fast time-reversible ones for rigid molecules, since each of which consists of relatively fewer operations of basic arithmetic and square root (see §2.4.3 and Table 2.1). Simulations of the FT algorithms will show much greater stability than that of the NET algorithm, and comparable stability to that of the Matubayasi-Nakahara

algorithm and symplectic algorithms.

2.2 Time-reversibility

The well-known velocity-Verlet algorithm [21] is a time-reversible algorithm for point atoms. It is quite simple and, in fact, shows accurate total energy conservation. In the velocity-Verlet algorithm, the position and velocity of an atom at step-$(n+1)$ are determined from the corresponding values at step-n as

$$\boldsymbol{r}^{(n+1)} = \boldsymbol{r}^{(n)} + \Delta t\ \boldsymbol{v}^{(n)} + \frac{\Delta t^2}{2m}\boldsymbol{f}^{(n)} \tag{2.1}$$

and

$$\boldsymbol{v}^{(n+1)} = \boldsymbol{v}^{(n)} + \frac{\Delta t}{2m}(\boldsymbol{f}^{(n+1)} + \boldsymbol{f}^{(n)}), \tag{2.2}$$

where $\boldsymbol{r}$, $\boldsymbol{v}$, $\boldsymbol{f}$, m, and Δt are the position, velocity, force, mass, and time step, respectively. Since

$$\boldsymbol{r}^{(n)} = \boldsymbol{r}^{(n+1)} + (-\Delta t)\boldsymbol{v}^{(n+1)} + \frac{(-\Delta t)^2}{2m}\boldsymbol{f}^{(n+1)} \tag{2.3}$$

is obtained by substituting eq. (2.2) to eq. (2.1), the algorithm has time reversibility. In this thesis, considering the above, I propose, for the angular position, a novel time-reversible algorithm that is as simple as the AMV algorithm, in close correspondence to the velocity-Verlet algorithm. To begin, I point out that the combination of eqs. (2.1) and (2.3) gives eq. (2.2). In other words, the combination of the position formula for the next step and the time-reversibility condition determines the proper velocity. The same idea can also be applied to the angular position. In the present algorithm, we will first calculate the angular position of a molecule at step-$(n+1)$ using the Taylor series expansion of the position at step-n up to the 2nd order of the time step. Second, we will determine the angular velocity at step-$(n+1)$ so as to satisfy the time-reversibility condition explicitly between the values at step-n and step-$(n+1)$. To do so, we will need to solve a non linear equation of the angular velocity at step-$(n+1)$, which

yields an algebraic equation with respect to angular velocities, whose degree is more than five. The numerically exact time-reversible (NET) algorithm ([12]) determines numerically the angular velocity by solving the equation by iteration and thus time-reversible. The angular velocities of the FT algorithm are determined by equations obtained by slightly modifying the equation, and the modified equations can be solved easily. In the next section I explain the FT algorithm.

2.3 Fast time-reversible (FT) algorithm

2.3.1 Introduction

The aim of this section is to propose fast and time-reversible algorithms for rigid body molecules without an iteration procedure indispensable for the NET algorithm. I call the algorithms Fast Time-reversible (FT) algorithms. I will give two FT algorithms. The difference between them lies only in the method of eliminating the iteration procedure. In a method, the three components of the angular velocity vector are treated as a set. Mutually different treatments are applied to the three components with the feature of phase-space conservation in the other method.

2.3.2 FT algorithms

The motion of a rigid molecule is decomposed into the translational motion and the rotational motion around the centroid. In applying FT algorithms to simulation of water (this chapter and Chapter 4), we employ the velocity-Verlet algorithm for translational motion. Since we use FT algorithms always coupled with the velocity-Verlet algorithm, I mention the velocity-Verlet algorithm again in the following.

The equations of the FT algorithms for the rotational motion are derived in §2.3.5 − §2.3.9. In §2.3.10 I show the procedure of the FT algorithm.

2.3.3 Equations of FT algorithms

For simplicity I present the set of equations common to two FT algorithms for a single rigid molecule, which is composed of the updates of centroid position, centroid velocity, angular position and angular velocity. In actual simulation of a molecular system, the equations will be applied in parallel to all the rigid molecules.

2.3.4 Determination of centroid position and velocity

In applying FT algorithms to simulation, we always employ the time-reversible velocity-Verlet algorithm to describe the translational motion of the centroid of a rigid molecule:

$$\vec{r}(t+\Delta t) = \vec{r}(t) + \Delta t \left(\vec{v}(t) + \frac{\Delta t}{2m} \vec{f}(t) \right), \tag{2.4}$$

$$\vec{v}(t+\Delta t) = \vec{v}(t) + \frac{\Delta t}{2m} (\vec{f}(t) + \vec{f}(t+\Delta t)). \tag{2.5}$$

Here the vectors $\vec{r}$ and $\vec{v}$ represent the position and velocity of the centroid, respectively. The $\vec{f}$ is the summation of the forces on the constituting atoms of the molecule, and m is the mass of the molecule. The force on each atom is assumed to be a function of the atomic positions only.

2.3.5 Determination of angular position

The angular position of a rigid molecule is described with the quaternion. The aim of this subsection is to derive Eq. (2.23) below, by which we update the angular position. We introduce a coordinate frame fixed to a rigid body molecule so that the moment of inertia tensor is diagonal; that is, the body-fixed coordinates of a point are obtained as its projections on the principal axes of inertia. We assume that the origin $O_{\rm b}$ of the body-fixed frame coincides with the centroid of the rigid molecule. Similarly the space-fixed frame is introduced, whose origin is denoted by $O_{\rm s}$.

Let a matrix $\overleftrightarrow{R}_q$ rotate the three axes of the space-fixed frame to be parallel to that of the body-fixed frame. The $\overleftrightarrow{R}_q$ is parametrized by a unit

quaternion $\vec{q} = {}^{\mathrm{t}}(q_0, q_1, q_2, q_3)$ (the superscript "t" means the transpose operation) with $|\vec{q}| = 1$:

$$\overleftrightarrow{R_q} = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}. \quad (2.6)$$

We add the superscript "(b)" or "(s)" to clarify that the associated vector is represented as a 3×1 matrix in the body-fixed or space-fixed frame, respectively. Then, for any point P,

$$\overrightarrow{O_{\mathrm{b}}P}^{(\mathrm{s})} = \overleftrightarrow{R_q}\overrightarrow{O_{\mathrm{b}}P}^{(\mathrm{b})}. \quad (2.7)$$

The quaternion and Euler angles [29] are related to each other through

$$q_0 = \cos(\theta/2)\cos[(\phi+\psi)/2],$$
$$q_1 = \sin(\theta/2)\cos[(\phi-\psi)/2],$$
$$q_2 = \sin(\theta/2)\sin[(\phi-\psi)/2],$$

and

$$q_3 = \cos(\theta/2)\sin[(\phi+\psi)/2].$$

Here θ, ϕ, and ψ are the three Euler angles of the body-fixed frame relative to the space-fixed one in the standard convention [29, 30]. Hereafter, we exploit the unit quaternion $\vec{q}$ exclusively to represent the rotational position of a rigid body in the space-fixed frame through Eq. (2.7).

Since we have assumed that the three axes of the body-fixed frame are the principal axes of inertia, we can write the angular momentum $\vec{L}^{(\mathrm{b})} = {}^{\mathrm{t}}(L_x, L_y, L_z)$ as follows:

$$L_x = I_x\omega_x(t), \quad L_y = I_y\omega_y(t), \quad L_z = I_z\omega_z(t), \quad (2.8)$$

where I_x, I_y, and I_z are the principal moments of inertia, and $\vec{\omega}^{(\mathrm{b})}(t) = {}^{\mathrm{t}}(\omega_x(t), \omega_y(t), \omega_z(t))$ is the angular velocity of the rigid molecule. Since angular velocity vectors are always represented in the body-fixed frame in

this thesis, we use $\vec{\omega}(t)$ to mean $\vec{\omega}^{(\mathrm{b})}(t)$ hereafter. It is known that the following identity holds [22, 23]:

$$\frac{d}{dt}\begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix}\begin{pmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}. \tag{2.9}$$

Let

$$\overleftrightarrow{A}[\vec{\omega}] = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix} \tag{2.10}$$

and

$$\vec{q}(t) = {}^{\mathrm{t}}(q_0(t), q_1(t), q_2(t), q_3(t)). \tag{2.11}$$

Notice here that $\overleftrightarrow{A}[\vec{\omega}]$ above is not a product of $\overleftrightarrow{A}$ and $\vec{\omega}$, but a matrix of $\vec{\omega}$. Then Eq. (2.9) is rewritten as ([22])

$$\frac{d}{dt}\vec{q}(t) = \overleftrightarrow{A}\left[\frac{1}{2}\vec{\omega}(t)\right]\vec{q}(t), \tag{2.12}$$

and then

$$\frac{d^2}{dt^2}\vec{q}(t) = \frac{d}{dt}\left(\frac{d}{dt}\vec{q}(t)\right)$$

$$= \left(\overleftrightarrow{A}\left[\frac{1}{2}\vec{\omega}(t)\right]\right)^2\vec{q}(t) + \overleftrightarrow{A}\left[\frac{1}{2}\frac{d}{dt}\vec{\omega}(t)\right]\vec{q}(t).$$

Since $\left(\overleftrightarrow{A}\left[\frac{1}{2}\vec{\omega}(t)\right]\right)^2 = -\left|\frac{1}{2}\vec{\omega}(t)\right|^2\overleftrightarrow{E}$ ($\overleftrightarrow{E}$ is the identity matrix), the equation above yields

$$\frac{d^2}{dt^2}\vec{q}(t) = -\left|\frac{1}{2}\vec{\omega}(t)\right|^2\vec{q}(t) + \overleftrightarrow{A}\left[\frac{1}{2}\frac{d}{dt}\vec{\omega}(t)\right]\vec{q}(t). \tag{2.13}$$

From Eqs. (2.12) and (2.13),

$$\begin{aligned}
&\vec{q}(t) + \frac{d}{dt}\vec{q}(t)\Delta t + \frac{1}{2}\frac{d^2}{dt^2}\vec{q}(t)\Delta t^2 = \vec{q}(t) + \overleftrightarrow{A}\left[\frac{1}{2}\vec{\omega}(t)\right]\vec{q}(t)\Delta t \\
&+\frac{1}{2}\left(-\left|\frac{1}{2}\vec{\omega}(t)\right|^2\vec{q}(t) + \overleftrightarrow{A}\left[\frac{1}{2}\frac{d}{dt}\vec{\omega}(t)\right]\vec{q}(t)\right)\Delta t^2 \\
&= \left(1 - \frac{1}{2}\left|\frac{1}{2}\vec{\omega}(t)\right|^2\Delta t^2\right)\vec{q}(t) \\
&+\overleftrightarrow{A}\left[\frac{1}{2}\left(\vec{\omega}(t) + \frac{1}{2}\frac{d}{dt}\vec{\omega}(t)\Delta t\right)\Delta t\right]\vec{q}(t), \qquad (2.14)
\end{aligned}$$

where we have used the fact that $\overleftrightarrow{A}[\vec{\omega}]$ is linear with respect to $\vec{\omega}$. Let me define

$$s = 1 - \frac{1}{2}\left|\frac{1}{2}\vec{\omega}(t)\right|^2\Delta t^2$$

and

$$\overrightarrow{V} = \frac{1}{2}\left(\vec{\omega}(t) + \frac{1}{2}\frac{d}{dt}\vec{\omega}(t)\Delta t\right)\Delta t. \qquad (2.15)$$

Then, Eq. (2.14) is equal to $\left(s\overleftrightarrow{E} + \overleftrightarrow{A}[\overrightarrow{V}]\right)\vec{q}(t)$, and thus

$$\begin{aligned}
\vec{q}(t+\Delta t) &= \vec{q}(t) + \frac{d}{dt}\vec{q}(t)\Delta t + \frac{1}{2}\frac{d^2}{dt^2}\vec{q}(t)\Delta t^2 + O(\Delta t^3) \\
&= \left(s\overleftrightarrow{E} + \overleftrightarrow{A}[\overrightarrow{V}]\right)\vec{q}(t) + O(\Delta t^3). \qquad (2.16)
\end{aligned}$$

Since $s > 0$ and $|\vec{V}| < 1$ for usual time steps [31] (see Addendum of this chapter and Appendix C, where these assumptions are reoved), it follows from Eq. (2.16) that

$$\vec{q}(t+\Delta t) = \left(\sqrt{1 - |\overrightarrow{V}|^2}\overleftrightarrow{E} + \overleftrightarrow{A}[\overrightarrow{V}]\right)\vec{q}(t) + O(\Delta t^3). \qquad (2.17)$$

Here we have used the fact

$$\begin{aligned} s &= 1 - \frac{1}{2}\left|\frac{1}{2}\vec{\omega}(t)\right|^2 \Delta t^2 \\ &= 1 - \frac{1}{2}\left|\frac{1}{2}\vec{\omega}(t) + \frac{1}{4}\frac{d}{dt}\vec{\omega}(t)\Delta t\right|^2 \Delta t^2 + O(\Delta t^3) \\ &= 1 - \frac{1}{2}|\vec{V}|^2 + O(\Delta t^3) = \sqrt{1 - |\overrightarrow{V}|^2} + O(\Delta t^3). \end{aligned}$$

Note that $\sqrt{1 - |\overrightarrow{V}|^2}\overleftrightarrow{E} + \overleftrightarrow{A}[\overrightarrow{V}]$ in Eq. (2.17) is an orthogonal matrix, which conserves the distance or the norm of the quaternion.

We define the orthogonal matrix $\overleftrightarrow{R}[\vec{v}]$ for a vector $\vec{v} = {}^{\mathrm{t}}(v_x, v_y, v_z)$ ($|\vec{v}| < 1$) as

$$\begin{aligned} \overleftrightarrow{R}[\vec{v}] &= \sqrt{1 - |\vec{v}|^2}\overleftrightarrow{E} + \overleftrightarrow{A}[\vec{v}] \\ &= \begin{pmatrix} \sqrt{1 - |\vec{v}|^2} & -v_x & -v_y & -v_z \\ v_x & \sqrt{1 - |\vec{v}|^2} & v_z & -v_y \\ v_y & -v_z & \sqrt{1 - |\vec{v}|^2} & v_x \\ v_z & v_y & -v_x & \sqrt{1 - |\vec{v}|^2} \end{pmatrix}. \end{aligned} \tag{2.18}$$

It is easy to see that $\overleftrightarrow{R}[\vec{v}]$ satisfies

$$\overleftrightarrow{R}[\vec{v}]\overleftrightarrow{R}[-\vec{v}] = \overleftrightarrow{E}. \tag{2.19}$$

Using the notations above, we rewrite Eq. (2.17) as

$$\vec{q}(t + \Delta t) = \overleftrightarrow{R}[\overrightarrow{V}]\vec{q}(t) + O(\Delta t^3). \tag{2.20}$$

Let me define

$$\vec{\phi}(t, \Delta t) = \vec{\omega}(t) + \frac{1}{2}\frac{d}{dt}\vec{\omega}(t)\Delta t. \tag{2.21}$$

Then we have $\overrightarrow{V} = \frac{1}{2}\vec{\phi}(t, \Delta t)\Delta t$ and Eq. (2.20) yields

$$\vec{q}(t + \Delta t) = \overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t, \Delta t)\Delta t\right]\vec{q}(t) + O(\Delta t^3). \tag{2.22}$$

From Eq. (2.22), we obtain an equation to update the quaternion $\vec{q}$ as

$$\vec{q}(t+\Delta t) = \overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t,\Delta t)\Delta t\right]\vec{q}(t). \tag{2.23}$$

Here $\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t,\Delta t)\Delta t\right]$ is an orthogonal matrix (i.e., distance conserving), and hence $|\vec{q}(t+\Delta t)| = |\vec{q}(t)|$. Consequently, the $\vec{q}$ is normalized automatically for any Δt.

There remains the task of calculating $\vec{\phi}(t,\Delta t)$. From Eq. (2.21),

$$\begin{aligned}
\phi_x(t,\Delta t) &= \omega_x(t) + \frac{1}{2}\frac{d\omega_x(t)}{dt}\Delta t \\
\phi_y(t,\Delta t) &= \omega_y(t) + \frac{1}{2}\frac{d\omega_y(t)}{dt}\Delta t \\
\phi_z(t,\Delta t) &= \omega_z(t) + \frac{1}{2}\frac{d\omega_z(t)}{dt}\Delta t.
\end{aligned} \tag{2.24}$$

Here, the derivatives of the angular velocity are given by the following Euler's equation of motion:

$$\frac{d\omega_i}{dt} = \frac{I_j - I_k}{I_i}\omega_j\omega_k + \frac{t_i}{I_i}, \tag{2.25}$$

where $(i,j,k) = (x,y,z), (y,z,x)$, and (z,x,y), and t_i is the component of the torque $\vec{\tau}^{(\mathrm{b})} = {}^{\mathrm{t}}(t_x, t_y, t_z)$. Then we have

$$\begin{aligned}
\phi_x(t,\Delta t) &= \omega_x(t) + (\alpha\omega_y(t)\omega_z(t) + \delta t_x(t))\Delta t \\
\phi_y(t,\Delta t) &= \omega_y(t) + (\beta\omega_x(t)\omega_z(t) + \lambda t_y(t))\Delta t \\
\phi_z(t,\Delta t) &= \omega_z(t) + (\gamma\omega_x(t)\omega_y(t) + \mu t_z(t))\Delta t,
\end{aligned} \tag{2.26}$$

where

$$\begin{aligned}
\alpha = \frac{I_y - I_z}{2I_x}, \quad \beta &= \frac{I_z - I_x}{2I_y}, \quad \gamma = \frac{I_x - I_y}{2I_z}, \\
\delta = \frac{1}{2I_x}, \quad \lambda &= \frac{1}{2I_y}, \text{ and } \quad \mu = \frac{1}{2I_z}.
\end{aligned} \tag{2.27}$$

We can calculate $\vec{\phi}(t,\Delta t)$ with Eq. (2.26), by which we update the angular position of the rigid molecule with Eq. (2.23).

Note that the equations of $\vec{\phi} = {}^{\mathrm{t}}(\phi_x, \phi_y, \phi_z)$ in Eq. (2.26) will be changed slightly later within the order of Δt^2 to avoid an iteration procedure in determining angular velocity. Even in that case, the order of error of updated quaternion remains within Δt^3 (see §2.4).

I remark that $\vec{\phi}(t, \Delta t)$ introduced in Eq. (2.21) can be regarded as the angular velocity at the midstep $t + \frac{1}{2}\Delta t$ in [22], and that we can derive Eq. (2.23) also by using the equations in that reference. However, our derivation above will be helpful to clarify our method.

2.3.6 Determination of angular velocity

The principal aim of this subsection is to derive Eq. (2.32) below that the updated angular velocity should obey. We determine the updated angular velocity in the same way as the NET algorithm, that is, we determine it *so as to* satisfy the time-reversibility condition [12]. From Eq. (2.23), the time-reversibility condition gives

$$\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t+\Delta t, -\Delta t)(-\Delta t)\right]\vec{q}(t+\Delta t) = \vec{q}(t). \tag{2.28}$$

Combining Eqs. (2.23) and (2.28), we have

$$\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t+\Delta t, -\Delta t)(-\Delta t)\right]\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t, \Delta t)\Delta t\right]\vec{q}(t)$$

$$= \vec{q}(t). \tag{2.29}$$

Therefore, we determine $\vec{\phi}(t + \Delta t, -\Delta t)$ by the following equation:

$$\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t+\Delta t, -\Delta t)(-\Delta t)\right]\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t, \Delta t)\Delta t\right] = \overleftrightarrow{E}. \tag{2.30}$$

Since

$$\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t, \Delta t)(-\Delta t)\right]\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t, \Delta t)\Delta t\right] = \overleftrightarrow{E} \tag{2.31}$$

from Eq. (2.19), we find

$$\vec{\phi}(t+\Delta t, -\Delta t) = \vec{\phi}(t, \Delta t) \tag{2.32}$$

from Eq. (2.30). Using Eqs. (2.26) and (2.32), we obtain the following equations:

$$\begin{aligned}
\omega_x^+ - (\alpha\omega_y^+\omega_z^+ + \delta t_x^+)\Delta t &= \omega_x + (\alpha\omega_y\omega_z + \delta t_x)\Delta t \\
\omega_y^+ - (\beta\omega_x^+\omega_z^+ + \lambda t_y^+)\Delta t &= \omega_y + (\beta\omega_x\omega_z + \lambda t_y)\Delta t \\
\omega_z^+ - (\gamma\omega_x^+\omega_y^+ + \mu t_z^+)\Delta t &= \omega_z + (\gamma\omega_x\omega_y + \mu t_z)\Delta t,
\end{aligned} \tag{2.33}$$

where we mean $\omega_i = \omega_i(t)$, $\omega_i^+ = \omega_i(t+\Delta t)$, $t_i = t_i(t)$, and $t_i^+ = t_i(t+\Delta t)$ for $i = \{x, y, z\}$. Equations (2.33) contain unknown variables ω_x^+, ω_y^+, and ω_z^+. If we combine these three equations to obtain a single equation of a single variable, the degree of the single variable equation is at least 5. It is difficult to solve it algebraically. In the NET algorithm, we solve similar equations numerically for every time step by an iteration method.

In [12], we derived the equations similar in meaning to Eqs. (2.23), (2.26), and (2.32). However, those equations in [12] were formulated suitable for iteration procedure. We cannot apply the following iteration-free method directly to those equations obtained in [12].

2.3.7 Ideas to remove iteration procedure

I propose two methods to determine the updated angular velocity without iteration. As mentioned before, we modify Eqs. (2.26) slightly within the order of Δt^2. We use the modified equations to evolve quaternion in time by Eq. (2.23) and to get the updated angular velocity by setting them into Eq. (2.32). Hereafter, we abbreviate
$\phi_x(t, \Delta t)$, $\phi_y(t, \Delta t)$, $\phi_z(t, \Delta t)$, $\phi_x(t+\Delta t, -\Delta t)$, $\phi_y(t+\Delta t, -\Delta t)$,
$\phi_z(t+\Delta t, -\Delta t)$, $\omega_x(t)$, $\omega_y(t)$, $\omega_z(t)$, $\omega_x(t+\Delta t)$, $\omega_y(t+\Delta t)$, and $\omega_z(t+\Delta t)$ as

$$\begin{aligned}
&\phi_x(t, \Delta t) = \phi_x, \quad \phi_y(t, \Delta t) = \phi_y, \quad \phi_z(t, \Delta t) = \phi_z, \\
&\phi_x(t+\Delta t, -\Delta t) = \phi_x^+, \quad \phi_y(t+\Delta t, -\Delta t) = \phi_y^+, \\
&\phi_z(t+\Delta t, -\Delta t) = \phi_z^+ \quad \omega_x(t) = \omega_x, \quad \omega_y(t) = \omega_y, \quad \omega_z(t) = \omega_z, \\
&\omega_x(t+\Delta t) = \omega_x^+, \quad \omega_y(t+\Delta t) = \omega_y^+, \quad \omega_z(t+\Delta t) = \omega_z^+,
\end{aligned}$$

respectively.

2.3.8 Method 1: modification to linear equations

Instead of Eqs. (2.26), we redefine

$$\vec{\phi}(t, \Delta t) = {}^{\mathrm{t}}(\phi_x, \phi_y, \phi_z)$$

by the following combined equations:

$$\tilde{\omega}_x = \omega_x + \delta t_x \Delta t, \quad \tilde{\omega}_y = \omega_y + \lambda t_y \Delta t, \quad \tilde{\omega}_z = \omega_z + \mu t_z \Delta t \tag{2.34}$$

and

$$\begin{aligned} \phi_x &= \tilde{\omega}_x + \alpha \tilde{\omega}_z \phi_y \Delta t, \\ \phi_y &= \tilde{\omega}_y + \beta \tilde{\omega}_x \phi_z \Delta t, \\ \phi_z &= \tilde{\omega}_z + \gamma \tilde{\omega}_y \phi_x \Delta t. \end{aligned} \tag{2.35}$$

We solve these linear equations (Eqs. 2.35) to get $\vec{\phi}(t, \Delta t)$. The so defined ϕ_x, ϕ_y, and ϕ_z differ from the original ones within the order of Δt^2, which is easy to see by solving Eqs. (2.34) and (2.35). For example,

$$\begin{aligned} \phi_x &= (\tilde{\omega}_x + \alpha \tilde{\omega}_y \tilde{\omega}_z \Delta t + \alpha\beta \tilde{\omega}_x \tilde{\omega}_z^2 \Delta t^2)/(1 - \alpha\beta\gamma \tilde{\omega}_x \tilde{\omega}_y \tilde{\omega}_z \Delta t^3) \\ &= \omega_x + (\alpha \omega_y \omega_z + \delta t_x)\Delta t + O(\Delta t^2). \end{aligned}$$

Therefore, the order of error of the updated quaternion given by Eq. (2.23) for $\vec{\phi} = {}^{\mathrm{t}}(\phi_x, \phi_y, \phi_z)$ defined above remains within Δt^3.

Then, $\vec{\phi}(t + \Delta t, -\Delta t) = {}^{\mathrm{t}}(\phi_x^+, \phi_y^+, \phi_z^+)$ is given similarly by

$$\begin{aligned} \tilde{\omega}_x^+ &= \omega_x^+ - \delta t_x^+ \Delta t, \\ \tilde{\omega}_y^+ &= \omega_y^+ - \lambda t_y^+ \Delta t, \\ \tilde{\omega}_z^+ &= \omega_z^+ - \mu t_z^+ \Delta t \end{aligned} \tag{2.36}$$

and

$$\begin{aligned} \phi_x^+ &= \tilde{\omega}_x^+ - \alpha \tilde{\omega}_z^+ \phi_y^+ \Delta t, \\ \phi_y^+ &= \tilde{\omega}_y^+ - \beta \tilde{\omega}_x^+ \phi_z^+ \Delta t, \end{aligned} \tag{2.37}$$

$$\phi_z^+ = \tilde{\omega}_z^+ - \gamma\tilde{\omega}_y^+\phi_x^+\Delta t.$$

Since $\phi_x^+ = \phi_x$, $\phi_y^+ = \phi_y$, and $\phi_z^+ = \phi_z$ from Eq. (2.32), we can calculate the updated angular velocity ${}^{\mathrm{t}}(\omega_x^+, \omega_y^+, \omega_z^+)$ by considering the reverse order operation of Eqs. (2.36) and (2.37). Equations (2.37) are linear in $\tilde{\omega}_x^+, \tilde{\omega}_y^+$, and $\tilde{\omega}_z^+$, which are easy to solve. Then we get the updated angular velocity ${}^{\mathrm{t}}(\omega_x^+, \omega_y^+, \omega_z^+)$ from Eqs. (2.36).

Method 1 requires solving three linear equations for three variables twice (one for Eq. 2.35 and the other for Eq. 2.37) to get the updated angular velocity. In the next subsection, a method consisting of substitutions only but losing symmetry with respect to the three components in Eqs. (2.26) will be proposed.

2.3.9 Method 2: successive substitution

Instead of Eqs. (2.26), we redefine $\vec{\phi}(t, \Delta t) = {}^{\mathrm{t}}(\phi_x, \phi_y, \phi_z)$ *successively* as follows. The arrow that is facing left in an equation below stands for substitution in a computer code. We proceed in the sequence:

$$\phi_x(1) \leftarrow \omega_x, \quad \phi_y(1) \leftarrow \omega_y, \quad \phi_z(1) \leftarrow \omega_z \tag{2.38}$$

$$\begin{aligned} \phi_x(2) &\leftarrow \phi_x(1) + \delta t_x \Delta t, \\ \phi_y(2) &\leftarrow \phi_y(1) + \lambda t_y \Delta t, \\ \phi_z(2) &\leftarrow \phi_z(1) + \mu t_z \Delta t \end{aligned} \tag{2.39}$$

$$\phi_x \leftarrow \phi_x(2) + \alpha\phi_y(2)\phi_z(2)\Delta t \tag{2.40a}$$
$$\phi_y \leftarrow \phi_y(2) + \beta\phi_x\phi_z(2)\Delta t \tag{2.40b}$$
$$\phi_z \leftarrow \phi_z(2) + \gamma\phi_x\phi_y\Delta t. \tag{2.40c}$$

Here, ϕ_x, ϕ_y, and ϕ_z defined in Eqs. (2.40a)-(2.40c) differ from the original ones within the order of Δt^2. Thus the order of error of the updated quaternion remains within Δt^3 as before.

Then, $\vec{\phi}(t+\Delta t, -\Delta t) = {}^{\mathrm{t}}(\phi_x^+, \phi_y^+, \phi_z^+)$ is given similarly as above, and we can solve it to get the updated angular velocity ${}^{\mathrm{t}}(\omega_x^+, \omega_y^+, \omega_z^+)$ as follows:

$$\begin{aligned}
\phi_z^+(2) &\leftarrow \phi_z^+ + \gamma\phi_x^+\phi_y^+\Delta t && \text{(2.41a)}\\
\phi_y^+(2) &\leftarrow \phi_y^+ + \beta\phi_x^+\phi_z^+(2)\Delta t && \text{(2.41b)}\\
\phi_x^+(2) &\leftarrow \phi_x^+ + \alpha\phi_y^+(2)\phi_z^+(2)\Delta t && \text{(2.41c)}
\end{aligned}$$

$$\begin{aligned}
&\phi_x^+(1) \leftarrow \phi_x^+(2) + \delta t_x^+\Delta t,\\
&\phi_y^+(1) \leftarrow \phi_y^+(2) + \lambda t_y^+\Delta t,\\
&\phi_z^+(1) \leftarrow \phi_z^+(2) + \mu t_z^+\Delta t
\end{aligned} \qquad (2.42)$$

$$\omega_x^+ \leftarrow \phi_x^+(1), \quad \omega_y^+ \leftarrow \phi_y^+(1), \quad \omega_z^+ \leftarrow \phi_z^+(1). \qquad (2.43)$$

For Eqs. (2.41) $\phi_i^+ = \phi_i$ $(i = \{x, y, z\})$ from Eq. (2.32). Note that Eqs. (2.41) are arranged in the direction opposite to Eqs. (2.40) so that the time-reversibility holds.

I denote the FT algorithms with Method 1 and 2 as FT1 and FT2 algorithms, respectively. Note that the FT2 algorithm satisfies the following equation of phase-space conservation (see Eq. 14 in [22]):

$$\left|\frac{\partial(\vec{r}(t+\Delta t), \vec{v}(t+\Delta t), \vec{q}(t+\Delta t), \vec{\omega}(t+\Delta t))}{\partial(\vec{r}(t), \vec{v}(t), \vec{q}(t), \vec{\omega}(t))}\right| = 1. \qquad (2.44)$$

2.3.10 FT algorithm for single time step

We describe here the procedure for a single time step of the FT algorithm, especially for FT2, for a single rigid molecule composed of atoms. In actual simulation of a molecular system, the algorithm will be applied in parallel to all the rigid molecules. The procedure of the FT1 algorithm is similar.

The procedure of the FT2 algorithm consists of the following eight steps. They are the steps required to evaluate $\vec{r}$, $\vec{v}$, $\vec{q}$, and $\vec{\omega}$ at time $t + \Delta t$ from that at time t. Step 4 is devoted to the calculation of the quaternion and Step 8 the angular velocity. These two steps distinguish the FT algorithm

from the other ones. The other steps are generally installed in every algorithm for the motion of rigid molecule in the quaternion representation.

Setting: The body-fixed frame is introduced with its axes corresponding to the principal axes of inertia of a rigid molecule, whose origin coincides with the centroid of the rigid molecule. We denote its principal moments of inertia by I_x, I_y, and I_z. Here we use the constants $\alpha, \beta, \gamma, \delta, \lambda$, and μ defined in Eqs. (2.27) . Take the data at time t of the molecule: the position of the centroid $\vec{r}(t)^{(\mathrm{s})}$, the velocity of the centroid $\vec{v}(t)^{(\mathrm{s})}$, the quaternion of the rigid molecule $\vec{q}(t)$, and the angular velocity of the rigid molecule $\vec{\omega}(t)^{(\mathrm{b})} = {}^{\mathrm{t}}(\omega_x, \omega_y, \omega_z)$.

Step 1: Calculate the atomic positions of the molecule in the space-fixed frame by Eqs. (2.6) and (2.7).

Step 2: Calculate the forces on the atoms using the atomic positions calculated in Step 1. Then calculate the force $\vec{f}(t)^{(\mathrm{s})}$ acting on the centroid, and the torque $\vec{\tau}(t)^{(\mathrm{b})} = {}^{\mathrm{t}}(t_x, t_y, t_z)$.

Step 3: Determine the updated position of the centroid by Eq. (2.4).

Step 4: Determine $\vec{\phi}(t, \Delta t) = {}^{\mathrm{t}}(\phi_x(t, \Delta t), \phi_y(t, \Delta t), \phi_z(t, \Delta t))$ by Eqs. (2.38)-(2.40). Set it into Eq. (2.23) to determine the updated quaternion.

Step 5: Calculate the updated positions of the atoms in the space-fixed frame by Eq. (2.7). (In actual simulation, we install the procedure to normalize the quaternion every thousand steps to avoid numerical error. It is not always necessary.)

Step 6: Calculate the updated forces on the atoms. Then obtain the torque $\vec{\tau}^{+(\mathrm{b})} = {}^{\mathrm{t}}(t_x^+, t_y^+, t_z^+)$ and the force $\vec{f}(t + \Delta t)$ acting on the centroid.

Step 7: Determine the updated velocity of the centroid by Eq. (2.5).

Step 8: Determine the updated angular velocity, ω_x^+, ω_y^+, and ω_z^+, by Eqs. (2.41)-(2.43) after setting $\phi_x^+ = \phi_x$, $\phi_y^+ = \phi_y$, and $\phi_z^+ = \phi_z$ where ϕ_x, ϕ_y, and ϕ_z are obtained in Step 4 and t_x^+, t_y^+, and t_z^+ in Step 6.
Go to Step 3.

Note that if we choose to use the FT1 algorithm, we have only to replace the equations in Step 4 and Step 8 by Eqs. (2.34) and (2.35) and Eqs. (2.37)

and (2.36), respectively. The FT algorithms can also be applied to linear molecules. For a linear molecule, we set axis-z of the body-fixed frame along the molecule. Then we have $I_x = I_y > 0$ and $I_z = 0$. Here we set $\omega_3 = 0$, $\alpha = \frac{1}{2}, \delta = 1/I_x, \beta = -\frac{1}{2}, \lambda = 1/I_y, \gamma = 0, \mu = 0$, and $t_z = 0$. Under such a setting, the procedure explained above can be used for linear molecules.

2.4 Computation results: Comparison among FT1, FT2, symplectic, and NET algorithms

The FT algorithms are applied to a water droplet in vacuum. We choose to use the TIP4P inter-molecular potential, in which an H_2O molecule is described as a rigid, planar four charged points. It is known that various physical properties [9,17] are reproduced well in both liquid and crystalline phases with the TIP4P potential. my purposes of the present application are to examine the stability and the computation timings of the FT algorithms in realistic settings through comparison of that of existing algorithms.

To prepare a water droplet, we firstly cut, from crystalline ice in Ih-phase [32], a collection of 499 molecules in spherical shape. Secondly, we keep the temperature of the system at around $T = 300$ K by controlling both translational and angular velocities for more than 1.0 ns to obtain a water droplet with a diameter of about 3.0 nm in vacuum at thermal equilibrium; no molecule is detached from the droplet. This preparation simulation is performed with $\Delta t = 2.5$ fs using either FT, symplectic, or NET algorithm to obtain three initial configurations for each algorithm.

For precise comparison, we follow the method in [22] and introduce the local error $\epsilon(1)$ and the global error $\tilde{\epsilon}$ (see, below). We define

$$\epsilon(n) = \left\langle \left| \frac{E(i+n)}{E(i)} - 1 \right| \right\rangle \ , \tag{2.45}$$

where $E(i)$ is the total energy of the system at step-i and the average $\langle \cdots \rangle$ is taken over all possible i and three runs of 10^5 steps starting from

different configurations. The $\epsilon(1)$ gives the relative error of the total energy after Δt. The global error is defined as $\tilde{\epsilon} = \lim_{n\to\infty} \epsilon(n)/n$. To reduce the numerical fluctuation, we, in practice, calculate

$$\tilde{\epsilon} = \frac{\epsilon(10000) - \epsilon(1000)}{9000} \ . \tag{2.46}$$

In this section, the symplectic algorithm is coded according to [27] after removing unnecessary parts as the thermostat part, and the quaternions are normalized every thousand steps to avoid numerical error as in the case of the FT algorithm.

2.4.1 Stability of FT1 algorithm

Figure 2.1 shows the local error, $\epsilon(1)$, and the global error, $\tilde{\epsilon}$, for the FT1, symplectic, and NET algorithms with various Δt [33]. It is seen in Fig. 2.1 that $\log \epsilon(1)$ grows linearly with $\log \Delta t$ with the slope of approximately 3 in the three algorithms. It reflects the fact that the algorithms contain the local errors of order Δt^3. We find in Fig. 2.1 that the global error, $\tilde{\epsilon}$, in the FT1 algorithm is about 10% of that in the NET algorithm and is intermediate of those in the NET and symplectic algorithms.

I note that there exist other possibilities of modifying Eqs. (2.26) similar to Eqs. (2.35). One is the way to use the equations below instead of Eqs. (2.35):

$$\phi_x = \tilde{\omega}_x + \alpha\phi_z\tilde{\omega}_y\Delta t, \quad \phi_y = \tilde{\omega}_y + \beta\phi_x\tilde{\omega}_z\Delta t, \quad \phi_z = \tilde{\omega}_z + \gamma\phi_y\tilde{\omega}_x\Delta t. \tag{2.47}$$

The other is to take the mean of Eqs. (2.35) and (2.47):

$$\begin{aligned}
\phi_x &= \tilde{\omega}_x + \frac{1}{2}\alpha(\tilde{\omega}_z\phi_y + \phi_z\tilde{\omega}_y)\Delta t, \\
\phi_y &= \tilde{\omega}_y + \frac{1}{2}\beta(\tilde{\omega}_x\phi_z + \phi_x\tilde{\omega}_z)\Delta t, \\
\phi_z &= \tilde{\omega}_z + \frac{1}{2}\gamma(\tilde{\omega}_y\phi_x + \phi_y\tilde{\omega}_x)\Delta t.
\end{aligned} \tag{2.48}$$

These two algorithms give quite similar results for $\epsilon(1)$ and $\tilde{\epsilon}$ to that of the FT1 algorithm.

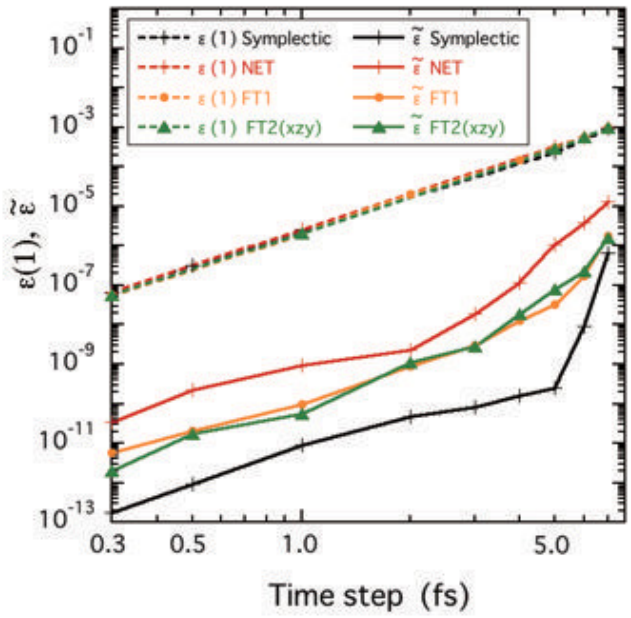

Figure 2.1: Local errors, $\epsilon(1)$, and global errors, $\tilde{\epsilon}$, in the water droplet simulation at $T = 300$ K in the symplectic, FT1, FT2 (xzy), and NET algorithms for various values of Δt.

2.4.2 Stability of FT2 algrithm

Here we compare both local and global errors of the FT2 algorithm with that of other algorithms. In §2.3.9 we have obtained ϕ_x, ϕ_y, and ϕ_z by substituting successively as Eqs. (2.38)-(2.40). However, the sequential order of Eqs. (2.40a), (2.40b), and (2.40c) can be different. There are $3! = 6$ ways of permuting Eqs. (2.40a), (2.40b), and (2.40c). I perform the simulations for all six cases, to find no significant differences. Here I show three simulations of the six cases. For the case denoted as (xyz), the order is Eqs. (2.40a), (2.40b),and (2.40c). For (yzx), the order is Eqs. (2.40b), (2.40c), and (2.40a). For (xzy), the order is Eqs. (2.40a), (2.40c), and (2.40b). In each case, Eqs. (2.41a)-(2.41c) are arranged in the direction

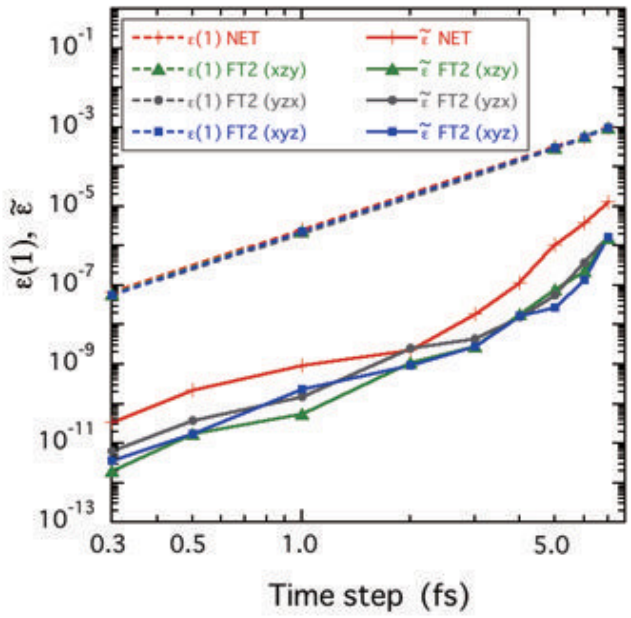

Figure 2.2: Local errors, $\epsilon(1)$, and global errors, $\tilde{\epsilon}$, in the water droplet simulation at $T = 300$ K in the FT2 and NET algorithms for various values of Δt.

opposite to Eqs. (2.40a)-(2.40c). In the present setting, the magnitudes of the principal moment of inertia are $I_x > I_z > I_y$.

Figure 2.2 shows the local error, $\epsilon(1)$, and the global error, $\tilde{\epsilon}$, of the simulations with the FT2 and NET algorithms for various Δt [33]. We find in Fig. 2.2 that the three global errors, $\tilde{\epsilon}(xyz)$, $\tilde{\epsilon}(yzx)$, and $\tilde{\epsilon}(xzy)$, in the FT2 algorithm are only about 10% of that in the NET algorithm.

Figure 2.1 also shows that the global errors of the FT1 and FT2 (we compare here especially FT2(xzy)) algorithms. The global errors are almost the same between the two algorithms and are inferior to the symplectic algorithm by about an order of magnitude.

2.4.3 Computation timings of FT algorithms

In this subsection, we discuss computation timings of various time-reversible algorithms of rotational motion that use quaternion to represent angular position. To compare computation timings of such algorithms, we compare the required timings of the parts that update quaternion and angular velocity of a rigid molecule. We assume the remaining part is almost same in every algorithm using quaternion representation.

Table 2.1 shows computation timings averaged over 10^6 measurements and numbers of operations [34] of the FT1, FT2, symplectic, Matubayasi-Nakahara, and NET algorithms required for updating angular velocity and quaternion. The number of iteration in the NET algorithm is set to one, which is not realistic, but gives us the lower limit of its computation timings. Computation timings are measured on 3.1GHz Intel Core i7, 2.5GHz Fujitsu SPARC64VII, and 3.0GHz Intel Xeon E5472. The FT2 algorithm requires less than 100 operations and all of the operations are four basic arithmetic operations except for one square root operation.

As is seen in Table 2.1, computation timings depend on machines and the order can be reversed. However, we see that the computation timing for updating angular velocity and quaternion of the FT2 algorithm is no more than that of the FT1, symplectic, Matubayasi-Nakahara, and NET algorithms. Thus assuming the remaining part of algorithm is almost same for every time-reversible one, we think that the FT2 algorithm is a fast algorithm in the time-reversible ones using quaternion representation.

2.5 Summary and concluding remarks

I have proposed the Fast Time-reversible (FT1 and FT2) simulation algorithms for rigid molecules. The stability of the FT algorithms for various values of the time step is compared with that of the NET and symplectic algorithms, through demonstrative simulation of a water droplet composed of 499 molecules at 300 K.

The global errors of the FT algorithms are only about 10% of that of

Table 2.1: Computation timings averaged over 10^6 measurements required to update angular velocity and angular position on various machines for various algorithms.

	Computation timing (10^{-8} sec.)			Number of operations		
algorithm	Core i7[a]	SPARC[b]	Xeon[c]	four rules[d]	trigonometric[e]	square root
FT1	7.2	20	9.5	154	0	1
FT2	5.7	19	8.8	82	0	1
Symplectic	21	52	29	274	10	0
MN[f]	11	65	15	166	18	1
NET[g]	7.5	26	12	169	0	1

[a] 3.1GHz Intel Core i7 950 with Intel Fortran compiler Ver. 12.0, option="-fast"

[b] 2.5GHz Fujitsu SPARC64VII with Fujitsu Fortran compiler, option="-Kfast"

[c] 3.0GHz Intel Xeon E5472 with Intel Fortran compiler Ver. 11.1, option="-fast"

[d] the four rules of arithmetic, i.e., addition, subtraction, multiplication (including exponentiation), division

[e] trigonometric functions

[f] Matubayasi-Nakahara algorithm [22]

[g] NET algorithm with one time iteration [12]

the NET algorithm. I think that the smallness is caused by various factors including the elimination of accumulation error in the iteration. The local and global errors of the FT algorithms are almost the same as that of Matubayasi-Nakahara [22] algorithm.

It may seem that the FT algorithms are similar to Matubayasi-Nakahara algorithm. However, there are differences between the two algorithms. The main difference lies in the choice of equations to be modified in order to get the updated angular velocity. To get the updated angular velocity, Matubayasi and Nakahara modified *differential equations* for the angular velocity, and got several differential equations. They integrated them one by one, determined the angular velocity at the midstep $t + \frac{1}{2}\Delta t$, integrated these equations in reverse order, and then determined the angular velocity at time $t + \Delta t$. The error involved in the algorithm is Δt^3. In the FT algorithms, on the other hand, we have introduced temporally (algebraic) equations by which we update the quaternion within the error of Δt^3, and have modified these *equations* within Δt^3. Then, we have solved these equations conversely to get the angular velocity at time $t + \Delta t$.

Noteworthy features of the FT algorithms are the following: (i) The FT2 algorithm will be a fast time-reversible algorithm for rotational mo-

tion comparable to other fast time-reversible ones using quaternion. (ii) From the viewpoint of total energy conservation, the FT algorithms are superior to the NET algorithm, and are comparable to the Matubayasi-Nakahara algorithm. (iii) Even if the equations of dynamics involve the friction term or an external force field, the FT algorithms are useful by interpreting the force appropriately for negative time step $-\Delta t$. (iv) The FT2 algorithm satisfies Eq. (2.44) as the Matubayasi-Nakahara algorithm does.

Finally, I comment on the stability of the symplectic algorithm. The global errors in the symplectic algorithm increase significantly for $\Delta t \geq$ 6 fs approaching to the FT1 results as seen in Fig. 2.1. I do not know the reason of this, but I think that it is because the trajectory of the conserved quantity $\tilde{H}$ of Miller's algorithm assured by the symplectic method may not be contained in an restricted area of (q, p) phase space or may not converge for some large Δt. For example, let us consider a simple case, a one-dimensional harmonic oscillator whose hamiltonian is given by $H = \frac{1}{2}(p^2+q^2)$. Then, $q\prime = q+\tau p$, $p\prime = p-\tau q\prime$ gives a symplectic transformation and conserves $\tilde{H} = \frac{1}{2}(p^2 + q^2) + \frac{1}{2}\tau pq$. If τ is sufficiently small, say, τ=1, $\tilde{H}$=const. means that p and q are on an ellipse close to H. On the other hand, if τ is large, $\tilde{H}$=const. is not an ellipse but a hyperbola (if $\tau > 2$). In this case, '$\tilde{H}$ is conserved' those not mean that '$H - \tilde{H}$ is small'. Of course $H - \tilde{H} = \frac{1}{2}\tau pq$ and τ is fixed, however, on the hyperbola, pq can be infinitely large (the asymptotic lines are not x- and y-axes). For example, let const.=$\frac{1}{2}$ and $\tau = 4$. Then, $\tilde{H}$=const. is equivalent to $p^2+q^2+4pq = 1$, and the discriminant of the quadratic equation for p is $D = 4q^2-(q^2-1) = 3q^2 + 4 > 0$. Thus we can find for arbitrary large q a solution p, q of the equation $\tilde{H}$=const., which implies that $H = \frac{1}{2}(p^2 + q^2) > \frac{1}{2}q^2$ can be infinitely large.

In addition, symplectic algorithm theory assures that there is a formal power series that is conserved. However, as far as I know, the convergence of the series is not proved. If $\tilde{H}$ does not converge, the $\tilde{H}$ conservation is meaningless. Therefore, even in symplectic algorithm, there may be time

steps that the convergence of total energy fail.

2.6 Addendum - modified FT algorithm

Just below the Eq.(2.16), I have stated that $|\vec{V}| < 1$ for usual time step. As shown in the reference [31], $|\vec{V}| < 1$ is always satisfied in normal situations. However, if there is a danger of $|\vec{V}| > 1$, we can take the following alternative (modified FT algorithm): Replace all the diagonal elements $\sqrt{1-|\vec{v}|^2}$ in the definition of $\overleftrightarrow{R}[\vec{v}]$ in Eq.(2.18) by $1-\frac{1}{2}|\vec{v}|^2$, and then, normalize the quaternion obtained by Eq.(2.23) every step. This modified algorithm removes the danger of computing square root of a negative number at the expense of increase of some additional operations. It can be shown that the so modified FT algorithm is time-reversible, and the stability is no less than original FT algorithm. Since the condition $|\vec{V}| < 1$ holds in usual temperature, I have presented the simpler method in [13]. However, if there is a danger of $|\vec{V}| > 1$ (i.e., in such cases that some particles may become extremely high temperature), it may be better to employ this modified FT algorithm to avoid computer crash.

The proof of the validity of this modification can be found in Appendix C.

Chapter 3

Making ice-Ih

3.1 Introduction

The purpose of this chapter is to describe the construction of the configurations of ice-Ih (=configurations of hydrogen atoms), especially for the first (bi)layer. The constructions of the layers below the first layer are given in Appendix A. Although the main difficulties to determine configurations of ice-Ih lie in determining the lower layers, the explanations given in this chapter will be helpful to understand the method of my construction of the algorithm.

There are already known some algorithms to generate initial configurations of ice-Ih without dislocation [35–38]. These methods, roughly speaking, give some examples of rectangle shaped configurations of hydrogen atoms of the first and second layers that can be connected to each other both horizontally, vertically, and ahead, by which we can make arbitrary large configurations. (There are some modified method: In [35, 36], the Monte Carlo method is employed.) Although the rectangle shaped configurations are random to some extent, the configurations given in the above works have connectable properties. These methods are simple and easy to use, however, it is difficult to make configurations of ices with dislocation by these methods as they are. Since actual ices are considered to contain dislocations and the dislocations will probably play important roles in surface melting [7, 8], it is desirable to prepare an algorithm that can make

configurations of ices with dislocation. Therefore, I devised an versatile algorithm to make configurations of ices with/without dislocation. The construction method is not so difficult, but I could not find similar methods like mine. I give in this chapter and in Appendix A the construction method of ice-Ih.

3.2 Basics

In this section, we recall some basics well-known on ice-Ih following [32,39].

3.2.1 Positions of the oxygen atoms

The basic structural features of ordinary hexagonal ice-Ih are well established. Every oxygen atom is at the center of a tetrahedron formed by four oxygen atoms each about 2.76 Å away. Every water molecule is hydrogen-bonded to its four nearest neighbors: its O-H bonds are directed towards lone-pairs of electrons on two of these neighbors, forming two O-H$\cdots$O hydrogen bonds; in turn, each of its lone-pairs is directed towards an O-H bond of one of the other neighbors, forming two O$\cdots$H-O hydrogen bonds. It can be seen from Fig. 3.1 that the lattice consists of puckered layers perpendicular to the c-crystal axis, containing hexagonal rings of water molecules that have the conformation of the 'chair' form of cyclohexane.

3.2.2 Positions of the hydrogen atoms

The essential feature of ice model is that there is no long-range order in the orientations of the H_2O molecules or of the hydrogen bonds. The disorder in the three dimensional structure is shown in Fig. 3.1. There are two possible hydrogen sites on each bond and four of these sites adjacent to each oxygen. The disorder of the hydrogens over these sites satisfies the two *ice rules* (Bernal-Fowler's rule [15]), which are:

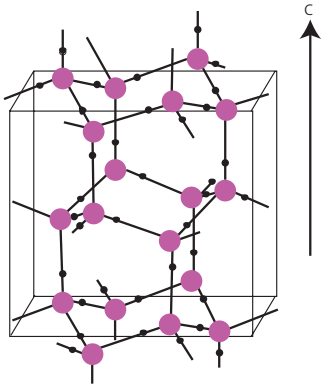

Figure 3.1: Structure of ice-Ih: Red spheres indicate oxygen atoms and small black ones hydrogen atoms.

1. There are two hydrogens adjacent to each oxygen.
2. There is only one hydrogen per bond.

In the following, we consider an algorithm that determines the configurations of ice-Ih randomly within the limits of the Bernal-Fowler's rule.

3.3 Construction

3.3.1 The numbering of water molecules

In order to construct configurations of hydrogen atoms, we begin with numbering each water molecule in an ice. We assign each water molecule (or, equivalently, oxygen atom) three non-negative integers as shown in Fig. 3.2. The third elements of the triplets of water molecules on the top puckered horizontal (bi)layer are set to zero, and those of the k-th horizontal layer are $k-1$, i.e., the assigned triplets of water molecules are of the form $(\ ,\ ,k-1)$. The first and second elements of water molecules on each layer are also determined as in Fig. 3.2. Figure. 3.3 indicates the first and second values which helps us understand the way of the numbering.

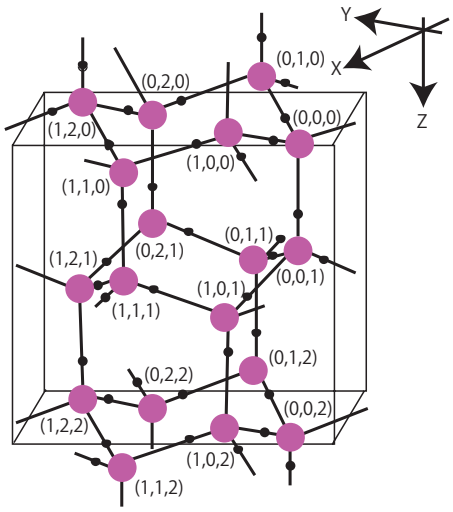

Figure 3.2: The numbering of the water molecules: z-axis is parallel to the c-axis.

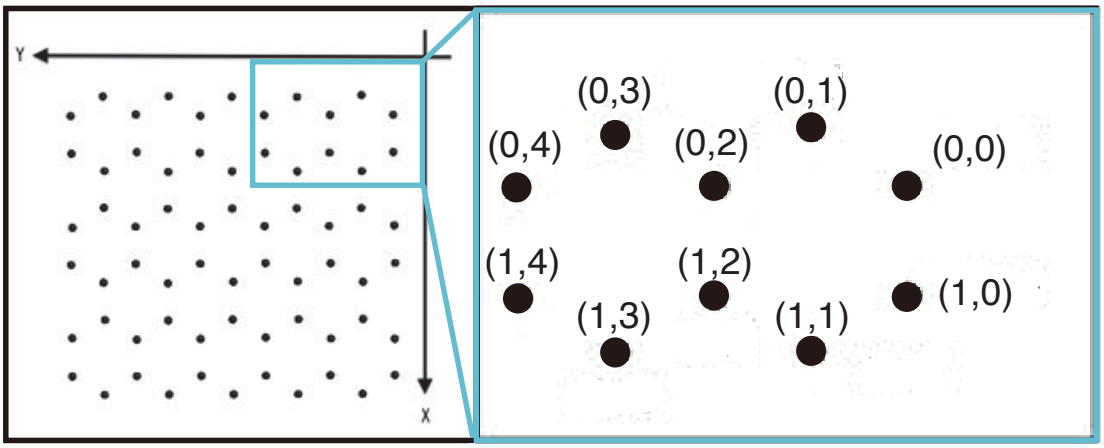

Figure 3.3: The numbering of the first and second values of an ice as seen from $z = -\infty$.

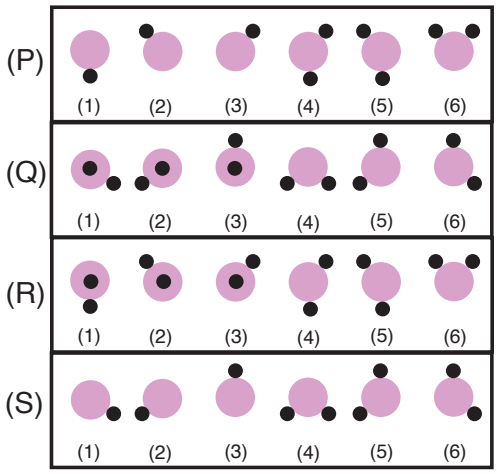

Figure 3.4: Possible directions of hydrogen atoms for the four types of oxygen atoms seen from $z = -\infty$. Here, for example, for the case (1) of (S), one hydrogen atom is just below the oxygen.

3.3.2 The specifications of the directions of water molecules

In ice-Ih, each oxygen atom is classified into four types with respect to the possibility of direction of hydrogen atoms belonging to it. For example, $(1, 0, 0)$ water molecule and $(0, 1, 0)$ water molecule have the same possible directions (here we ignore the configurations of the nearest four water molecules). Figure 3.4 shows the possible directions of the four types of oxygen atoms as seen from $z = -\infty$ (top view). Some water molecules seem to have only one hydrogen atom, say (1) of (P). The hidden hydrogen atom is just below the oxygen atom and thus we cannot see it from above .

Let (m, n, l) be the triplet assigned to an oxygen atom. Then the four figures (P), (Q), (R), and (S) in Fig. 3.4 correspond to the four types of oxygen atoms in the following manner:
(P) gives the possible directions for l =even, $m + n$=even,
(Q) gives the possible directions for l =even, $m + n$=odd,
(R) gives the possible directions for l =odd, $m + n$=even,
(S) gives the possible directions for l =odd, $m + n$=odd,

respectively.

If a water molecule is assigned to (m, n, l), then we denote by $rot(m, n, l)$ the number of its direction determined by Fig. 3.4. For example, $rot(1, 2, 3) = 2$ means that the direction of the molecule assigned to (1,2,3) is given by (2) of (S). Hereafter, we use $rot(m, n, l)$ to indicate the directions of water molecules via Fig. 3.4.

3.3.3 Construction of ice-Ih of the first layer ($l = 0$)

To construct the first layer of ice-Ih, we proceed as follows:
(1): Determine the positions of two hydrogen atoms belonging to the (0,0,0) oxygen, that is to say, determine $rot(0, 0, 0)$.
(2): Determine $rot(0, n, 0)$ $n \geq 1$ successively.
(3): Determine $rot(m, 0, 0)$ $m \geq 1$ successively.

Here we have determined all directions of water molecules on the x- and y-axes (= on the wavy lines close to the x- and y-axes).
(4): 'Determine $rot(m, n, 0)$ successively for $n \geq 1$' successively for $m \geq 1$.

Specifically, the construction above is carried out as follows:

(1): $rot(0, 0, 0)$

$$rot(0, 0, 0) = \{1, 2, 3, 4, 5, 6\}.$$

The above expression implies that we select a number from {1,2,3,4,5,6}, and define $rot(0, 0, 0)$ by the selected number.

(2): $rot(0, n, 0)$
(2-1) If n=even; then

$$rot(0, n, 0) = \begin{cases} 1, 2, 5 \ (rot(0, n-1, 0) = 2, 4, 5) \\ 3, 4, 6 \ (rot(0, n-1, 0) = 1, 3, 6) \end{cases}$$

Here, this means that **if** $rot(0, n-1, 0) =$**2,4, or 5, then we determine the direction** $rot(0, n, 0)$ **by selecting a number from {1,2,5}.** In the following, we use the similar notation as this.

This and the followings are obtained by observing Fig. 3.2.
(2-2) If n=odd; then

$$rot(0,n,0) = \begin{cases} 2,3,5\ (rot(0,n-1,0) = 2,5,6) \\ 1,4,6\ (rot(0,n-1,0) = 1,3,4) \end{cases}$$

(3): $rot(m,0,0)$
(3-1) If m=even; then

$$rot(m,0,0) = \{1,2,3,4,5,6\}.$$

(3-2) If m=odd; then

$$rot(m,0,0) = \begin{cases} 1,2,4\ (rot(m-1,0,0) = 1,4,5) \\ 3,5,6\ (rot(m-1,0,0) = 2,3,6) \end{cases}$$

(4): $rot(m,n,0)$
(4-1) If $m+n$=even; then

$$rot(m,n,0) = \begin{cases} 3,4,6\ (rot(m,n-1,0) = 1,3,6) \\ 1,2,5\ (rot(m,n-1,0) = 2,4,5) \end{cases}$$

(4-2) If $m+n$=odd; then

$$rot(m,n,0) = \begin{cases} 6 \begin{cases} rot(m-1,n,0) = 2,3,6 \\ rot(m,n-1,0) = 1,3,4 \end{cases} \\ 3,5 \begin{cases} rot(m-1,n,0) = 2,3,6 \\ rot(m,n-1,0) = 2,5,6 \end{cases} \\ 1,4 \begin{cases} rot(m-1,n,0) = 1,4,5 \\ rot(m,n-1,0) = 1,3,4 \end{cases} \\ 2 \begin{cases} rot(m-1,n,0) = 1,4,5 \\ rot(m,n-1,0) = 2,5,6 \end{cases} \end{cases}$$

In (4), we determine $rot(m, n, 0)$ in the way as follows:

do $m = 1$, $width_x$
  do $n = 1$, $width_y$
    determine $rot(m, n, 0)$
  end do
end do

where $width_x$ and $width_y$ are widths of ice of interest in x-direction and y-direction, respectively. In this chapter and Appendix A, in constructing the configurations of inner ice (namely, (m, n, l) $m \geq 1$ or $n \geq 1$), we proceed as in the way as above: That is, we always determine $rot(m, n, l)$ for a given m in the direction parallel to the y-axis from n=1 to the $width_y$, and m runs through from 1 to $width_x$.

The above conditions in (4-2) mean, for example, that if $rot(m-1, n, 0)$=2,3,6 and $rot(m, n-1, 0)$=1,3,4, then $rot(m, n, 0)$=6, and if $rot(m-1, n, 0)$=2,3,6 and $rot(m, n-1, 0)$=2,5,6, then $rot(m, n, 0)$=3 or 5. In the later case, we select a number from {3,5}.

We can now determine the first layer of ice-Ih. Figure 3.5 is the example made by the method above.

For the layers below the first layer, I have summarized the method in the Appendix A. It can be understood by observing Fig. 3.1.

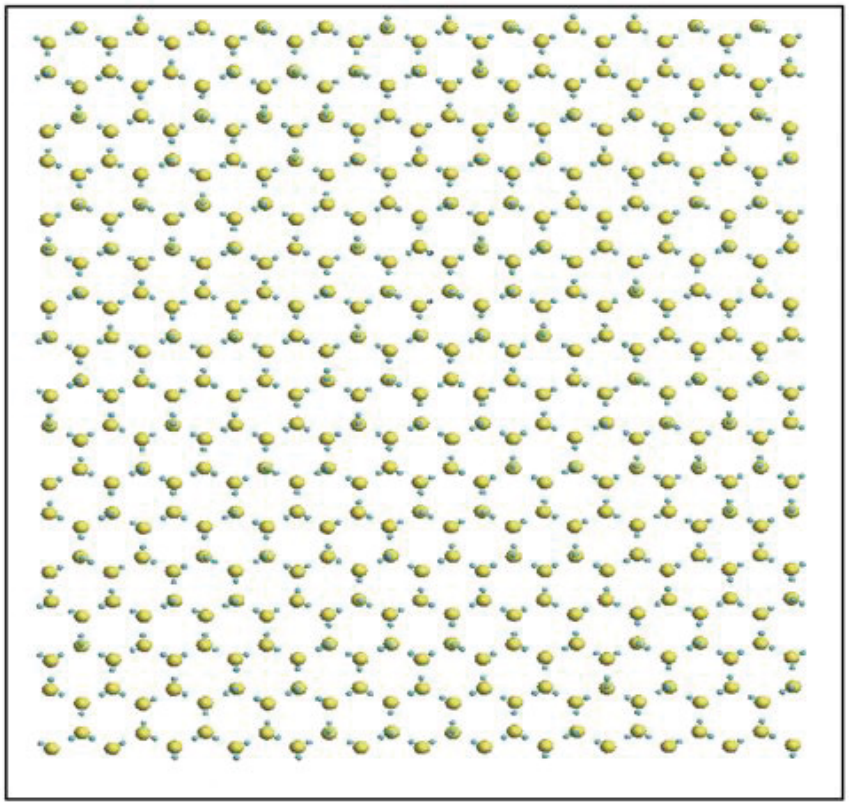

Figure 3.5: The first layer of ice-Ih given by the method presented in this chapter.

Chapter 4

Simulations of QLL

4.1 Introduction

It is well known that the surface of ice melts below the bulk melting temperature to form a quasi-liquid layer (QLL) [1] since it was first predicted by Faraday [40]. The QLL of ice is considered to play very important roles in many aspects, e.g., in the skating mechanics while neither pressure melting nor frictional heating can explain slipperiness [2] and in the acid snow formation by dissolving acidic substances in the QLL [3]. Hence many experimental and theoretical studies have been carried out to investigate the properties of the QLLs (e.g., [41–43]). Detailed theoretical studies of the QLL using the molecular dynamics (MD) simulation method were reported in the literature [4–6]. However, relatively small slab systems (the numbers of molecules are at most 2,130) under periodic boundary conditions were used in those simulations. There should be phenomena that can be observed only by large-scale simulation. For instance, it has been reported recently through experiments that there are two different morphologies in QLLs with super-micrometer scales on ice [7, 8], which cannot be reproduced by simulation with a small simulation box. The purpose of this chapter is to seek possible dynamics in QLLs of ices through large-scale MD simulation and investigate the properties of the QLLs.

In this chapter I perform the MD simulation runs of sub-micrometer-scale ice-Ih in a vacuum at various temperatures to unveil the characteris-

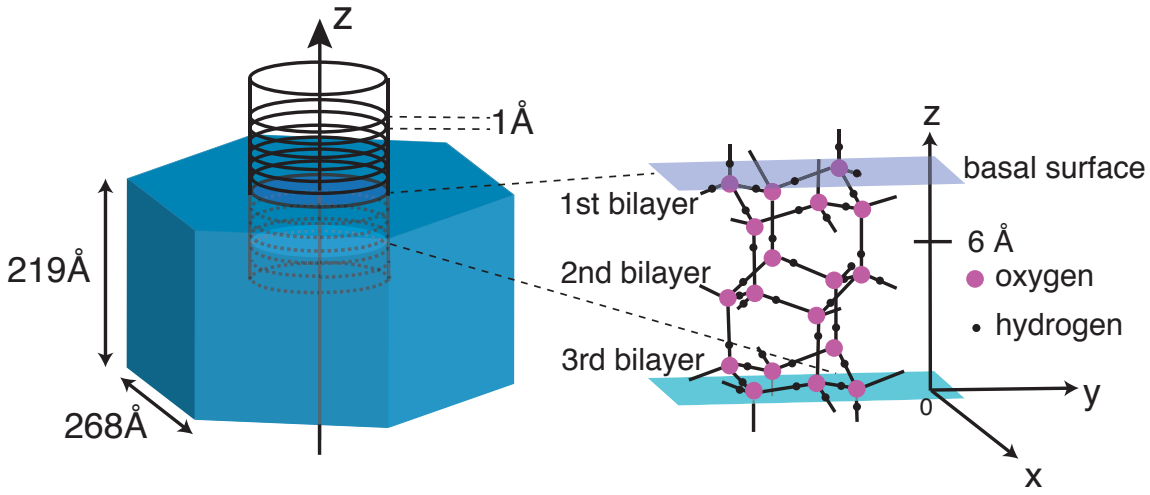

Figure 4.1: Ice-Ih crystal in a hexagonal prism shape composed of 1,317,600 H_2O molecules (61 bilayers $\times$ 21,600 molecules). The z-axis is perpendicular to the basal (0001) surface; x and y denote the $[10\bar{1}0]$ and $[\bar{1}2\bar{1}0]$ axes, respectively. We set $z = 0$ at the bottom of the third bilayer from the outside. The radius of the virtual cylinder for analyses is 114 Å.

tics of the QLLs formed on a (0001) surface.

4.2 Method

4.2.1 System

The initial configuration of H_2O molecules in the present simulation is prepared as follows. We first set the O positions of the ice-Ih with a hexagonal prism shape, as depicted in Fig. 4.1. Second, we determine the H positions randomly following Bernal-Fowler rules [15] as stated in Chapter 3, that is, there exists a single H per O-O bond and each O has just two adjacent H's, to realize a nearly zero macroscopic Coulomb dipole. The z-axis is set perpendicular to the basal (0001) surface of the ice. The present system is composed of 1,317,600 molecules.

4.2.2 Simulation program

We reduce the computation cost by regarding an H_2O molecule as a rigid body and by employing the TIP4P intermolecular interaction potential [9]. The bulk melting temperature $T_m = 228$ K [16] for the TIP4P potential deviates considerably from the experimental value of 273 K; however, as I stated in the previous chapter, the potential provides a qualitatively correct description of the phase diagram of ice [18] and is also widely used for simulations of the melting of ice, the nucleation of a droplet, and the freezing of water [19,20]. We adopt the Fast Time-reversible algorithm [13] for the time integration of the rotational motion of molecules and the Velocity-Verlet algorithm for the calculation of translational motion.

I also installed the parallelized FMM program coded by Prof. Ogata [11] on my simulation program by modifying slightly. As is well-known, the Fast Multipole Method (FMM) was first established by Greengard and Rokhlin [10]. FMM reduces the N-body problem to an order $O(N)$. Multipole expansion represents potential energy by expansion with respect to Legendre's associated function. The expansions are in themselves known before Greengard and Rokhlin, however, they introduced local expansions induced by multipole expansion and summed up the coefficients of multipole/local expansions for a given order in an ingenious manner to reduce the calculation of the total potential energy caused by particles each other farther than a (conveniently given) distance d to an order $O(N)$. The force acting on an atom is calculated by differentiating the potential energy with respect to the position of the atom. The higher the order of multipoles, the accuracy and the computation cost increase. The potential energy and the force caused by near particles (within the given distance d) is calculated directly, which is obviously $O(N)$. In this way, the N-body problem is reduced to $O(N)$.

In my simulation program, the maximum order of multipoles is set to six and the Coulomb interactions are calculated directly between charged points within at least 8.9 Å. The Lennard-Jones potential part in the TIP4P is truncated continuously. The system is decomposed spatially

to 256, 512, or 1024 compute-nodes for parallel machines. The time step is 4.0 fs and the Lennard-Jones potential part in the TIP4P is truncated continuously at 7.9 Å. The system temperature is kept to a given value by simply multiplying an appropriate number to both translational and angular velocities every 1,000 steps.

4.3 Results

4.3.1 Settings

In the runs, the temperatures are controlled to $T = 205$, 215, and 227 K. The temperatures correspond to $T_{\rm m}-23$, $T_{\rm m}-13$, and $T_{\rm m}-1$ K; that is, the reduced undercooling $(T - T_{\rm m})/T_{\rm m} = -10.0$, -5.6, and -0.4%. Regarding the depth of the QLL, no substantial differences exist between a variety of potential models at the same reduced undercooling [44]. It suggests that if we regard $T_{\rm m} - 1$ K=272 K, it makes almost no difference. At each temperature, the system is thermalized for more than 1.0 ns. After the thermalization, the simulation runs are performed for 3.0, 4.0, and 5.0 ns at 205, 215, and 227 K. We denote the final time as $t_{\rm final}$ for all the runs.

I confirm that the QLL depths in the virtual cylindrical region, illustrated in Fig. 4.1, are almost stable after $t_{\rm final} - 2.0$ ns. Here, the depth is measured using the tetrahedral order parameter q_i for oxygen-i defined by

$$q_i = \left[1 - \frac{3}{8} \sum_{j=1}^{3} \sum_{k=j+1}^{4} \left(\cos(\theta_{i,j,k}) + \frac{1}{3} \right)^2 \right],$$

where i, j, and k are indices for the O atoms. The angle $\theta_{i,j,k}$ is formed by two of the four nearest O atoms O-j and O-k of O-i. This parameter is defined so that it is unity when four nearest-neighbor O's of the O-i form a regular tetrahedron [44, 45]. Each water molecule is regarded as icelike if $q > 0.91$, as was done in Fig. 1 of [44]. In the literature, M. M. Conde et al. investigated the thickness of QLLs and discussed on the parameter q to show that the parameter works well to classify water

molecules as being either icelike or liquidlike. I got in my simulation that the depths at 205, 215, and 227 K measured by the parameter are 4.8, 6.3, and 9.4 Å, respectively. They agree reasonably well with the results shown in Fig. 7 of [44]. I do not use the total energy to judge whether the simulation period is sufficient for analyses because the edges of the present ice continue melting for a longer period, as observed in [45]. (Hexagonal ice with 9600 water molecules at 215 K continued melting for about 30 ns.) I note that the numbers of molecules in the virtual cylindrical region (see Fig. 1) remain almost unchanged throughout the runs and the thicknesses are almost the same as above after $t_{\rm final}$-2.0 ns.

4.3.2 Bumpiness of QLL

Figure 4.2 depicts the top views of QLLs formed on the (0001) surface of the ice at $t_{\rm final}$ at (a) 205, (b) 215, and (c) 227 K. The molecules are colored according to the z-positions, where $z = 0$ corresponds to the bottom of the third bilayer (see Fig. 4.1). We find in Fig. 4.2 that the QLLs in all the runs are bumpy, that is, composed of bumps and trenches, and that both the vertical and horizontal sizes of the bumps (i.e., red regions around the center) increase significantly with the rise of temperature. The average heights of the liquid bumps reach $\sim$ 9, $\sim$ 10, and $\sim$ 12 Å at 205, 215, and 227 K, respectively. The prism surfaces, which we do not discuss here, are also bumpy. Figures 2(d) and 2(e) depict the time evolution of liquid bumps at 227 K at (d) 5.0 and (e) 4.5 ns. We therein find that the locations of liquid bumps change significantly after 0.5 ns. Similar time evolutions of the liquid bumps are also observed at 205 and 215 K. The behaviors of bumps and trenches are almost stable for time $t > t_{\rm final} - 2.0$ ns.

4.3.3 Features of the bumpy QLL

We now investigate the z-dependences of the structure and dynamics of the QLLs. To this end, we define a virtual slice $S(z)$ with its center located at $z = \{-5, -4, \cdots, 15\ \mathring{A}\}$ as depicted in Fig. 4.1. Figures 4.3(a), 4.3(c), and 4.3(e) show the normalized densities of the molecules in $S(z)$ at all the

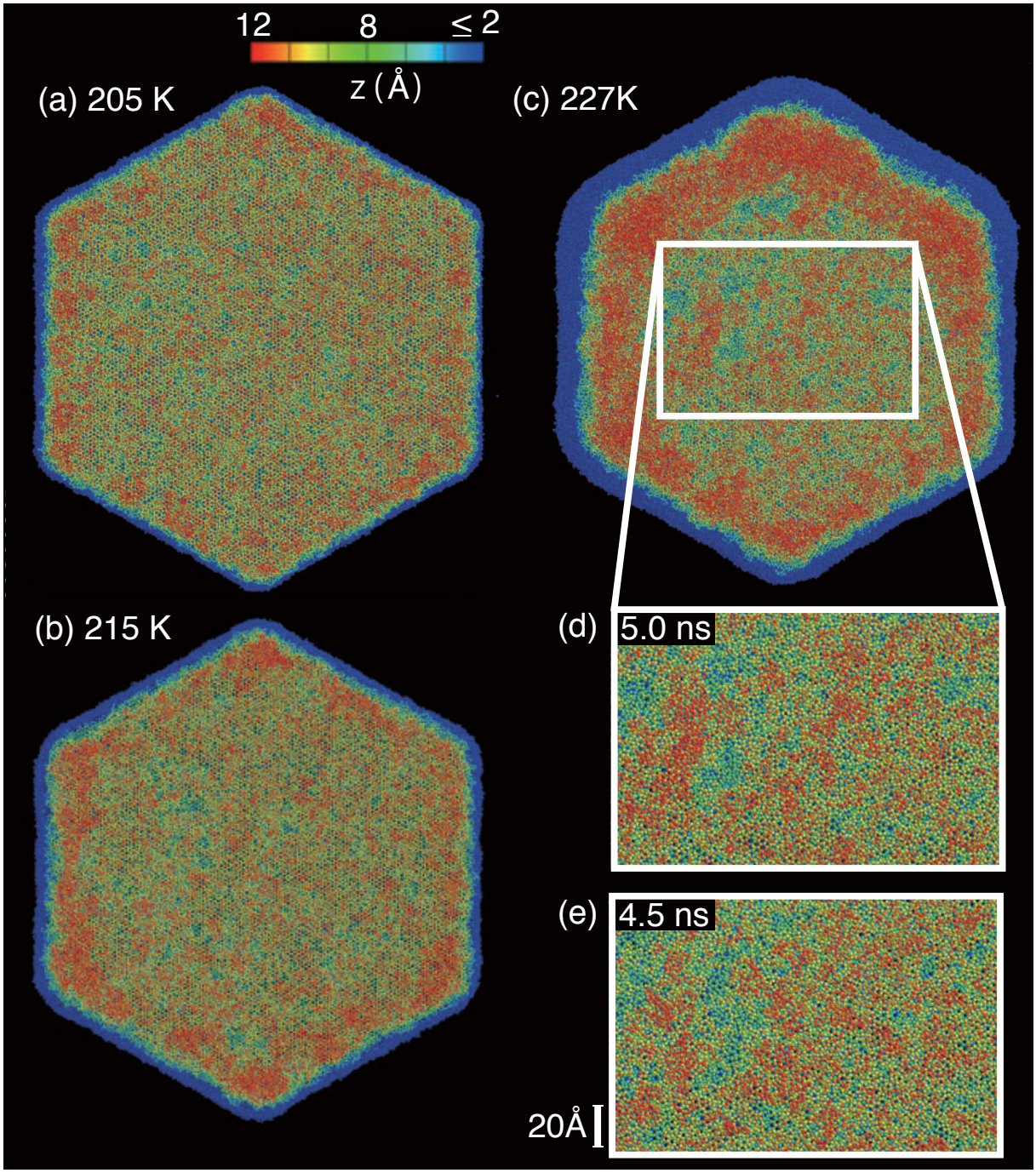

Figure 4.2: $x - y$ views of (0001) surfaces of the ice at time t_{final} at 205, 215, and 227 K, and the time evolution of the surface at 227 K. The molecules are colored according to the z-positions.

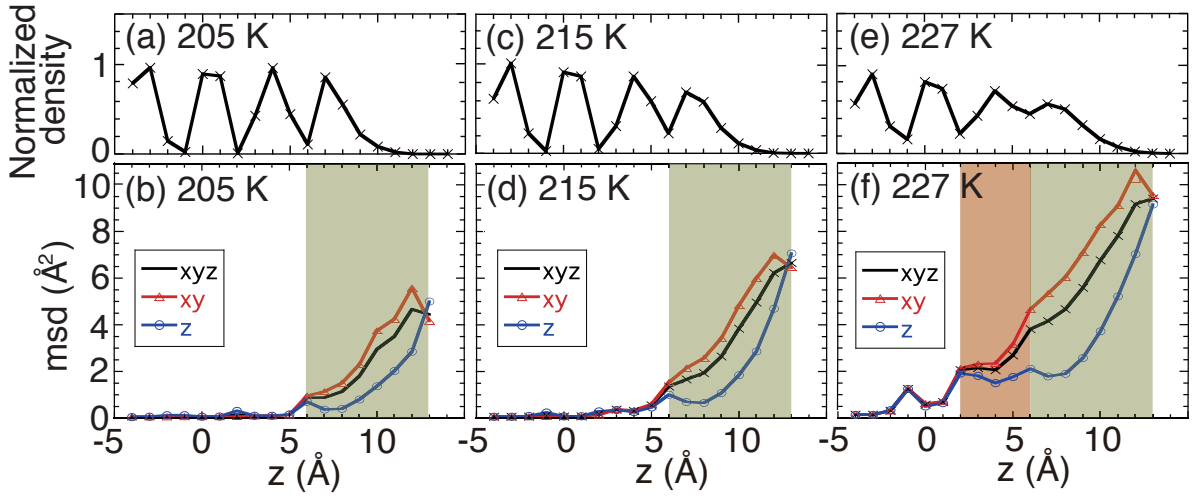

Figure 4.3: Normalized densities of H_2O molecules in the virtual slices of the ice (see Fig. 1) at (a) 205, (c) 215, and (e) at 227 K. Mean-squared displacements (Eq. 1) of O's during 0.1 ns in the virtual slices at (b) 205, (d) 215, and (f) at 227 K. The inequality $D_{xy} \gg D_z$ holds in the green-hatched z-range; $D_{xy} \approx D_z$ in the red-hatched z-range.

temperatures. Since the density assumes distinct minima at $z = 6.0$, 2.0, and -1.5 Å at all three temperatures, we decompose the QLL into the following three z-regions: $\mathrm{QLL}_{6.0\ \mathring{A}<z}$, $\mathrm{QLL}_{2.0<z<6.0\ \mathring{A}}$, and $\mathrm{QLL}_{-1.5\ \mathring{A}<z<2.0\ \mathring{A}}$, which correspond respectively to the first, second, and third bilayers of the original ice. We call the regions QLL because they are melted locally, at least, as we will see below.

Figures 4.3(b), 4.3(d), and 4.3(f) show the the mean-squared displacements (MSDs) of O's in $S(z)$ during 0.1 ns defined as

$$\begin{aligned} D_\alpha(z) &\equiv \frac{1}{2}\left\langle (\mathrm{O}_{i,\alpha}(t+0.1\mathrm{ns}) - \mathrm{O}_{i,\alpha}(t))^2 \right\rangle , \\ &\text{where } \mathrm{O}_{i,\alpha}(t) \in S(z),\ \alpha = \{x,y,z\} \\ D_{xy}(z) &\equiv \frac{1}{2}\left(D_x(z) + D_y(z)\right), \\ D_{xyz}(z) &\equiv \frac{1}{3}\left(D_x(z) + D_y(z) + D_z(z)\right), \end{aligned} \tag{4.1}$$

where $\mathrm{O}_{i,\alpha}(t)$ is the position of O-i at time t in the α-coordinate. The aver-

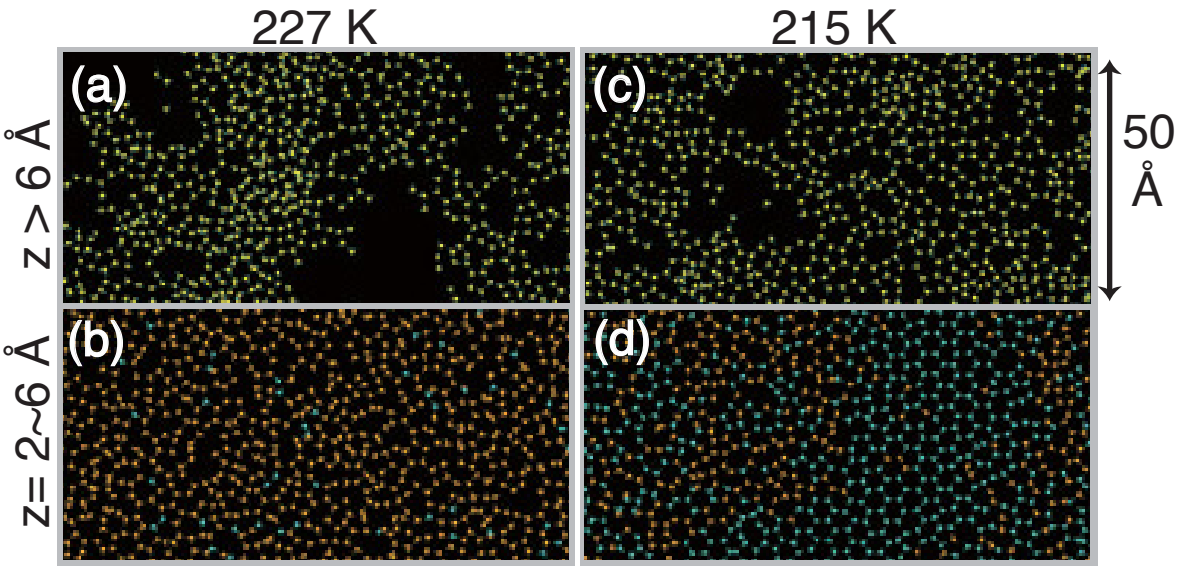

Figure 4.4: x-y views of molecules in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ and $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ at time t_{final}: (a) and (b) at 227 K; (c) and (d) at 215 K. In (b) and (d), each molecule is colored red if its displacement from $t_{\mathrm{final}} - 0.5$ ns to t_{final} is larger than 2.8 Å, and blue otherwise.

age $\langle \ldots \rangle$ in Eq. (4.1) is taken over all O(t)'s in $S(z)$, where t runs through $[t_{\mathrm{final}} - 0.6\ \mathrm{ns}, t_{\mathrm{final}} - 0.1\ \mathrm{ns}]$ with an interval of 8.0 ps. In Fig. 4.3, we confirm that the molecules in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ melt at all the temperatures. $D_{xy} \gg D_z$ holds for $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ at all the temperatures, which is a reflection of the bumpy nature. At 227 K, we find that the molecules in $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ are in the liquid phase and satisfy $D_{xy} \approx D_z$ [see Fig. 3(f)]. Ikeda-Fukazawa and Kawamura reported that the MSDs in the QLL satisfy $D_c > D_a$ and $D_c > D_b$ (where a, b, and c denote the $[11\bar{2}0]$, $[10\bar{1}0]$, and [0001] axes, respectively) [4], which contradicts the present results. I consider that their simulation box was so small that the molecular motion in the a- and b-directions was inhibited artificially.

4.3.4 Relationship between $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ and $\mathrm{QLL}_{6.0\ \mathring{A}<z}$

Figure 4.4 shows a comparison of the x-y configurations of molecules in $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ and in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ at 227 and 215 K. Here, each molecule in Figs. 4.4(b) and 4.4(d) is colored red if its displacement during 0.5 ns exceeds 2.8 Å ($\approx$ the O-O distance in water) and blue otherwise. We regard such red molecules as assuming the liquid phase. In Figs. 4.4(a) and 4.4(c), we observe the liquid bumps and trenches of the QLLs. As seen in Fig. 4.4(b), the molecules in $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ (i.e., the second bilayer) melt completely to form a liquid sheet at 227 K. At 205 and 215 K, on the other hand, the molecules in this region are partly melted, as seen in Fig. 4.4(d) for 215 K. In $\mathrm{QLL}_{-1.5\ \mathring{A}<z<2.0\ \mathring{A}}$ (i.e., the third bilayer), nearly complete crystalline structures are seen at 205 and 215 K, while partially melted areas are seen at 227 K.

As seen in Figs. 4.4(a) and 4.4(b), the trenches (or hollows) in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ do not affect the molecular structures in $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ at 227 K. At 215 K, on the other hand, we find in Figs. 4.4(c) and 4.4(d) that the melted areas in $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ nearly coincide with the locations of trenches in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$. This coincidence is confirmed through detailed investigations of several configurations of molecules at either 205 or 215 K. Since the horizontal locations of the trenches in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ change over time, the tendency suggests that the melted area of the second bilayer should recrystallize when it is covered with the liquid bumps.

4.3.5 Recrystallization

Let me confirm the mechanism through the time evolution of the molecular snapshots at 215 K in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ and $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$, as depicted in Fig. 4.5. I specify four x-y regions with relatively high molecular densities in Fig. 4.5(i) by ellipses 1-4. In the bottom panels of Fig. 4.5, a molecule is colored red if it resides initially in the first bilayer, while it is blue otherwise. We see in Figs. 4.5(b), 4.5(d), 4.5(f), 4.5(h), and 4.5(j) that the percentage of *crystalline* molecules coming from the first bilayer increases monotonically over time. We also find a substantial correlation

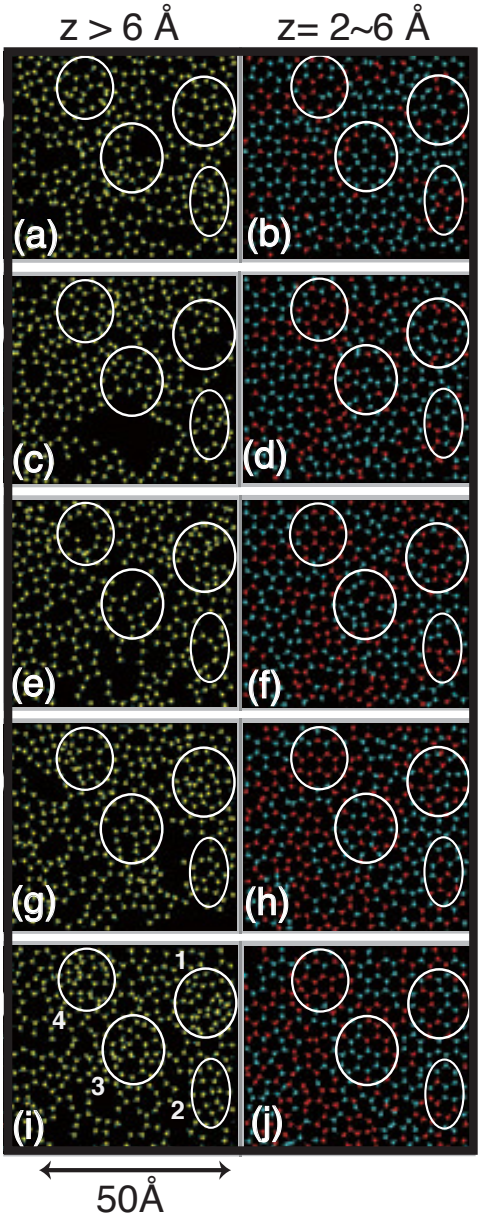

Figure 4.5: x-y views of molecules at 215 K. (a), (c), (e), (g), and (i) correspond respectively to $t_{\rm final}-2.0$ ns, $t_{\rm final}-1.5$ ns, $t_{\rm final}-1.0$ ns, $t_{\rm final}-0.5$ ns, and $t_{\rm final}$ in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$; (b), (d), (f), (h), and (j) corresponding to the same times in $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$. The molecules in (b), (d), (f), (h), and (j) are colored red if they come from the first bilayer of the original ice, and are blue otherwise. Ellipses are placed in the x-y regions with relatively high molecular densities in (i).

between the locations of the liquid bumps in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ and those of recrystallization in $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$. To see this, let me observe the molecules in region1 in Fig. 4.5(i). The molecules under them [see region1 in Fig. 4.5(j)] have crystalline order. They are in disorder in Figs. 4.5(d) and 4.5(f) and slightly disordered in Fig. 4.5(h). We find hollows in region1 in Figs. 4.5(c) and 4.5(e). I therefore state that the molecules in region1 in $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ recrystallize when they are covered with molecules of $\mathrm{QLL}_{6.0\ \mathring{A}<z}$. A similar statement applies to regions2 and 3. In region4, the molecules maintain the crystalline order in $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ [Figs. 4.5(b)-4.5(j)]. I consider that they do not fall into disorder because the molecules in region4 in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ have a high density throughout the simulation run [except for Fig. 4.5(e), where there is a small hole]. Putting these results and those shown in the preceding paragraphs (the locations of trenches change; the molecules under the trenches melt; the total number of melted molecules is almost stable) together, we conclude that a local area in the second bilayer melts easily when the trenches move on it and that the melted area recrystallizes when the liquid bumps cover it. It is observed that a larger disordered area appears to require a longer time to recrystallize under the liquid bumps. I will investigate the relation in future work.

Let me consider the possibility that the molecules in a liquid bump diffuse collectively. From Fig. 4.3, the MSD of molecules in xy-directions during 0.1 ns is 10 $\mathrm{\AA}^2$ at most, which implies the squared migration length of a molecule in xy-directions during 0.1 ns is less than $4 \times 10 = 40$ $\mathrm{\AA}^2$ (here the multiplier 4 comes from the definition of D_{xy} in Eq. 4.1). Thus the migration length is less than $\sqrt{40} \approx 6.3$ Å. Therefore a molecule moves in xy-directions during 0.5 ns by $6.3 \times \sqrt{0.5/0.1} \approx 14$Å at most. On the other hand, we find in Figs. 4.2(d) and 4.2(e) that the typical migration distances of liquid bumps (if they move in one united body) are much larger than 14 Å. Moreover, it is confirmed through x-z views of the virtual cut surfaces that the molecules in the liquid bumps, trenches, and $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ mix at both 205 and 227 K. Fig. 4.6 illustrates the side views of water molecules around the center of the ice at 227 K. I therefore state that the molecules

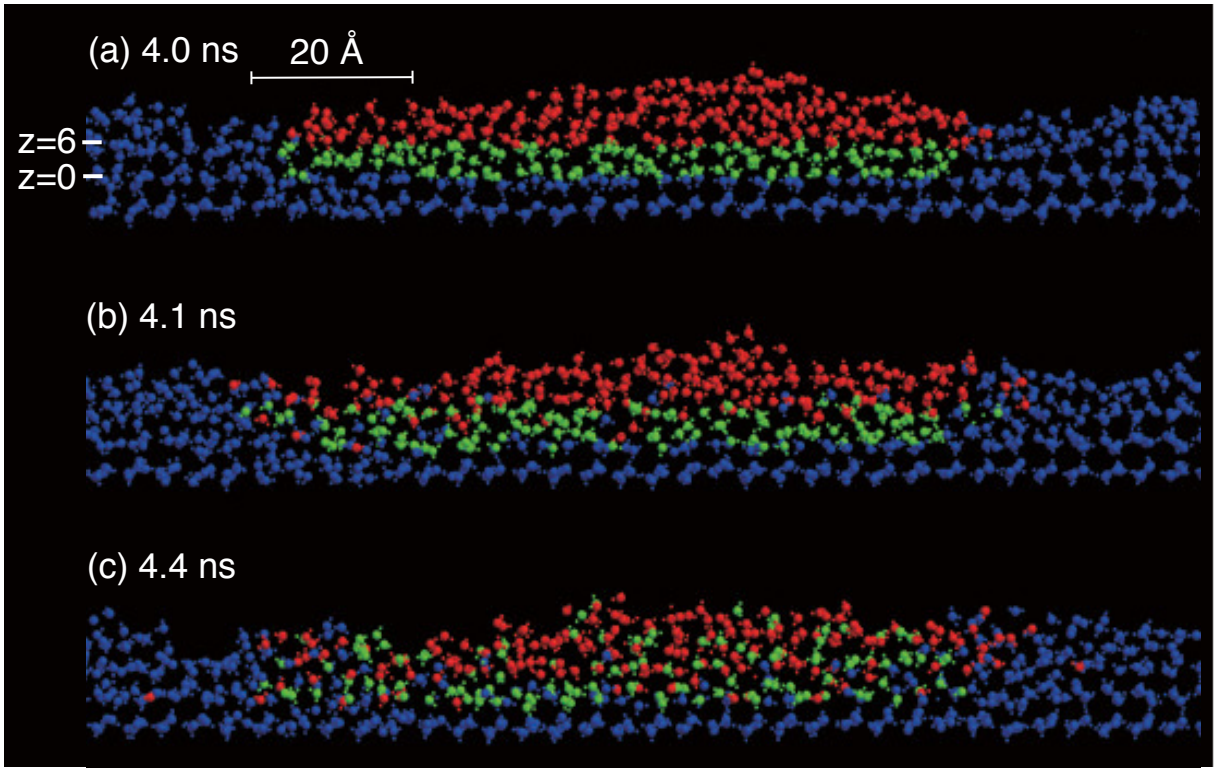

Figure 4.6: The side views of water molecules in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ and $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$ around the center of the ice with no defects at 227 K. Those molecules located in a given horizontal span at 4.0 ns are colored red for molecules in $\mathrm{QLL}_{6.0\ \mathring{A}<z}$ or green for $\mathrm{QLL}_{2.0\ \mathring{A}<z<6.0\ \mathring{A}}$. (a) at 4.0 ns, (b) at 4.1 ns, and (c) at 4.4 ns.

in a liquid bump do not diffuse collectively. The liquid bumps repeatedly form and break in a random manner at variety of places.

I have explained in a preceding paragraph that the liquid bumps and liquid sheet at 227 K differ in molecular diffusivity. In addition, I give here for reference the roughly estimated averages of O-O distances. They are about 2.82 Å for the liquid bumps and 2.80 Å for the liquid sheet. The O-O distance is 2.78 Å (2.81 Å) for ice-Ih at 227 K (bulk water at 230 K) at zero pressure in my simulation. Note that, though the distance in the ice is shorter than that in water, the ice density is less than the water density because of the orientation ordering of molecules. We find that the intermolecular distance in the liquid sheet has an intermediate value between those in the ice and water, and that the distance in the liquid bumps is larger than that in water.

I comment that the liquid bumps and liquid sheet at 227 K are respectively similar to the α- and β-phases of the QLL found on the (0001) surface in recent experiments [7, 8] of the hundreds of micrometer-scale ice. Hemispherical α-QLLs with a micrometer size emerges at a relatively wide range of temperatures, and a flat β-QLL appears under the α-QLL at temperatures slightly below the melting point. As explained above, the present simulation of the sub-micrometer-scale ice has demonstrated that a liquid sheet appears under the liquid bumps on the (0001) surface when the temperature is raised to just 1 K below the melting point. Although the liquid bumps are not round and their heights are as small as 10 Å (see Fig. 4.2), there may be some connections between the liquid bumps and the α-QLL and between the liquid sheet and the β-QLL.

4.3.6 Ices with screw dislocations

I have performed simulation runs for ices each with a screw dislocation for 3.0, 4.0, and 5.0 ns at 205, 215, and 227 K, respectively, similar to the ices with no defects explained in §4.2. Although α-QLLs always appeared from the outcrops of the screw dislocations in the experiments [7, 8], I cannot find a significant relationship between the screw dislocation and bumps

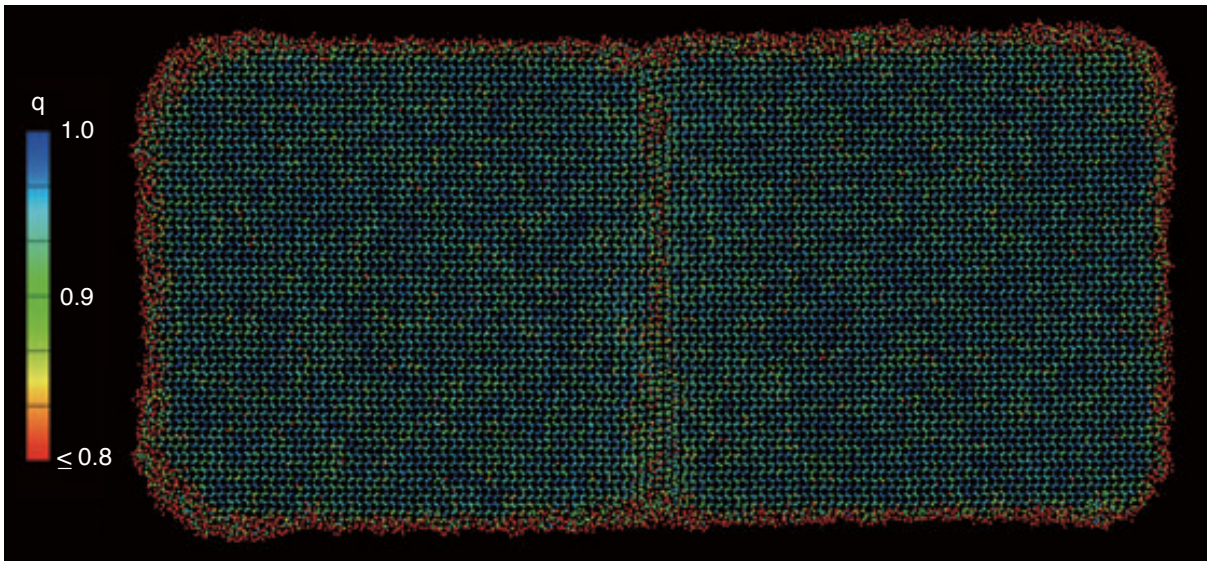

Figure 4.7: The cut view of the ice at 227 K with the screw dislocation line located on the cut plane. The water molecules are colored according to the tetrahedral order parameter q.

in the runs. In the following I mention mainly the simulation results at 227 K. Figure 4.7 shows a cut view of the ice with the dislocation line located on the cut plane. The water molecules are colored according to the tetrahedral order parameter q [18]. The differences I find between such views of the ices with and without dislocation at 227 K are summarized below (similar differences are observed with lesser extents at 205 K and 215 K):

(1) The QLL near the outcrop of the dislocation is thicker compared to that at other places. However the influence of the dislocation is limited within a radius of 25 Å from the outcrop of the dislocation on the basal surface.

(2) The ices around the dislocation line melt partly. However it is limited within a radius of 15 Å from the dislocation line.

(3) The QLL is thicker (by 2 times at most) near the steps of the screw dislocation on the basal surface.

A possible reason for the qualitative differences with respect to the effects of the screw dislocation on the formation of QLL between the present

results and experimental observations [7, 8] is the following. The present ice is put in a vacuum while the ice in the experiment is put in a supersaturated water vapor environment. The QLL in the experiment might form by vapor deposition. Another possible reason is that the present ice with a screw dislocation is much smaller than the experimental one. The concentration of external stress, that is inevitable in experiments, around the dislocation line might be significant, resulting in spouting of melted water formed around the dislocation line.

4.4 Conclusions

To summarize, I have performed the MD simulation of a sub-micrometer-scale ice-Ih crystal with no defects for 3.0, 4.0, and 5.0 ns at 205, 215, and 227 K (= −10.0, −5.6, and −0.4% lower than $T_{\rm m}$) by employing the TIP4P potential. I thereby found the following:
(i) At all the temperatures, the QLLs formed on the (0001) surface are bumpy and the liquid bumps repeatedly form and break in a random manner at a variety of places on the second bilayer. The average height of the liquid bumps increases as the temperature is raised: $\sim$ 9 Å at 205 K, $\sim$ 10 Å at 215 K, and $\sim$ 12 Å at 227 K. The diffusivity of molecules in the liquid bumps is much larger along the xy-directions than along the z-direction.
(ii) At 205 and 215 K, the local areas of the second bilayer of ice under the trenches (i.e., thin QLL areas) fluctuate to melt easily. Following the change in the locations of the liquid bumps over time, the melted areas newly covered by the liquid bumps tend to recrystallize. The third bilayer preserves the crystalline structure.
(iii) The melted areas of the second bilayer become wider with increasing temperature, and the second bilayer becomes a liquid sheet at 227 K. The liquid bumps sit on the liquid sheet.
(iv) At 227 K, the diffusivity of molecules in the liquid sheet becomes isotropic. The melted areas of the third bilayer are uncorrelated with the locations of the liquid bumps. The fourth bilayer preserves the crystalline

structure.

I could not find experimental results corresponding to (ii) above. However, I think that the recrystallization in the inner bilayer under the QLL will make environmental hydrophilic substances dissolve to be incorporated substantially in the bilayer even if the QLL is thin, and that a fine measurement about such environmental adsorption will probably confirm the recrystallization.

I have also performed simulation for an ice with a screw dislocation. Influence of the screw dislocation on the melting and formation of QLL has been limited to a nearby region around the dislocation line as follows:
(v) The QLL near the outcrop of the dislocation is thicker compared to that at other places. However the influence of the dislocation is limited within a radius of 25 Å from the outcrop of the dislocation on the basal surface.
(vi) The ices around the dislocation line melt partly. However it is limited within a radius of 15 Å from the dislocation line.
(vii) The QLL is thicker (by 2 times at most) near the steps of the screw dislocation on the basal surface.

Note

The simulations above have been performed by several super computers, especially by PRIMERGY at Research Center for Computational Science (Okazaki). The total computation time required was more than 2400 hours with 16 nodes.

Chapter 5

Summary

In this thesis, I have presented the first time-reversible (FT) algorithms and performed large-scale molecular dynamic simulation of quasi-liquid layers of ices-Ih with or without dislocation. In order to perform the simulation, I have devised a parallelized rigid body molecular dynamics computer code adopting one of the FT algorithms for the time-integration of the rotational motion. Employing this code, I have performed large-scale simulation of quasi-liquid layers of ices-Ih with or without dislocation, each put in a vacuum box by using several supercomputers. In addition, I have also presented an algorithm for making configurations of ices Ih, which can be used for making ices with dislocation as well.

Let me summarize the results obtained in this thesis as follows:

I. I have proposed novel time-reversible algorithms for rotational motion, the Fast Time-Reversible (FT) algorithms. They are time-reversible and perhaps the simplest (fastest) numerical algorithm for rotational motion.

II. I have shown an algorithm to make configurations of ice-Ih. It makes configurations of water molecules in ice randomly within the limits of Bernal-Fowler's rule, and can be used to make ice even with dislocation.

III. As an application of the FT algorithm, I have devised a parallelized rigid body molecular dynamics computer code using MPI libraries, adopting the FT algorithm and fast multipole method (FMM) for the time-integration of the rotational motion and computation of the Coulomb forces. I regard a water molecule as a rigid body, and employ the TIP4P intermolecular interaction potential with an established reputation.

IV. Employing this code, I have performed large-scale simulation of quasi-liquid layers (liquidlike layers on the surface of ices at temperatures below the bulk melting point: QLLs) of ices-Ih (without dislocation and with screw dislocation) each put in a vacuum box by using several supercomputers. The largest dimension of the ices used in our simulation is 0.06 micrometer. Simulations have been performed at three temperatures $T_{\rm m} - 23$, $T_{\rm m} - 13$, and $T_{\rm m} - 1$ K for ices without dislocation and with screw dislocation, where $T_{\rm m}$ is the melting point of the TIP4P bulk-ice. In ices with screw dislocation, the dislocation line is parallel to z-axis and the magnitude of the Burgers vector is about 7.4 Å. As results of our simulation, I have found the following:

◦ **Ice without dislocation:**

(a) The QLLs are bumpy with the bump heights as large as a few inter-bilayer distances of ice (9 to 12 Å), and the widths of trenches reach 100Å. The liquid bumps fluctuate to form and break at a variety of places in a random manner.

(b) At relatively lower temperatures, the molecules in the second bilayer from the outside are partly melted. The ice molecules in the second bilayer melt easily when they are located under the trenches surrounding the liquid bumps. Furthermore, by the change of the locations of liquid bumps over time, the melted areas newly covered by thick bumps recrystallize.

(c) At a temperature slightly below the melting point, the second

bilayer under the liquid bumps melts entirely to form a liquid sheet.

(d) Microscopic properties (i.e., mean squared displacement and O-O distance) of the liquid bumps differ from that of the liquid sheet.

◦ **Ices with screw dislocation:**

(e) The properties of QLL are almost the same as those of ices without dislocation. Influence of the screw dislocation on the melting and formation of QLL has been limited to a nearby region around the dislocation line. Though α-QLL with a hemispherical shape always appeared from the outcrops of the screw dislocations in the experiments [7], I could not have found a relationship between the sites of the screw dislocation and those of the bumps in the simulation. A possible reason for the qualitative differences with respect to the effects of the screw dislocation on the formation of QLL between the present results and experimental observations [7, 8] is the following. The present ice is put in a vacuum while the ice in the experiment is put in a supersaturated water vapor environment. The QLL in the experiment might form by vapor deposition. Another possible reason is that the present ice with a screw dislocation is much smaller than the experimental one. The concentration of external stress, that is inevitable in experiments, around the dislocation line might be significant, resulting in spouting of melted water formed around the dislocation line.

The present results (a) and (b) will offer a novel picture of surface melting of sub-micrometer-scale ice, and indicate that the recrystallization will make environmental hydrophilic (acidic) substances dissolve to be incorporated substantially in the layer even if the QLL is thin.

In addition to the results above, I mention some other results obtained by simple observations and make some comments. In the cases where

$T = T_m - 23$ and $T = T_m - 13$ K, the third bilayer preserves crystalline structure. In the case where T_m-1 K, it is partly melted and I could not find any relations between the partly melted regions of the third bilayer and the locations of the trenches (Here the second layer is a liquid sheet). However, it is likely that there exist some relations between them. In the future work, I will investigate the relations between trenches and the third bilayer in detail. I will challenge to perform simulation in future work to evaluate the absorption caused by recrystallization. The results will help us to understand the formation of acid snow.

Appendix A

Making ices Ih under the first layer

We have shown the method to determine the directions of the water molecules of the first layer (i.e., $rot(m, n, 0)$) in Chapter 3. In this appendix, I give a method to determine those of the molecules under the first layer. I give only a few explanations for the method, since it can be obtained only by observing Fig. 3.2 with a brute force method.

A.1 Determine $rot(m, 0, l)$ and $rot(0, n, l)$

In this section, we determine the directions of water molecules on two vertical sides of the form $(m, 0, l)$ and $(0, n, l)$ where m and n are non-negative integers and $l \geq 1$. First of all, we determine the directions of molecules on the intersection of the two sides, i.e., $rot(0, 0, l)$. It is determined successively as follows:

A.1.1 $rot(0, 0, l)$

(1) If l=even, then

$$rot(0, 0, l) = \{1, 2, 3, 4, 5, 6\}.$$

(2) If l=odd, then

$$rot(0,0,l) = \begin{cases} 4,5,6 \ (rot(0,0,l-1) = 1,2,3) \\ 1,2,3 \ (rot(0,0,l-1) = 4,5,6). \end{cases}$$

As in Chapter 3, The expression above means that if $rot(0,0,l-1)$ =1, 2, or 3, we can take $rot(0,0,l)$ in $\{4,5,6\}$, and so on.

A.1.2 $rot(0,n,l)$

Here and in A.1.3, we determine the directions of the two sides. We define $rot(0,n,l)$ successively as follows:

do $l = 1$, $width_z$
  do $n = 1$, $width_y$
    determine $rot(0,n,l)$
  end do
end do

where the $rot(0,n,l)$'s are determined by the following conditions:

(1) If n=even and l=even, then

$$rot(0,n,l) = \begin{cases} 3,4,6 \ (rot(0,n-1,l-1) = 1,3,6) \\ 1,2,5 \ (rot(0,n-1,l-1) = 2,4,5). \end{cases}$$

(2) If n=even and l=odd, then

$$rot(0,n,l) = \begin{cases} 3 \begin{cases} rot(0,n-1,l)=1,3,6 \\ rot(0,n,l-1)=4,5,6 \end{cases} \\ 1,2 \begin{cases} rot(0,n-1,l)=2,4,5 \\ rot(0,n,l-1)=4,5,6 \end{cases} \\ 4,6 \begin{cases} rot(0,n-1,l)=1,3,6 \\ rot(0,n,l-1)=1,2,3 \end{cases} \\ 5 \begin{cases} rot(0,n-1,l)=2,4,5 \\ rot(0,n,l-1)=1,2,3 \end{cases} \end{cases}$$

(3) If n=odd and l=even, then

$$rot(0,n,l)=\begin{cases} 1 \begin{cases} rot(0,n-1,l)=1,3,4 \\ rot(0,n,l-1)=4,5,6 \end{cases} \\ 4,6 \begin{cases} rot(0,n-1,l)=1,3,4 \\ rot(0,n,l-1)=1,2,3 \end{cases} \\ 2,3 \begin{cases} rot(0,n-1,l)=2,5,6 \\ rot(0,n,l-1)=4,5,6 \end{cases} \\ 5 \begin{cases} rot(0,n-1,l)=2,5,6 \\ rot(0,n,l-1)=1,2,3 \end{cases} \end{cases}$$

(4) If n=odd and l=odd, then

$$rot(0,n,l)=\begin{cases} 1,4,6 \ (rot(0,n-1,l-1)=1,3,4) \\ 2,3,5 \ (rot(0,n-1,l-1)=2,5,6). \end{cases}$$

A.1.3 $rot(m,0,l)$

We define $rot(m,0,l)$ successively as follows:

do $l=1$, $width_z$
  do $m=1$, $width_x$
    determine $rot(m,0,l)$
  end do
end do

where the $rot(m,0,l)$'s are determined by the following conditions:

(1) If m=even and l=even, then

$$rot(m,0,l)=\{1,2,3,4,5,6\}.$$

(2) If m=even and l=odd, then

$$rot(m,0,l)=\begin{cases} 4,5,6 \ (rot(m,0,l-1)=1,2,3) \\ 1,2,3 \ (rot(m,0,l-1)=4,5,6). \end{cases}$$

(3) If m=odd and l=even, then

$$rot(m,0,l)=\begin{cases} 3 \begin{cases} rot(m-1,0,l)=2,3,6 \\ rot(m,0,l-1)=4,5,6 \end{cases} \\ 1,2 \begin{cases} rot(m-1,0,l)=1,4,5 \\ rot(m,0,l-1)=4,5,6 \end{cases} \\ 5,6 \begin{cases} rot(m-1,0,l)=2,3,6 \\ rot(m,0,l-1)=1,2,3 \end{cases} \\ 4 \begin{cases} rot(m-1,0,l)=1,4,5 \\ rot(m,0,l-1)=1,2,3 \end{cases} \end{cases}$$

(4) If m=odd and l=odd, then

$$rot(m,0,l) = \begin{cases} 3,5,6 \ (rot(m-1,0,l) = 3,2,6) \\ 1,2,4 \ (rot(m-1,0,l) = 1,4,5). \end{cases}$$

A.2 Determine $rot(m,n,l)$ with l=odd

So far, we have determined the first layer and the vertical two sides of an ice. Next we determine the directions of water molecules layer by layer under the condition that $rot(m,0,l)$, $rot(0,n,l)$, and $rot(m,n,l-1)$ have been already determined. (They have been determined in the previously.) In this section, we determine $rot(m,n,l)$ for the case where l=odd. It is not so difficult compared with the case where l=even ($l \geq 2$), which will be treated in the next section.
We proceed as

do $m = 1$, $width_x$
  do $n = 1$, $width_y$
    determine $rot(m,n,l)$
  end do
end do

as before, where the $rot(m,n,l)$'s are determined by the following con-

ditions (we consider l is fixed):

(1) If $m+n$=even, then

$$rot(m,n,l)= \begin{cases} 3 \begin{cases} rot(m,n-1,l)=1,3,6 \\ rot(m,n,l-1)=4,5,6 \end{cases} \\ 1,2 \begin{cases} rot(m,n-1,l)=2,4,5 \\ rot(m,n,l-1)=4,5,6 \end{cases} \\ 5 \begin{cases} rot(m,n-1,l)=2,4,5 \\ rot(m,n,l-1)=1,2,3 \end{cases} \\ 4,6 \begin{cases} rot(m,n-1,l)=1,3,6 \\ rot(m,n,l-1)=1,2,3 \end{cases} \end{cases}$$

(2) If $m+n$=odd, then

$$rot(m,n,l)= \begin{cases} 6 \begin{cases} rot(m-1,n,l)=2,3,6 \\ rot(m,n-1,l)=1,3,4 \end{cases} \\ 3,5 \begin{cases} rot(m-1,n,l)=2,3,6 \\ rot(m,n-1,l)=2,5,6 \end{cases} \\ 2 \begin{cases} rot(m-1,n,l)=1,4,5 \\ rot(m,n-1,l)=2,5,6 \end{cases} \\ 1,4 \begin{cases} rot(m-1,n,l)=1,4,5 \\ rot(m,n-1,l)=1,3,4 \end{cases} \end{cases}$$

A.3 Determine $rot(m,n,l)$ with l=even

In this section, we determine $rot(m,n,l)$ for the case where l=even under the condition that $rot(m,0,l)$, $rot(0,n,l)$, and $rot(m,n,l-1)$ have been already determined.

A.3.1 (1) $m+n$=odd

The case $m+n$=odd is not difficult. It is given by the following:

$$rot(m,n,l)= \begin{cases} 3 \begin{cases} rot(m-1,n,l)=2,3,6 \\ rot(m,n,l-1)=4,5,6 \end{cases} \\ 5 \begin{cases} rot(m-1,n,l)=2,3,6 \\ rot(m,n,l-1)=1,2,3 \\ rot(m,n-1,l)=2,5,6 \end{cases} \\ 6 \begin{cases} rot(m-1,n,l)=2,3,6 \\ rot(m,n,l-1)=1,2,3 \\ rot(m,n-1,l)=1,3,4 \end{cases} \\ 4 \begin{cases} rot(m-1,n,l)=1,4,5 \\ rot(m,n,l-1)=1,2,3 \end{cases} \\ 1 \begin{cases} rot(m-1,n,l)=1,4,5 \\ rot(m,n,l-1)=4,5,6 \\ rot(m,n-1,l)=1,3,4 \end{cases} \\ 2 \begin{cases} rot(m-1,n,l)=1,4,5 \\ rot(m,n,l-1)=4,5,6 \\ rot(m,n-1,l)=2,5,6 \end{cases} \end{cases}$$

A.3.2 (2) $m+n$=even

The case $m+n$=even is a little troublesome. Thus far, we could determine the directions successively. However, in the case where $m+n$=even, we must take into account some additional conditions. Roughly speaking, if we proceed as in the way described before, there can occur the situation that we cannot determine the direction. To see this, let us see Fig. A.1. Suppose that there occurred the situation as shown in Fig. A.1(a). Then, if we determine the direction of (1,1,2) water molecule as in Fig. A.1(b), we cannot determine the direction of the (1,2,2) water molecule. We could fortunately determine the directions successively so far, it was because that whatever direction of each molecule in the possible options we chose, there

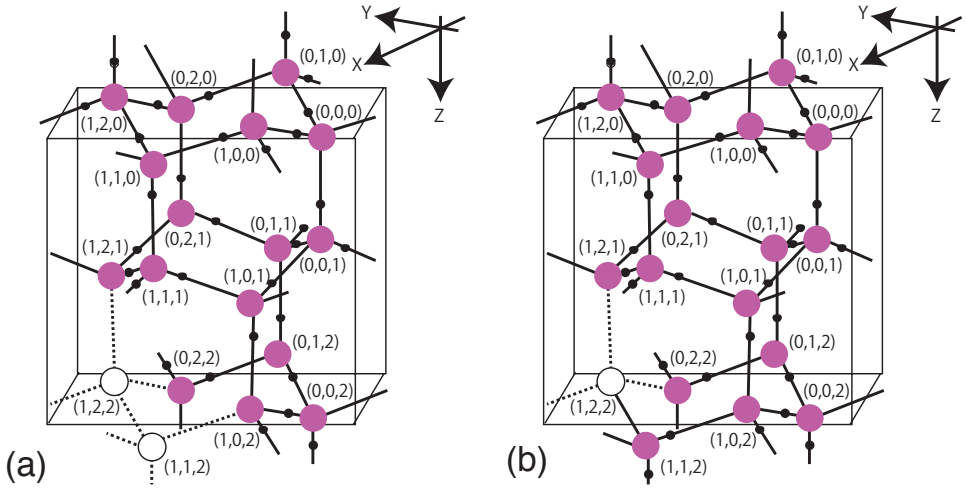

Figure A.1: The difficulties to determine the directions of water molecules for the case l=even and $m+n$=even.

was a possibility to choose direction of the next molecule.

To avoid such inextricable maze, we have to take into account of the directions of the nearest neighbor molecules of the molecule which will be determined in the next step. We proceed as follows:

(2) If $m+n$=even, then we consider the following two conditions:
(A) $rot(m, n+1, l-1)$=$\{$4,5,6$\}$ and $rot(m-1, n+1, l-1)$=$\{$1,2,3$\}$
(B) $rot(m, n+1, l-1)$=$\{$1,2,3$\}$ and $rot(m-1, n+1, l-1)$=$\{$4,5,6$\}$.

I. If (A) is satisfied, then

$$rot(m,n,l) = \begin{cases} 2,5 \text{ if } rot(m,n-1,l) = 2,4,5 \\ \quad 6 \text{ if } rot(m,n-1,l) = 1,3,6. \end{cases}$$

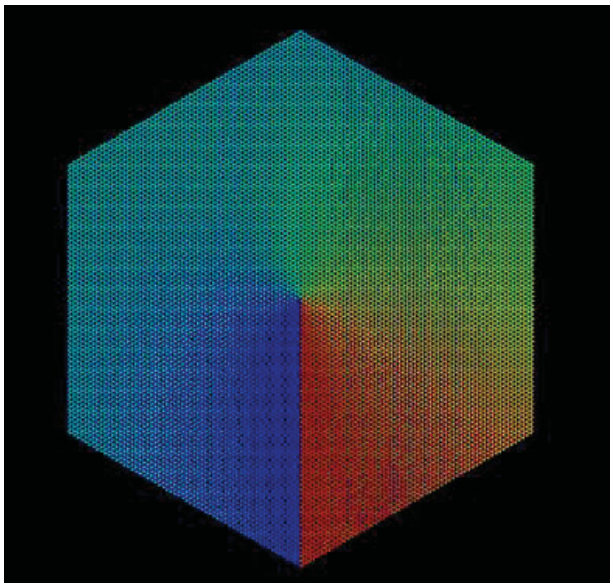

Figure A.2: Snapshot of ice with screw dislocation seen from above. Colored with respect to its height.

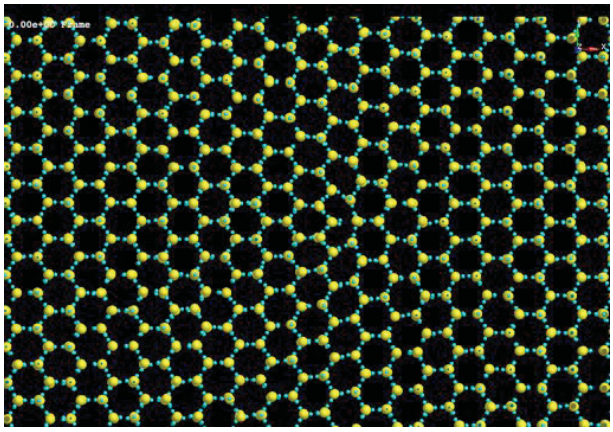

Figure A.3: Sliced (contain almost two layers) enlarged snapshot of ice with edge dislocation. (The simulation results of ices with edge dislocation are not presented in this thesis.)

II. If (B) is satisfied, then

$$rot(m,n,l) = \begin{cases} 1 \text{ if } rot(m,n-1,l) = 2,4,5 \\ 3,4 \text{ if } rot(m,n-1,l) = 1,3,6. \end{cases}$$

III. Otherwise,

$$rot(m,n,l) = \begin{cases} 1,2,5 \text{ if } rot(m,n-1,l) = 2,4,5 \\ 3,4,6 \text{ if } rot(m,n-1,l) = 1,3,6. \end{cases}$$

The above are the list of conditions necessary to make configurations of ice-Ih. In choosing numbers, we use a random generator.

Here I comment the way to generate ices with screw dislocation. They can be obtained by sticking two rectangular horizontally, and stacking the layers. The rectangles are made so that the half of their edges fit with the neighbor rectangles, and the left half edge fit with lower (or upper) rectangles. The method explained above can be applied to make such layers easily. Also for the case of edge dislocation, we can proceed similarly. Figures A.2 and A.3 present the snapshots of the ices with screw dislocation and with edge dislocation made by the present method.

Appendix B

Quaternion and its application

The notion of quaternions was first announced by William Hamilton and has been used to describe rotation of points in three dimensional spaces. The original definition of quaternion together with its multiplication is not presented in most of the books on dynamics except for some large books, however, the use of quaternion in that sense has an advantage that we can treat linear transformations as numbers. Therefore we recall some notion of quaternion and reconsider some results obtained in the previous chapters from the viewpoint of quaternion. The basic facts on quaternion can be found in standard textbooks on algebra such as [46].

B.1 Quaternion

B.1.1 Definition

We consider the four dimensional vector space over the real numbers spanned by four linearly independent basis $\{1,i,j,k\}$. The four basis commutes with real numbers and the multiplications of the basis are given as follows:

$$i^2 = j^2 = k^2 = -1,\ ij = -ji = k,\ jk = -kj = i,\ ki = -ik = j. \quad \text{(B.1)}$$

We call each element of the space as quaternion together with its multiplication. In this appendix, we understand that quaternions are combined with the multiplication.

Quaternions are written as $q = q_0 + q_1 i + q_2 j + q_3 k$ where q_0, q_1, q_2, and q_3 are real numbers. We define the conjugate of a quaternion $q = q_0 + q_1 i + q_2 j + q_3 k$ by

$$\overline{q} = q_0 - q_1 i - q_2 j - q_3 k. \tag{B.2}$$

Then $q\overline{q} = q_0^2 + q_1^2 + q_2^2 + q_3^2$ is a non-negative real number and we define the norm of q by

$$N(q) = \sqrt{q\overline{q}} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}. \tag{B.3}$$

It is easy to see that

$$N(qr) = N(q)N(r) \tag{B.4}$$

and

$$\overline{qr} = \overline{r}\ \overline{q}, \tag{B.5}$$

for any quaternions q and r, and $N(q) = 0$ holds if and only if $q = 0$. Therefore, for $q \neq 0$ the inverse of q exists and is given by

$$q^{-1} = \frac{\overline{q}}{N(q)^2}, \tag{B.6}$$

which implies that we can carry out divisions by every quaternion other than 0 (i.e., quaternions form a field). Here note that $\overline{q} = N(q)^2 q^{-1}$ and the conjugate of a unit quaternion ($N(q) = 1$) is equal to its inverse ($\overline{q} = q^{-1}$).

B.1.2 Rotation

We identify a point $P = (x_P, y_P, z_P)$ of the three dimensional Euclidean space E^3 with a quaternion q_P through

$$q_P = x_P i + y_P j + z_P k. \tag{B.7}$$

Note that $\overline{q_P} = -q_P$.

Let us consder the mapping

$$q_P \longrightarrow q q_P q^{-1}\ (= q q_P \overline{q} \frac{1}{N(q)^2}). \tag{B.8}$$

The real part of $qq_P\overline{q}$ is equal to 0 since $\overline{qq_P\overline{q}} = \overline{\overline{q}}\ \overline{q_P}\ \overline{q} = -qq_P\overline{q}$ (by Eq.(B.5)), and so qq_Pq^{-1} is identified with a point in E^3 through Eq.(B.7).

Thus we can see that $q_P \longrightarrow qq_Pq^{-1}$ is a mapping from E^3 to E^3 and it is easy to see that the mapping is a linear transformation which is represented by

$$\frac{1}{N(q)^2}\begin{pmatrix} q_0^2+q_1^2-q_2^2+q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{pmatrix}\begin{pmatrix} x_P \\ y_P \\ z_P \end{pmatrix}. \tag{B.9}$$

The transformation is distance-preserving.

Let us take a unit quaternion ($N(q) = 1$) defined by $q_i = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}i$. It can be seen that the quaternion q_i yields the rotation of points around the x-axis by the angle of θ by conjugation:

$$q_P \longrightarrow q_iq_Pq_i^{-1}. \tag{B.10}$$

Similarly, we can show that $q_j = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}j$ and $q_k = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}k$ rotate points around the y- and z-axes, respectively. These are verified by straightforward calculation.

Let us take a quaternion q_1 such that

$$q_1iq_1^{-1} = u_xi + u_yj + u_zk \tag{B.11}$$

for a given unit vector (u_x, u_y, u_z). We can find such q_1 by multiplying two of q_i, q_j and q_k. Since

$$N(u_xi + u_yj + u_zk) = N(q_1iq_1^{-1}) = N(q_1)N(i)N(q_1^{-1}) = N(i) = 1, \tag{B.12}$$

we assumed the vector (u_x, u_y, u_z) to be a unit vector.
Then, the rotation around the unit vector (u_x, u_y, u_z) by the angle of θ is given by

$$q_P \longrightarrow q_1q_iq_1^{-1}q_Pq_1q_i^{-1}q_1^{-1} = (q_1q_iq_1^{-1})q_P(q_1q_iq_1^{-1})^{-1} \tag{B.13}$$

from Eq.(B.11) and $i = q_1^{-1}(u_x i + u_y j + u_z k)q_1$, where $q_i = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}i$. Here we have

$$q_1 q_i q_1^{-1} = q_1(\cos\frac{\theta}{2} + \sin\frac{\theta}{2}i)q_1^{-1} = q_1\cos\frac{\theta}{2}q_1^{-1} + q_1\sin\frac{\theta}{2}iq_1^{-1} \tag{B.14}$$

$$= \cos\frac{\theta}{2} + \sin\frac{\theta}{2}(u_x i + u_y j + u_z k). \tag{B.15}$$

Hence the unit quaternion r defined by

$$r \equiv \cos\frac{\theta}{2} + \sin\frac{\theta}{2}(u_x i + u_y j + u_z k) \tag{B.16}$$

rotates points in the space around the unit vector (u_x, u_y, u_z) by the angle of θ through

$$q_P \longrightarrow rqr^{-1}(= rq\overline{r}). \tag{B.17}$$

B.2 Updating quaternion

In this section, we reconsider the results obtained in previous chapters from the view point of quaternion.

B.2.1 Derivative of quaternion

In the following, we consider two frames; body-fixed frame and space (laboratory)-fixed frame, and the body-fixed frame is so chosen as to coincide with principal moments of inertia of the rigid of interest. For a point in either of the frames, we add the quaternion q_P superscript "(b)" or "(s)" as $q_P^{(b)}$ or $q_P^{(s)}$ respectively to clarify the frame in which the point is represented.

Let us consider the rotation along an angular velocity vector $\vec{\omega}$, and put $q_\omega^{(b)} = \omega_1 i + \omega_2 j + \omega_3 k$. For the sake of simplicity, we put $\Omega \equiv q_\omega^{(b)} = \omega_1 i + \omega_2 j + \omega_3 k$. Then, the position of a rotated point P at a time t is represented as (see Eq.(B.16))

$$q_P^{(b)} \longrightarrow r_{\omega,t} q_P^{(b)} r_{\omega,t}^{-1} \tag{B.18}$$

where

$$r_{\omega,t} = \cos\frac{|\omega|t}{2} + \left(\sin\frac{|\vec{\omega}|t}{2}\right)\frac{\omega_1 i + \omega_2 j + \omega_3 k}{|\vec{\omega}|}. \tag{B.19}$$

Take a quaternion q_t that maps $q^{(b)}$ to $q^{(s)}$ at time t, i.e.,

$$q^{(b)} \longrightarrow q^{(s)} = q_t q^{(b)} q_t^{-1}. \tag{B.20}$$

Then, since $q_{t+\Delta t} = q_t r_{\omega,\Delta t} + O(\Delta t^2)$ for some ω, we have

$$\frac{\mathrm{d}q_t}{\mathrm{d}t} = \frac{1}{2} q_t \Omega \tag{B.21}$$

where $\Omega = \omega_1 i + \omega_2 j + \omega_3 k$. If we write down the relation above, we get the well-known formula:

$$\frac{d}{dt}\begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix}\begin{pmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{B.22}$$

where $q_t = q_0 + q_1 i + q_2 j + q_3 k$. Accordingly, we have

$$\frac{\mathrm{d}^2 q_t}{\mathrm{d}t^2} = \frac{1}{2}\frac{\mathrm{d}q_t}{\mathrm{d}t}\Omega + \frac{1}{2} q_t \frac{\mathrm{d}\Omega}{\mathrm{d}t} = \frac{1}{2} q_t \left(\frac{1}{2}\Omega^2 + \frac{\mathrm{d}\Omega}{\mathrm{d}t}\right). \tag{B.23}$$

B.2.2 Equations for updating quaternion

Now we define the time evolution of the quaternion by

$$\begin{aligned} q_{t+\Delta t} &\equiv q_t + \Delta t \frac{\mathrm{d}q_t}{\mathrm{d}t} + \frac{1}{2}\Delta t^2 \frac{\mathrm{d}^2 q_t}{\mathrm{d}t^2} \\ &= q_t \left(1 + \frac{1}{2}\Delta t \Omega + \frac{1}{4}\Delta t^2 \left(\frac{1}{2}\Omega^2 + \frac{\mathrm{d}\Omega}{\mathrm{d}t}\right)\right). \end{aligned} \tag{B.24}$$

Put

$$\tilde{\Omega}^+ = 1 + \frac{1}{2}\Delta t \Omega + \frac{1}{4}\Delta t^2 \left(\frac{1}{2}\Omega^2 + \frac{\mathrm{d}\Omega}{\mathrm{d}t}\right) = \tilde{\Omega}_1 + \tilde{\Omega}_x i + \tilde{\Omega}_y j + \tilde{\Omega}_z k. \tag{B.25}$$

Since Ω^2 is a real number and Ω and $\frac{d\Omega}{\mathrm{d}t}$ are pure imaginary (real part $= 0$), we have

$$\tilde{\Omega}_1 = 1 + \frac{1}{8}\Omega^2\Delta t^2 = 1 - \frac{1}{8}N(\Omega)^2\Delta t^2 = 1 - \frac{1}{8}(\omega_x^2 + \omega_y^2 + \omega_z^2)\Delta t^2$$

and

$$\tilde{\Omega}_x i + \tilde{\Omega}_y j + \tilde{\Omega}_z k = \frac{1}{2}\Delta t\left(\Omega + \Delta t\frac{1}{2}\frac{\mathrm{d}\Omega}{\mathrm{d}t}\right).$$

Then using Euler's equations for rigid body, we have

$$\tilde{\Omega}_x = \frac{\Delta t}{2}\left(\omega_x + \Delta t\left(\frac{I_2 - I_3}{2I_1}\omega_y\omega_z + \frac{t_x}{2I_1}\right)\right) = \frac{\Delta t}{2}\phi_x \tag{B.26}$$

$$\tilde{\Omega}_y = \frac{\Delta t}{2}\left(\omega_y + \Delta t\left(\frac{I_3 - I_1}{2I_2}\omega_x\omega_z + \frac{t_y}{2I_2}\right)\right) = \frac{\Delta t}{2}\phi_y \tag{B.27}$$

$$\tilde{\Omega}_z = \frac{\Delta t}{2}\left(\omega_z + \Delta t\left(\frac{I_1 - I_2}{2I_3}\omega_x\omega_y + \frac{t_z}{2I_3}\right)\right) = \frac{\Delta t}{2}\phi_z, \tag{B.28}$$

where we understand the ϕ_x, ϕ_y, ϕ_z are defined by the equations above.

The ϕ_x, ϕ_y, ϕ_z can be written as

$$\begin{aligned}
\phi_x &= \omega_x^{(n)} + (\alpha\omega_y^{(n)}\omega_z^{(n)} + \delta t_x^{(n)})\Delta t \\
\phi_y &= \omega_y^{(n)} + (\beta\omega_x^{(n)}\omega_z^{(n)} + \lambda t_y^{(n)})\Delta t \\
\phi_z &= \omega_z^{(n)} + (\gamma\omega_x^{(n)}\omega_y^{(n)} + \mu t_z^{(n)})\Delta t
\end{aligned} \tag{B.29}$$

where α, β, γ, δ, λ, μ are the same as defined in Eq.(2.27) in Chapter 2 and $t_x^{(n)}$, $t_y^{(n)}$, $t_z^{(n)}$ are components of torque vector and the superscript (n) is added to specify that the variables are of n-th time step. The equations above are the same as Eqs.(2.26).

We slightly modify $\tilde{\Omega}_1$ as follows

$$\tilde{\Omega}_1 = \sqrt{1 - (\tilde{\Omega}_x^2 + \tilde{\Omega}_y^2 + \tilde{\Omega}_z^2)}. \tag{B.30}$$

The difference between the modified $\tilde{\Omega}_1$ and the original $\tilde{\Omega}_1$ is within $O(\Delta t^3)$.

We time evolve q_t by

$$q_{t+\Delta t} = q_t \tilde{\Omega}^+, \tag{B.31}$$

where $\tilde{\Omega}^+ = \tilde{\Omega}_1 + \tilde{\Omega}_x i + \tilde{\Omega}_y j + \tilde{\Omega}_z k$. Since $N(\tilde{\Omega}^+) = 1$, $q_{t+\Delta t}$ is a unit quaternion.

We now determine angular velocity $\vec{\omega}^{(n+1)}$ at $n+1$-th step so as to satisfy time-reversibility:

$$q_{t+\Delta t} \tilde{\Omega}^- = q_t, \tag{B.32}$$

where $\tilde{\Omega}^-$ is obtained by replacing Δt by $-\Delta t$ and the data of n-th step by $n+1$-th step in Eqs.(B.26)$\sim$(B.29). Then we have

$$\tilde{\Omega}^- \equiv \tilde{\Omega}_1^- + \tilde{\Omega}_x^- i + \tilde{\Omega}_y^- j + \tilde{\Omega}_z^- k = \tilde{\Omega}_1^- - \frac{\Delta t}{2}\phi_x^- i - \frac{\Delta t}{2}\phi_y^- j - \frac{\Delta t}{2}\phi_z^- k \tag{B.33}$$

where

$$\begin{aligned}
\phi_x^- &= \omega_x^{(n+1)} - (\alpha \omega_y^{(n+1)} \omega_z^{(n+1)} + \delta t_x^{(n+1)})\Delta t \\
\phi_y^- &= \omega_y^{(n+1)} - (\beta \omega_x^{(n+1)} \omega_z^{(n+1)} + \lambda t_y^{(n+1)})\Delta t \\
\phi_z^- &= \omega_z^{(n+1)} - (\gamma \omega_x^{(n+1)} \omega_y^{(n+1)} + \mu t_z^{(n+1)})\Delta t
\end{aligned} \tag{B.34}$$

and

$$\tilde{\Omega}_1^- = \sqrt{1 - ((\tilde{\Omega}_x^-)^2 + (\tilde{\Omega}_y^-)^2 + (\tilde{\Omega}_z^-)^2)}. \tag{B.35}$$

Eq.(B.32) is satisfied if

$$\tilde{\Omega}^- = \overline{\tilde{\Omega}^+}. \tag{B.36}$$

The equation is equivalent to

$$\phi_x = \phi_x^-, \ \phi_y = \phi_y^-, \ \phi_z = \phi_z^-. \tag{B.37}$$

We determine $\omega_x^{(n+1)}$, $\omega_y^{(n+1)}$, $\omega_z^{(n+1)}$ to be the solutions of these equations (these equations are the same as Eq.(2.33) in Chapter 2), and the so obtained angular velocity vector $\vec{\omega}^{(n+1)}$ satisfies time-reversibility.

Appendix C

Modified FT algorithm

In the last of Chapter 2, I have mentioned that I can modify FT algorithm to remove square root used in updating angular velocity (except for calculations for normalizing quaternion). When the simulation is performed in abnormally high temperature or very long time-step, there is a danger that the value in the square root may be a negative number and the computer may crash. Therefore, it is desirable to remove such a square root in FT algorithm. Thus I show in this appendix the FT algorithm without the square root. I call the algorithm *modified FT algorithm*. I present the method of this modified FT algorithm below, which can also be obtained even by using quaternion algebra introduced in Appendix B.

C.1 Modification of the matrix

We first modify the matrix in Eq.(2.18) as below.

$$\overleftrightarrow{R}[\vec{v}] = \begin{pmatrix} 1-\frac{1}{2}|\vec{v}|^2 & -v_x & -v_y & -v_z \\ v_x & 1-\frac{1}{2}|\vec{v}|^2 & v_z & -v_y \\ v_y & -v_z & 1-\frac{1}{2}|\vec{v}|^2 & v_x \\ v_z & v_y & -v_x & 1-\frac{1}{2}|\vec{v}|^2 \end{pmatrix}. \tag{C.1}$$

Note that we have

$${}^t\overleftrightarrow{R}[\vec{v}] = \overleftrightarrow{R}[-\vec{v}] \tag{C.2}$$

as in the original matrix. The differences between this matrix and the original one lie only in diagonal elements, which are within $O(\Delta t^3)$. The

matrix in Eq.(2.18) is orthogonal, on the other hand, the modified matrix above is not orthogonal but satisfies

$$\overleftrightarrow{R}[\vec{v}]\ {}^{\mathrm{t}}\overleftrightarrow{R}[\vec{v}] = (1+\frac{1}{4}|\vec{v}|^4)\overleftrightarrow{E}, \tag{C.3}$$

which implies that the matrix $\overleftrightarrow{R}[\vec{v}]$ is "(orthogonal matrix) $\times$ (constant)".

C.2 Time-reversibility and angular velocity

In the modified FT algorithm, we update the quaternion through

$$\vec{q}(t+\Delta t) = \overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t,\Delta t)\Delta t\right]\vec{q}(t) \tag{C.4}$$

as in Chapter 2 where $\vec{\phi}$ is the same as defined in that chapter. The angular velocity vector $\vec{\omega}^+ = (\omega_x^+,\ \omega_y^+,\ \omega_z^+)$ of the next step is determined by the following equation

$$\vec{\phi}(t+\Delta t,-\Delta t) = \vec{\phi}(t,\Delta t) \tag{C.5}$$

as before.

I show in the following that the angular velocity vector $\vec{\omega}^+ = (\omega_x^+,\ \omega_y^+,\ \omega_z^+)$ obtained as above satisfies time-reversibility. From Eq.(C.5), we have

$$\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t+\Delta t,-\Delta t)(-\Delta t)\right] = \overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t,\Delta t)(-\Delta t)\right]. \tag{C.6}$$

Since

$$\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t,\Delta t)(-\Delta t)\right]\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t,\Delta t)\Delta t\right] = c\overleftrightarrow{E} \tag{C.7}$$

from Eqs.(C.2) and (C.3), we have

$$\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t+\Delta t,-\Delta t)(-\Delta t)\right]\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t,\Delta t)\Delta t\right] = c\overleftrightarrow{E}. \tag{C.8}$$

Since we update the quaternion through

$$\vec{q}(t+\Delta t) = \overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t,\Delta t)\Delta t\right]\vec{q}(t), \tag{C.9}$$

the $\vec{q}_t\prime$ obtained by time-reversing $\vec{q}_{t+\Delta t}$ for $-\Delta t$ is, or in other words, by operating the matrix

$$\overleftrightarrow{R}\left[\frac{1}{2}\vec{\phi}(t+\Delta t,-\Delta t)(-\Delta t)\right] \tag{C.10}$$

is, not equal to $\vec{q}_t$ but equal to $c'\vec{q}_t$ for some scalar c'. Hence the normalized quaternion coincides with $\vec{q}_t$. Therefore we conclude that by taking the solution $\vec{\omega}^+ = (\omega_x^+, \omega_y^+, \omega_z^+)$ of Eq.(C.5) as the angular velocity vector of the next step, the modified FT algorithm (where we normalize quaternions every step) satisfies time-reversibility. (If $\vec{q}_{t+\Delta t}$ is normalized, $c' \neq c$ in general.)

C.3 Modified FT algorithm for single time step

We describe here the procedure for a single time step of the modified FT algorithm especially for FT2, as in Chapter 2. The deferences between modified and non-modified FT algorithm are only in the step 4, and denoted by bold-face letters.

Step 1: Calculate the atomic positions of the molecule in the space-fixed frame by Eqs. (2.6) and (2.7).

Step 2: Calculate the forces on the atoms using the atomic positions calculated in Step 1. Then calculate the force $\vec{f}(t)^{(\mathrm{s})}$ acting on the centroid, and the torque $\vec{\tau}(t)^{(\mathrm{b})} = {}^{\mathrm{t}}(t_x, t_y, t_z)$.

Step 3: Determine the updated position of the centroid by Eq. (2.4).

Step 4: Determine $\vec{\phi}(t,\Delta t) = {}^{\mathrm{t}}(\phi_x(t,\Delta t), \phi_y(t,\Delta t), \phi_z(t,\Delta t))$ by Eqs. (2.38)-(2.40). Set it into **Eq. (C.9)** to determine the updated quaternion and **normalize the quaternion**.

Step 5: Calculate the updated positions of the atoms in the space-fixed frame by Eq. (2.7).

Step 6: Calculate the updated forces on the atoms. Then obtain the torque $\vec{\tau}^{+(\mathrm{b})} = {}^{\mathrm{t}}(t_x^+, t_y^+, t_z^+)$ and the force $\vec{f}(t+\Delta t)$ acting on the centroid.

Step 7: Determine the updated velocity of the centroid by Eq. (2.5).

Step 8: Determine the updated angular velocity, ω_x^+, ω_y^+, and ω_z^+, by Eqs. (2.41)-(2.43) after setting $\phi_x^+ = \phi_x$, $\phi_y^+ = \phi_y$, and $\phi_z^+ = \phi_z$ where ϕ_x, ϕ_y, and ϕ_z are obtained in Step 4 and t_x^+, t_y^+, and t_z^+ in Step 6.
Go to Step 3.

C.3.1 Listing of sample subroutines of modified FT algorithm

Here we show listing of two sample subroutines of modified FT algorithm, `q_update` and `angv_update`, for a particle. In actual simulation of a molecular system, the subroutines will be applied in parallel to all the rigid molecules. The motions of centroids are calculated separately for all particles in a usual manner. The subroutine `q_update` updates angular velocity vectors, and the subroutine `angv_update` calculate angular velocity vectors at the next time step.

C.3.1.1 subroutine q_update

```
c----- This subroutine updates angular velocities.
      subroutine q_update(qua,omega,phi,torque,dt,ai)
      implicit none
      real*8 qua(4), omega(3), torque(3), tb(3)
      real*8 Ain(1:3,1:3),ai(3), aa,c,e
      real*8 qqq,trans(4),dt,ppp
      real*8 omegax(3),phi(3),b,d,f,qqua(4)
      integer i

c----- The subroutine below get matrix Ain from quaternion.
c----- qua is a quaternion, where qua(1)=q_0, qua(2)=q_1,
c----- qua(3)=q_2, qua(4)=q_3. (p.12)
c----- Ain (=(2.6)) is a matrix, which transform the coordinates
c----- of a vector represented in space-fixed system to that
c----- in the body-fixed coordinates.
c----- Ain is the inverse of matrix (2.6). (p.12)

      call matrix(qua, Ain)

c----- (torque(1),torque(2),torque(3)) is a torque vector
c----- represented in space-fixed frame which is calculated
```

```
c----- by other subroutine. (p.16).
c----- tb(3) is the torque vector represented in body-fixed
c----- frame.

      tb(1)=Ain(1,1)*torque(1)+Ain(1,2)*torque(2)
     &   +Ain(1,3)*torque(3)
      tb(2)=Ain(2,1)*torque(1)+Ain(2,2)*torque(2)
     &  +Ain(2,3)*torque(3)
      tb(3)=Ain(3,1)*torque(1)+Ain(3,2)*torque(2)
     &  +Ain(3,3)*torque(3)

c----- ai are principal moment of inertia, ai(1)=I_x=1.9167,
c----- ai(2)=I_y=0.66632, and ai(3)=I_z=1.25037. (p.12)
c----- aa,c, and e are equal to \alpha, \beta, and \gamma,
c----- defined in p.16.

      b=tb(1)/(2d0*ai(1))
      d=tb(2)/(2d0*ai(2))
      f=tb(3)/(2d0*ai(3))
      aa=(ai(2)-ai(3))/(2d0*ai(1))
      c=(ai(3)-ai(1))/(2d0*ai(2))
      e=(ai(1)-ai(2))/(2d0*ai(3))

c----- omega is an angular velocity vector, omega(1)=\phy_x
c----- and so on. See (2.39). (p.12)

       omegax(1)=omega(1)+b*dt
       omegax(2)=omega(2)+d*dt
       omegax(3)=omega(3)+f*dt

c----- (2.40).
       omegax(1)=omegax(1)+aa*omegax(2)*omegax(3)*dt
       omegax(2)=omegax(2)+c*omegax(1)*omegax(3)*dt
       omegax(3)=omegax(3)+e*omegax(1)*omegax(2)*dt

c----- Vector phi  is used to get
c----- angular velocity at the next time step. (C.5)
       do i=1,3
       phi(i)=omega(i)
       end do
```

```
c----- See (C.4), where we multiply phi by dt*0.5.
       trans(1)=omega(1)*dt*0.5d0
       trans(2)=omega(2)*dt*0.5d0
       trans(3)=omega(3)*dt*0.5d0
       trans(4)=1d0-(trans(1)**2+trans(2)**2+trans(3)**2)*0.5d0

c----- (C.4)
       qqua(1)=qua(1)*trans(4)
     &  -qua(2)*trans(1)-qua(3)*trans(2)-qua(4)*trans(3)
       qqua(2)=qua(1)*trans(1)
     &  +qua(2)*trans(4)+qua(3)*trans(3)-qua(4)*trans(2)
       qqua(3)=qua(1)*trans(2)
     &  -qua(2)*trans(3)+qua(3)*trans(4)+qua(4)*trans(1)
       qqua(4)=qua(1)*trans(3)
     &  +qua(2)*trans(2)-qua(3)*trans(1)+qua(4)*trans(4)

       qua(1)=qqua(1)
       qua(2)=qqua(2)
       qua(3)=qqua(3)
       qua(4)=qqua(4)

       ppp=dsqrt(qua(1)**2+qua(2)**2+qua(3)**2+qua(4)**2)

c----- Normalize quaternion.
       qua(1)=qua(1)/ppp
       qua(2)=qua(2)/ppp
       qua(3)=qua(3)/ppp
       qua(4)=qua(4)/ppp


        return
        end
```

C.3.1.2 subroutine angv_update

```
c-----This subroutine calculate angular velocity vector of the next step.
       subroutine angv_update(torque,nextomega,phi,dt,ai)

       implicit none
       real*8 torque(3),dt,ai(3),nextomega(3)
       real*8 phi(3),aa,b,c,d,e,f
```

```
        aa=(ai(2)-ai(3))/(2d0*ai(1))
        c=(ai(3)-ai(1))/(2d0*ai(2))
        e=(ai(1)-ai(2))/(2d0*ai(3))
        b=torque(1)/(2d0*ai(1))
        d=torque(2)/(2d0*ai(2))
        f=torque(3)/(2d0*ai(3))

        om(1)=phi(1)
        om(2)=phi(2)
        om(3)=phi(3)

c------ (2.41)
        phi(3)=phi(3)+e*phi(1)*phi(2)*dt
        phi(2)=phi(2)+c*phi(1)*phi(3)*dt
        phi(1)=phi(1)+aa*phi(2)*phi(3)*dt

c------ (2.42)
        phi(1)=phi(1)+b*dt
        phi(2)=phi(2)+d*dt
        phi(3)=phi(3)+f*dt

c------ (2.43)
        nextomega(1)=phi(1)
        nextomega(2)=phi(2)
        nextomega(3)=phi(3)

        return
        end
```

Thus getting the angular positions, we calculate the coordinates of atoms in the space-fixed frame by adding the coordinate of the centroid of each molecule to the coordinates of each atom in the molecule in the space-parallel frame (with its origin coincides with the centroid of the molecule), which is obtained as the image of the atom in the body-fixed frame by the operation of inverse matrix of `Ain`. Then we can calculate the force acting on atoms.

C.3.2 Subroutines of the algorithm

Time-evolution of a molecule is performed by updating the quaternion `qua`, position of the centroid `cent`, the angular velocity `omega`, and velocity of the centroid `v`. One step time-evolution from step n to step $n+1$ is executed mainly by the following subroutines, which correspond to steps from 3 to 8 in the beginning of C.3.

We assume that we know already `qua`, position of each atom `r`, `cent`, `omega`, `v`, force `F`, and torque `torque`, at step n. We add superscript + to indicate that the corresponding data is of step $n+1$.
(1) `c_update(r,F,v,cent,cent`$^+$`dt)`: Update the position of centroid `cent` of each molecule from the force `F` and velocity `v`.
(2) `q_update(qua,qua`$^+$`,omega,phi,torque,dt,ai)`: Update the quaternion of molecule.
(3) `qua2pos(qua`$^+$`,cent`$^+$`,r`$^+$`)`: Get the position of each atom `r`$^+$ by the quaternion `qua`$^+$ and the position of the centroid `cent`$^+$ of the molecule containing the atom.
(4) `getforce(r`$^+$`,F`$^+$`)`: Calculate force `F`$^+$ acting on each atom from the positions of atoms `r`$^+$.
(5) `gettorque(r`$^+$`,F`$^+$`,torque`$^+$`)`: Calculate torque `torque`$^+$ from the force.
(6) `v_update(F,F`$^+$`,v,v`$^+$`,dt)`: Update the velocity of centroid `v` using the force `F` and `F`$^+$.
(7) `angv_update(torque`$^+$`,omega`$^+$`,phi,dt,ai)`: Update angular velocity. (`omega`$^+$ is equal to `nextomega` in C.3.1.2.)

C.4 Conclusion

The modified FT algorithm differs from original one in the following two points:

I. We modify the matrix Eq.(2.18) in FT algorithm as Eq.(C.1) and time evolve quaternion by this matrix.

II. Normalize the quaternion above every step in the FT algorithm.

The modification of the modified FT algorithm yields the increase of number of operations. Eleven operations are required for the modification, nevertheless the number of operations are least of all the other algorithms (see table 2.1). I performed simulations with this modified FT algorithm with timestep=2 fs and 4 fs for 100,000 steps, the stability was no less than FT algorithm.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Shuji Ogata, for his guidance throughout my studies. I wish to thank Dr. M. Hiyama, Dr. R. Kobayashi, Dr. T. Tamura and the members of Ogata Laboratory in Nagoya Institute of Technology for helpful discussions.

The computations were performed using Fujitsu FX10 and Hitachi HA8000 at Information Technology Center of Univ. of Tokyo, Fujitsu FX10 at Institute for Solid State Physics of Univ. of Tokyo, Hitachi SR16000 at Institute of Material Research of Tohoku Univ., Fujitsu FX1 and FX10 at Information Technology Center of Nagoya Univ., the K computer at RIKEN AICS, and Fujitsu PRIMERGY at Research Center for Computational Science (Okazaki).

I thank these facilities/institutions for giving us allowances to use the supercomputers.

Bibliography

[1] Y. Li and Q. A. Somorjai, J. Phys. Chem. C **111**, 9631 (2007).

[2] R. Rosenberg, Phys. Today **58**, 12 (2005).

[3] K. Diehl, S. K. Mitra, and H. R. Pruppacher, Atmos. Environ. **29**, 975 (1995).

[4] T. Ikeda-Fukazawa and K. Kawamura, J. Chem. Phys. **120**, 1395 (2004).

[5] K. Bolton and J. B. C. Petterson, J. Phys. Chem. B **104**, 1590 (2000).

[6] C. L. Bishop, D. Pan, L. M. Liu, G. A. Tribello, A. Michaelides, E. G. Wang, and B. Slater, Faraday Discuss **141**, 277 (2009).

[7] G. Sazaki, H. Asakawa, K. Nagashima, S. Nakatsubo, and Y. Furukawa, Cryst. Growth & Des. **13**, 1761 (2013).

[8] G. Sazaki, S.Zepeda, S. Nakatsubo, M. Yokomine, and Y. Furukawa, Proc. Natl. Acad. Sci. **109**, 1052 (2012).

[9] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein, J. Chem. Phys. **79**, 926 (1983).

[10] L. Greengard and V. Rokhlin, J. Comp. Phys. **73**, 325-348 (1987).

[11] S. Ogata, T. J. Campbell, R. K. Kalia, A. Nakano, P. Vashishta, and S. Vemparala, Comput. Phys. Commun. **153**, 445 (2003).

[12] Y. Kajima, M. Hiyama, S. Ogata, and T. Tamura, J. Phys. Soc. Jpn. **80**, 114002 (2011).

[13] Y. Kajima, M. Hiyama, S. Ogata, R. Kobayashi, and T. Tamura, J. Chem. Phys. **136**, 234105 (2012).

[14] Y. Kajima, S. Ogata, R. Kobayashi, M. Hiyama, and T. Tamura, J. Phys. Soc. Jpn. **83**, 83601 (2014).

[15] J. D. Bernal and R. H. Fowler, J. Chem. Phys. **1**, 515 (1933).

[16] R. G. Fernandez, J. L. F. Abascal, and C. Vega, J. Chem. Phys. **124**, 144506 (2006).

[17] A. Leach, *Molecular Modelling: Principles and Applications, 2nd Ed.* (Prentice Hall, NJ, U.S.A., 2001).

[18] E. Sanz, C. Vega, J. L. F. Abascal, and L. G. MacDowell, Phys. Rev. Lett. **92**, 255701 (2004).

[19] M. Matsumoto, S. Saito, and I. Ohmine, Nature **416**, 409 (2002).

[20] K. Mochizuki, M. Matsumoto, and I. Ohmine, Nature **498**, 350 (2013).

[21] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford Univ. Press, Oxford, U.K., 1987).

[22] N. Matubayasi and M. Nakahara, J. Chem. Phys. **110**, 3291 (1999).

[23] T. F. Miller III, M. Eleftheriou, P. Pattnaik, A. Ndirango, D. Newns, and G. J. Martyna, J. Chem. Phys. **116**, 8649 (2002).

[24] A. Koi, B. B. Laird, and B. J. Leimkuhler, J. Chem. Phys. **107**, 2580 (1997).

[25] A. Dullweber, B. Leimkuhler, and R. McLachlan, J. Chem. Phys. **107**, 5840 (1997).

[26] R. van Zon and J. Schofield, Phys. Rev. E **75**, 056701 (2007) .

[27] H. Okumura, S. G. Itoh, and Y. Okamoto, J. Chem. Phys. **126**, 084103 (2007).

[28] M. Hiyama, T. Kinjo, and S. Hyodo, J. Phys. Soc. Jpn. **77**, 064001 (2008).

[29] H. Goldstein, *Classical Mechanics, 2nd Ed.* (Addison-Wesley, MA, U.S.A., 1980).

[30] In Ref. 11, we used (ξ, η, ζ, χ) instead of (q_0, q_1, q_2, q_3). The relations are, $q_0 = \chi,\ q_1 = \eta,\ q_2 = -\xi$, and $q_3 = \zeta$.

[31] If $s = 1 - \frac{1}{2}\left|\frac{1}{2}\vec{\omega}(t)\right|^2 \Delta t^2 < 0$, it implies that $|\vec{\omega}(t)|\Delta t > 2\sqrt{2} > \frac{2\pi}{3}$. Roughly speaking, it means that three steps make one rotation. It is the time step too long for a molecular dynamics simulation to be applied. Actually, in our simulation, $1 \geq s > 0.99$ (Δt=2fs), and $1 \geq s > 0.75$ (Δt=10fs). The situation of $s < 0$ never happened in our simulations. Similarly, if $|\vec{V}| > 1$, we have $|\vec{\omega}(t) + \frac{1}{2}\frac{d}{dt}\vec{\omega}(t)\Delta t|\Delta t > 2 \approx \frac{2\pi}{3} (= 2.09\cdots)$. Since $\vec{\omega}(t) + \frac{1}{2}\frac{d}{dt}\vec{\omega}(t)\Delta t$ is roughly equal to the mean of angular velocities of the present step and the next time step, $|\vec{V}| < 1$ is always satisfied in actual simulation.

[32] D. Eisenberg and W. Kauzman, *The Structure and Properties of Water* (Oxford Univ. Press, Oxford, U.K., 2005).

[33] Since the system temperature increases significantly after 10^5 steps for Δt =7.0 fs, we calculate $\epsilon(1)$ in a short run of 10^3 steps, and regard $(\epsilon(2000) - \epsilon(100))/(2000 - 100)$ in a run of 5000 steps as $\tilde{\epsilon}$ in such a case.

[34] We define the number of operations by the sum of mathematical operations appeared in the algorithm for updating angular velocity and quaternion of a rigid molecule. Here mathematical operations are addition, subtraction, multiplication, division, exponentiation, trigonometric function, and square root, and each of these operations is counted as one operation. Substitution is not considered as operation. For example, the number of operations of Eqs. (2.39) is 9. Operations in the FT2 algorithm mean the operations appeared in

Eqs. (2.38)-(2.43), and (2.23), and operations in the symplectic algorithm are operations appeared in Eqs. (58)-(84) in Ref. 8, omitting operations concerning with thermostat and translational motion.

[35] A. Rahman and F. H. Stillinger, J. Phys. Chem. **57**, 4009 (1972)

[36] V. Buch, P. Sandler, and J. Sadlej, J. Phys. Chem. **102**, 8641 (1998)

[37] E. Cota and W. G. Hoover, J. Chem. Phys. **67**, 3839 (1977)

[38] J. A. Hayward and J. R. Reimers, J. Chem. Phys. **106**, 1518 (1997)

[39] V. F. Petrenko and R. W. Whiteworth, *Physics of ice* (Oxford Univ. Press, Oxford, U.K., 1999).

[40] M. Faraday, Proc. R. Soc. London **10**, 440 (1860).

[41] A. Kouchi, Y. Furukawa, and T. Kuroda, J. Phys. Colloq. **48**, C1-675 (1987).

[42] Y. Furukawa, M. Yamamoto, and T. Kuroda, J. Cryst. Growth. **82**, 665 (1987).

[43] Y. Furukawa and H. Nada, J. Phys. Chem. B **101**, 6167 (1997).

[44] M. M. Conde, C. Vega, and A. Patrykiejew, J. Chem. Phys. **129**, 014702 (2008).

[45] D. Pan, L-M. Liu, B. Slater, A. Michaelides, and E. Wang, ACS Nano. **5**, 4562 (2011).

[46] Serge Lang, *Algebra* (Graduate Texts in Mathematics, Springer, 2005).