Computational techniques in molecular dynamics and detonation shock dynamics

Permalink
https://escholarship.org/uc/item/8np637h4

Author
Grogan, Francesca Catherine

Publication Date
2017

Peer reviewed|Thesis/dissertation
UNIVERSITY OF CALIFORNIA, SAN DIEGO

Computational techniques in molecular dynamics and detonation shock dynamics

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy

in

Mathematics

by

Francesca Catherine Grogan

Committee in charge:

Professor Michael Holst, Chair
Professor Rommie Amaro
Professor Randolph Bank
Professor Melvin Leok
Professor Lee Lindblom
Professor John Weare

2017
The dissertation of Francesca Catherine Grogan is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

Chair

University of California, San Diego

2017
DEDICATION

To Sinan and my parents.
# TABLE OF CONTENTS

Signature Page ................................................. iii  
Dedication ....................................................... iv  
Table of Contents ........................................... v  
List of Figures ............................................... vii  
List of Tables ............................................... ix  
Acknowledgements .......................................... x  
Vita and Publications .................................... xii  
Abstract of the Dissertation ............................. xiii  

Chapter 1  Introduction ...................................... 1  
1.1 Overview of thesis .................................... 1  
1.2 Introduction to error quantification for large-scale molecular dynamics (MD) ......................... 3  
1.2.1 The MD equations ................................. 5  
1.2.2 Solving the MD equations ....................... 6  
1.2.3 Ensemble computations .......................... 8  
1.2.4 Approximating methods ......................... 9  

Chapter 2  Large-scale molecular dynamics .............. 12  
2.1 Preliminaries .......................................... 12  
2.2 Representation of $\delta \vec{x}_A$ using normal mode analysis ........................................ 14  
2.3 Evolution equations for large-scale dynamics ............................. 16  
2.3.1 Approximation using a normal mode basis representation ............................................. 17  
2.3.2 Large-scale dynamics approximation .......... 19  
2.4 Error quantification using large-scale dynamics approximation ............................................. 24  

Chapter 3  Numerical results for the large-scale dynamics approximation ................................................. 30  
3.1 Setup .................................................. 30  
3.2 Results ................................................. 33  
3.2.1 Error in thermodynamic averages using our large-scale dynamics approximation .................. 33  
3.2.2 Error in macroscopic quantities and mode amplitudes using our large-scale dynamics approximation 34
Figure 2.1: Comparison of normal-mode basis approximation using a carbon fullerene \((C_{20})\): 1 analytical mode versus all 54 analytical modes. Dotted curves are results using a single analytical mode, solid curves are results using all analytical modes. 

Figure 3.1: The \(C_{20}, C_{26}, C_{60}, \) and \(C_{70}\) (left to right) molecules used in our molecular dynamics example simulations.

Figure 3.2: \(L_2\) energy error using large-scale dynamics in \(C_{20}, C_{26}, C_{60}, \) and \(C_{70}\) molecules. Left: \(A_\mu = 0\). Right: sinusoidal evolution of \(A_\mu\).

Figure 3.3: \(L_2\) error in linear and angular momentum setting \(A_\mu = 0\). Top row: \(C_{26}\). Bottom row: \(C_{70}\).

Figure 3.4: \(L_2\) error in linear and angular momentum setting \(A_\mu = A_\mu^0 \sin(\omega_\mu t + \phi_\mu) \vec{e}_A\) where \(A_\mu^0 \neq 0\). Top row: \(C_{26}\). Bottom row: \(C_{70}\).

Figure 3.5: \(L_2\) error in position and velocity variables and spatial orientation variables setting \(A_\mu = 0\). Top row: \(C_{26}\). Bottom row: \(C_{70}\).

Figure 3.6: \(L_2\) error in position and velocity variables and spatial orientation variables setting \(A_\mu = A_\mu^0 \sin(\omega_\mu t + \phi_\mu) \vec{e}_A\) where \(A_\mu^0 \neq 0\). Top row: \(C_{26}\). Bottom row: \(C_{70}\).

Figure 3.7: Energy and momenta comparison between \(A_\mu\) being evolved sinusoidally (NM amps) versus zero mode amplitudes \((A_\mu = 0)\). Top row: \(C_{26}\). Bottom row: \(C_{70}\).

Figure 3.8: Variable error comparison between \(A_\mu\) being evolved sinusoidally (NM amps) versus zero mode amplitudes \((A_\mu = 0)\). Top row: \(C_{26}\). Bottom row: \(C_{70}\).

Figure 3.9: Runtimes for a \(C_{20}\) molecule with a classical MD solver versus the large-scale dynamics approximation with varying degrees of tolerance.

Figure 3.10: Runtimes for a \(C_{20}, C_{26}, C_{60}, \) and \(C_{70}\) molecule at tolerance level \(1E-7\).

Figure 3.11: Differences in atom position in Angstroms (Å). Convergences for a classical MD simulation versus an exact normal mode basis representation.

Figure 3.12: \(L_2\) energy error. Solid curves are results from the exact normal mode variable algorithm, dashed curves are results from a classical MD algorithm. Left: \(C_{26}\). Right: \(C_{70}\).

Figure 3.13: \(L_2\) error in linear and angular momentum. For the angular momentum graphs, solid curves are results from the exact normal mode variable algorithm, dashed curves are results from a classical MD algorithm. Top row: \(C_{26}\). Bottom row: \(C_{70}\).
Figure 5.1: Example of level set evolution: Propagation of a circle, i.e. the zero contour of $\phi$, on an unstructured mesh from $t_0$ (red) to $t_{final}$ (black).

Figure 5.2: Reinitialization preserves the signed distance function in this merging spheres problem.

Figure 5.3: Where does DSD fit in?

Figure 6.1: An initially distorted level set function (left) is reinitialized to a proper signed distance function (right) using $Q_2$ finite elements.

Figure 7.1: Deformation of circle from $t = 0$ (top left) to $t = 8$ (bottom right). In the third image, at $t = 4$, the circle reverses direction.

Figure 7.2: One clockwise rotation of notched disk (top left to bottom right). The notch maintains its integrity and does not deform as the disk rotates.

Figure 7.3: $L_2$ error over entire mesh for an expanding circle on a structured square. The floor reached with the modified redistancing is due to the addition of the diffusion coefficient $\mu$.

Figure 7.4: A circle, initialized at the star’s center, stably propagated outwards (left to right).

Figure 7.5: A detonation front (black), initialized as a small circle on the left edge (red), is propagated through the region of interest.

Figure 7.6: Top row: Simulation of an idealized detonation front with an expanding sphere. Bottom row: Simulation of merging detonation fronts with two expanding spheres.

Figure 7.7: (a) Initialization and propagation of front on an L-shape mesh. (b) Reinitialization correctly adapts and propagates level set contours around the sharp corner and compares well to final exact solution (c) and (d).

Figure 7.8: Propagation of a circular front around a hole of radius 0.2 (top left to bottom right).

Figure 7.9: Propagation using order one elements results in stable but inappropriate behavior. Top: The level set $\phi(t_0)$. Bottom: $\phi(t_{final})$.

Figure 7.10: Propagation using order two elements results in appropriate behavior for earlier times but goes unstable. Top: The level set $\phi(t_0)$. Bottom: $\phi(t_{final})$. 
LIST OF TABLES

Table 7.1: Convergence Rates with Classic Redistancing . . . . . . . . . . . 68
Table 7.2: Convergence Rates with Modified Redistancing ($\mu = 0.00025/p$) 69
ACKNOWLEDGEMENTS

There are many who made this work possible. First, of course, to my advisor Professor Michael Holst, for your invaluable support during my time at UCSD. For the past five years, your advice and comments have always come when I needed it the most; for this, I am very grateful.

To my committee members: Professors Rommie Amaro, Randolph Bank, Melvin Leok, Lee Lindblom, and John Weare, thank you for your time and support. I would like to especially thank Lee Lindblom, whose countless hours and e-mails these past three years were crucial to the first half of this thesis work. I am indebted to you for your time, advice, and patience. Thanks to several people at Lawrence Livermore National Laboratory: the organizers of the HEDP Summer Student Program, especially Lila Chase; Veselin Dobrev, Tzanio Kolev, Vladimir Tomov, and the rest of the MFEM team, for answering my never ending questions about MFEM; and my mentor, Rob Rieben, whom despite being a very busy guy was somehow always available to talk about level sets and DSD, thank you for making that summer wonderful.

To friends and colleagues: Jeremy Schmitt, my longest-running officemate, for putting up with me the past five years; Theresa Wong, for the times when I needed a break from math; everyone in the old 5412 office, for all our ridiculous conversations (even though it prevented us from getting work done); to those not in my year, especially to Rob Won and Mike Tait, who forced me to get back into the pool and “swam to Santa Barbara” with me.

To Mom and Dad, for their unconditional love and support, I obviously could not have done it without their encouragement. To Mindy, the greatest dog ever.

And lastly, to Sinan, who I actually only met at the beginning of grad school, but has been by me these past five years. Words cannot express how lucky I am that you came here instead of Davis. I got through the hardest moments because of you; thanks for being my best friend.

Chapters 1 - 3, in part, and Appendix B, in full, is currently being prepared for submission for publication. This work is coauthored with Holst, Michael;
Lindblom, Lee.

Chapters 5 - 7, in part is currently being prepared for submission for publication. This work is coauthored with Dobrev, Veselin; Kolev, Tzanzio; Rieben, Robert; Tomov, Vladimir.

A portion of this work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.
VITA

2012 B. S. in Applied Mathematics, University of California, Los Angeles

2014 M. A. in Applied Mathematics, University of California, San Diego

2017 Ph. D. in Mathematics, University of California, San Diego

PUBLICATIONS


ABSTRACT OF THE DISSERTATION

Computational techniques in molecular dynamics and detonation shock dynamics

by

Francesca Catherine Grogan

Doctor of Philosophy in Mathematics

University of California, San Diego, 2017

Professor Michael Holst, Chair

This thesis comprises of the study of two physical problems. In the first half, we look at molecular dynamics (MD) simulation, which is widely used to study the motion and thermodynamic properties of molecules, and is applicable to a variety of problems in biochemistry, physics, and other fields. Computational limitations and the complexity of problems, however, result in the need for error quantification. We examine the inherent two-scale nature of MD to construct a large-scale dynamics approximation where internal motion of the atoms are approximated. This approximation is useful for evaluating the differences between full, classical MD simulations and those based on large-scale approximation schemes. We provide numerical results examining error in momenta, energy, and the macroscopic variables
used in these large-scale dynamics as a means of error estimation. Accuracy of the macroscopic variables varies depending on the particular user-chosen approximation for the individual atomic motion. Our approximation conserves momenta and, although energy is not explicitly set to be conserved, fractional error is small.

In the second half, we turn our attention to another differential equation – the level set equation, which is a popular approach to modeling evolving interfaces. The solver’s ability to implicitly track moving fronts lends itself to a number of applications; in particular, our focus is on modeling high-explosive (HE) burn and detonation shock dynamics (DSD). We present a level set advection solver in two and three dimensions using the discontinuous Galerkin method with high-order finite elements. During evolution, the level set function is reinitialized to a signed distance function to maintain accuracy. Our approach leads to stable front propagation and convergence on high-order, curved, unstructured meshes. We provide results for two- and three-dimensional benchmark problems as well as applications to DSD.
Chapter 1

Introduction

1.1 Overview of thesis

This thesis consists of two projects, both presenting computational techniques for the study of differential equations with applications to physical problems.

Immediately after this overview, Section 1.2 introduces the first project of this thesis, which studies error quantification for large-scale molecular dynamics (MD) approximation. In Sections 1.2.1 - 1.2.4, we provide related background knowledge regarding the MD equations, the Dormand and Prince numerical ordinary differential equation (ODE) solver we use, particular ensemble averages we calculate, and a brief overview of approximation methods and why they are used. In Appendix A, we list the function evaluations needed to solve for the coefficients for the Dormand and Prince ODE integrator used later in numerical results.

Chapter 2 is an exposition on the theoretical development and derivation of our class of approximations, which we call the large-scale dynamics approximation. We begin in Section 2.1 by building the equation for atom position. Along with introducing our notation, these preliminaries will provide the foundation for our large-scale approximation. As we approximate the internal atom deviations, Section 2.2 explains how the theory of normal mode analysis is used for this purpose. In particular, we describe atom deviations using a linear combination of the molecule’s normal modes. Appendix B includes a more detailed discussion of the zero-frequency normal modes associated with the translation and rotation eigenvectors.
Section 2.3 traces the development of our large-scale approximation. First, we describe our original method (discussed in Section 2.3.1 and fully derived in Appendix C) which is based upon the idea that we could evolve some of the normal mode amplitudes analytically while evolving others numerically using an exact representation. We will see that this did not produce ideal results, and use these revelations to formulate our large-scale dynamics approximation. This approximation scheme, derived in Section 2.3.2, allows for user-chosen mode amplitude approximation; thus it is actually a class of schemes. We derive evolution equations for the macroscopic variables (in particular, we use unit quaternions to evolve our rotations as they are more numerically stable – see Appendix D for details). Section 2.4 briefly discusses analytic error bounds for our approximation.

Chapter 3 contains numerical results for our large-scale approximation. These results focus on error quantification between our approximation and classical MD. To illustrate that our large-scale dynamics approximation yields a class of schemes, we include comparisons between two particular choices of mode amplitude approximation: one where these amplitudes are set to zero, and one where we evolve them sinusoidally. In Section 3.1, we discuss the setup of our numerical simulations and present the results in Section 3.2. Appendix E discusses converting between Cartesian variables and normal mode variables. We conclude in Chapter 4 and summarize our results for the first project of this thesis.

Chapter 5 is an introduction to the second project of this thesis, which studies level sets for detonation shock dynamics (DSD) using high-order finite elements. We briefly review relevant background for this topic, with Section 5.1 containing a brief introduction to level sets. We also discuss reinitialization, which we use for stability purposes, along with other approaches used by the level set community. Lastly, in Section 5.2, we explore applications of level sets to DSD for the purposes of modeling high-explosive burn.

Chapter 6 presents the details of our level set algorithm. Section 6.1 provides the finite element spatial discretization, which uses the discontinuous Galerkin method. Section 6.2 provides a description of the time integrator used for both level set advection and redistancing, and gives the overall algorithm. Chapter 7
is a discussion of the numerical results for the level set method using high-order finite elements. Here, we include typical level-set benchmarks as well as more complex front propagation problems involving corner turning and shadow surfaces. In Chapter 8, we conclude the second project of this thesis, and mention ideas for future work.

1.2 Introduction to error quantification for large-scale molecular dynamics (MD)

Molecular dynamics (MD) is a popular tool that uses Newton’s equations to simulate the evolution of atoms and molecules. MD simulations can be applied to a variety of problems, such as drug discovery [23], in many fields including biochemistry and physics. MD has been studied by application scientists as well as mathematicians for several decades. Some standard references for numerical methods for MD include [3, 4, 14, 26, 35, 42].

In this first part of the thesis, we examine whether it is possible to exploit the natural two-scale structure in MD for method development and uncertainty quantification (UQ), the quantification of uncertainty and error in a given system [67]. UQ is a growing field used to address the accuracy of multiscale and multiphysics modeling techniques, as well as numerical methods. Given the wide use of computer simulations in scientific research, it is important to understand and look to control sources of error inherent in mathematical models. Potential sources of error are numerous, including (but not limited to) parameter uncertainty, errors in experimental measurements and gathering of data, and algorithmic uncertainty. To address these issues, UQ has enabled researchers to better quantify how sources of uncertainty affect error in their MD models [48, 55, 57, 58].

With this goal of error quantification in mind, we construct a large-scale approximation that utilizes the multi-scale nature of molecular evolution. Namely, the movement of a molecule is governed by large-scale, rigid rotations and translations, and also by small deviations of the individual atoms. As we use an exact representation for the rotations and translations, we only approximate the devia-
tions of the atoms. In particular, we approximate these internal movements using
the theory of normal modes, which was first introduced by Wilson [74] in the
early 1940’s. Overall, our large-scale approximation consists of an equation for
the atom positions in terms of variables describing rotation, translation, and the
molecules normal modes, along with the evolution equations for these variables.
This large-scale dynamics scheme allows for the user to specify the normal mode
amplitude evolution, which dictates atomic motion. In this way, our large-scale
dynamics approximation in fact yields a class of approximation schemes.

We then apply this approximation class to quantify various sources of
error. First, we study error in the macroscopic variables between full, classical
MD simulations and approximations focused on large-scale dynamics. Because
the atom positions are given in terms of rotational, translational, and vibrational
movement, it is easier to quantify the contributed error from each part, which will
be shown through numerical results. As we will see, position and velocity variables
are accurately approximated, while the accuracy of spatial orientation variables
depends on the choice of mode amplitude evolution.

Error quantification is also particularly useful in the analysis of MD ap-
proximation schemes. Such approximations are primarily geared toward efficiency,
such as constraint algorithms like SHAKE [59] or RATTLE [5], which fix the
fastest vibrational frequencies to allow for larger timesteps. Our scheme similarly
allows for analytical approximations for the vibrations; thus the error estimates
given with our large-scale approximation can be used as a baseline comparison for
these approximations. To illustrate this, we compare in the numerical results our
large-scale scheme when fixing atomic evolution versus evolving them sinusoidally.

Lastly, since MD simulations are often performed to gather statistical
averages of certain observables, it is important to study the error in these quantities.
In particular, while it is well-known that $n$-body systems exhibit chaotic behavior
in which divergent trajectories result from small perturbations in initial conditions
[49], statistical averages are still valid under the assumption of ergodicity [4, 14, 35].
With this in mind, we study how accurately our two-scale method captures some
of these statistical quantities. We compute momenta and energy error, and our
numerical results show our large-scale method is momentum-conserving. Energy conservation varies with choice of mode amplitude evolution, but in the worst-case still exhibits only small fractional changes.

The first half of this thesis contains four chapters and is organized as follows. In the remaining sections of this chapter, we further discuss background material. In Chapter 2, we derive the large-scale molecular dynamics approximation to a classical MD system. We implement our method and provide numerical results in Chapter 3, and conclude in Chapter 4. We summarize the main results of this first project below.

- We construct a large-scale dynamics approximation which can be used to evaluate differences between classical MD simulations and those based on our scheme.
- We derive analytic error bounds for this large-scale approximation.
- Implementing code to evaluate differences between classical MD equations and those based on the large-scale approximation, we find:
  - Our method performs well in determining the position and velocity variables.
  - The accuracy of rotation and angular velocity variables varies when varying mode amplitude evolution.
  - Our method is momentum-conserving and keeps fractional change in energy small.

1.2.1 The MD equations

Classical molecular dynamics simulations involve numerically evolving a system of atoms and molecules using Newton’s second law. That is, letting \( \vec{x}_{A_i}(t) \) denote the location of atom \( A \) in the \( i \)th molecule, we evolve

\[
m_{A_i} \frac{d^2 \vec{x}_{A_i}}{dt^2} = -\frac{\partial U(\vec{x}_{C_i})}{\partial \vec{x}_{A_i}},
\]  

(1.1)
where $m_A$ is the mass of atom $A$ and $U(\vec{x}_{Ai})$ is the inter-atomic energy potential.

Classical molecular dynamics simulation enjoys a rich history. One of the earliest known MD simulation of particles was performed by Fermi, Pasta, Ulam, and Tsingou in the 1950’s. Using the MANIAC I computer at Los Alamos National Laboratory, they studied a one-dimensional system of 64 particles [25]. In 1957, Alder and Wainwright used MD to study phase transitions for hard spheres [2], and in 1964, Rahman produced results for atoms in liquid argon [53]. As computing power increased (Rahman’s simulations used 800 more particles than that of Fermi et. al.), more complex structures were open to simulation. McCammon, Gelin, and Karakuls produced protein simulations in 1977 with the bovine pancreatic trypsin inhibitor [41]. Today, MD is ubiquitous in many scientific disciplines.

1.2.2 Solving the MD equations

While the MD equations, i.e. Newton’s law, are quite simple to state, their numerical solution can be quite complicated. Consequently, this has led to the ongoing development of several numerical methods [61, 72, 71]. Even with modern supercomputers, computation power is limited, and choices must be made regarding simulation specifications, such as:

- **size** – most realistic systems involve large numbers of particles, and molecules can be very complex, such as in runs involving proteins. In particular, force evaluations on nonlinear potential functions are often time-consuming in MD simulations and thus are an important factor to consider when determining simulation size.

- **duration** – simulations must be run long enough for results to hold weight, and typical simulations can be on the order of nanoseconds (1E-9) to microseconds (1E-6). Care must be taken, however, to make sure the task terminates after a feasible amount of time.

- **timestep** – as an added hindrance to the issue of duration, timesteps must be small enough in order to avoid discretization error and ensure accurate calculation of the force term. Generally, this means the timestep should be
taken on the order of the fastest molecular vibration – usually on the order of femtoseconds (1E-15). Note that, in comparison with the feasible duration times mentioned above, this is orders of magnitude smaller. To balance the need for accuracy and efficiency, methods that are primed for efficiency can be chosen to deal with this.

In addition to the above specification considerations, there are also many MD algorithms available to choose from. In choosing such algorithms, a primary consideration for us is maintaining the integrity of certain thermodynamic properties of the simulation. Namely, we develop our large-scale dynamics approximation in Chapter 2 to conserve momenta. To stably evolve this system, we use an eighth-order Dormand and Prince integrator [29], which is an explicit Runge-Kutta method. Given stepsize $h$ and coefficients $c_i$, $b_i$, and $a_{ij}$, a Runge-Kutta method solves the initial value problem

$$ y'(t) = f(t, y(t)), \quad y(t_0) = y_0 \quad (1.2) $$

through the following steps:

$$ y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i, \quad (1.3) $$

where

$$ k_1 = f(t_n, y_n), $$
$$ k_2 = f(t_n + c_2 h, y_n + h(a_{21} k_1)), $$
$$ \vdots $$
$$ k_s = f(t_n + c_s h, y_n + h(a_{s1} k_1 + a_{s2} k_2 + \cdots + a_{s,s-1} k_{s-1})), \quad (1.4) $$

are the $s$ stages. The eighth-order Dormand and Prince integrator has 13 stages with 12 function evaluations per stage. The coefficients $c_i$, $b_i$, and $a_{ij}$ are described in Appendix A, and the code is freely available at [28]. In addition to using adaptive timesteps to control error, this method also has embedded fifth- and third-order formulas which describe error estimation. The error estimate $\text{err}$ [52] is calculated as

$$ \text{err} = \text{err}_5 \frac{\text{err}_5}{\sqrt{(\text{err}_3)^2 + 0.01(\text{err}_5)^2}}, \quad (1.5) $$
where err$_3$ and err$_5$ are the errors computed using the embedded 3rd and 5th-order methods, respectively. This Dormand and Prince integrator allows for control of both absolute error, given by the tolerance atol, and relative error, given by the tolerance rtol. Given a function $\tilde{y}$, rtol keeps the local error of $y(i)$ below rtol*|$y(i)$|+atol by adjusting its own internal time step. That is, if the estimated error is larger than the prescribed tolerance, the original timestep is rejected and the integrator repeats the calculation with a smaller timestep [60]. Refer to Hairer et al. [29] for more about error control in this method.

### 1.2.3 Ensemble computations

The above integrator, along with other algorithms, computes the position and velocity of the particles in the system. With that being said, it is well known that n-body systems are chaotic [49]; that is, identical systems with slight perturbations in initial conditions will have divergent trajectories. Assuming ergodicity, however, the thermodynamic properties of the system are not dependent on its initial conditions, and statistical averages are valid [4, 14, 35]. A primary motivation behind most MD simulations is to gain information on the statistical averages of certain quantities in a given system. As such, one possibility for error quantification of MD simulations should be focused on these quantities, rather than the trajectories themselves. As will be seen in Section 3, we look at the numerical error in three statistical quantities: linear and angular momenta, and energy.

Let $A(z(t))$ be some function where $z(t)$ is a trajectory of our system with initial conditions $z(0) = z_0$. We can define the time or trajectory average as

$$\langle A \rangle_T := \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau A(z(t))dt,$$  \hspace{1cm} (1.6)

where $\tau$ is the duration time. The ensemble average is defined as

$$\langle A \rangle_{\text{ens}} := \int_{\Omega} A(z)\rho_{\text{ens}}(z)dz,$$  \hspace{1cm} (1.7)

where $\rho_{\text{ens}}$ is a probability density function which describes the distribution of the configurations in our ensemble. An important assumption commonly made is that the system is ergodic, and so averages taken along trajectories, Eq. (1.6), reproduce
ensemble averages, Eq. (1.7), or

\[ \langle A \rangle_T = \langle A \rangle_{\text{ens}}. \]  

(1.8)

Therefore, we can compute ensemble averages by averaging along the computed trajectories. Below we define the quantities that are studied in the numerical results in Chapter 3.

Computable quantities

- **Energy, \( E \):** total energy is the sum of kinetic energy and potential energy in the system, i.e.,

\[ E = E_{\text{kin}} + E_{\text{pot}} = \frac{1}{2} \sum_{i=1}^{N_M} \sum_{A=1}^{N_A} m_{A_i} \left( \frac{d\vec{x}_{A_i}}{dt} \cdot \frac{d\vec{x}_{A_i}}{dt} \right) + U, \]

(1.9)

where \( N_M \) is the number of molecules and \( N_A \) is the number of atoms in the system.

- **Linear momenta, \( \vec{P} \):**

\[ \vec{P} = \sum_{i=1}^{N_M} \sum_{A=1}^{N_A} m_{A_i} \frac{d\vec{x}_{A_i}}{dt}, \]

(1.10)

- **Angular momenta, \( \vec{J} \):**

\[ \vec{J} = \sum_{i=1}^{N_M} \sum_{A=1}^{N_A} m_{A_i} \vec{x}_{A_i} \times \frac{d\vec{x}_{A_i}}{dt}. \]

(1.11)

1.2.4 Approximating methods

In addition to studying error in statistical averages, another focus in the first half of this thesis is to perform error quantification to compare classical, full MD simulations (i.e. those based on Newton’s second law), and approximation methods. MD approximation methods were derived, unsurprisingly so, as response to the issues listed in Section 1.2.2. Our large-scale dynamics approximation, to be developed in Chapter 2, can also be used to study differences between approximation methods. As we will see, our approximation yields a class of approximation schemes
obtained by varying the choice of evolution of the normal mode amplitudes (mode amplitudes and their use will be explained in more detail in Chapter 2). Numerical results in Chapter 3 will show an example of this by performing a comparison between two different amplitude evolution approximations. We include these results to provide an example of how our large-scale dynamics method can be considered as a baseline with which to compare more widely-used approximation methods. Although we do not compare our approximation with these methods in the numerical results, for expository purposes we briefly mention some popular methods below.

**Constraint algorithms**

As mentioned, timesteps of classical MD methods typically depend on the fastest vibrational frequencies, whose small size poses numerical challenges. As such, constraint algorithms arose to circumvent this issue. Treating the molecules as rigid bodies, constraint methods fix bond lengths and thus fix the corresponding vibrations. Given a set of $n$ constraints $g_k(t)$, Eq. (1.1) can be formulated as

$$m_{A_i} \frac{d^2 x_{A_i}}{dt^2} = -\frac{\partial}{\partial x_{A_i}} \left( U + \sum_{k=1}^{n} \lambda_k g_k \right),$$  

where $\lambda_k$ are the Lagrange multipliers. The positions at time $t + \Delta t$ are given by

$$\vec{x}_{A_i}(t + \Delta t) = \hat{x}_{A_i}(t + \Delta t) - (\Delta t)^2 \sum_{k=1}^{n} \lambda_k \frac{\partial g_k}{\partial \vec{x}_{A_i}},$$  

where $\hat{x}_{A_i}(t + \Delta t)$ is the position obtained from the unconstrained equations of motion. The Lagrange multipliers $\lambda_k$ are chosen such that $g_k(t + \Delta t) = 0$. There are several algorithms that use this overall method. One well-known method is SHAKE [59], which uses the Verlet algorithm [72] as its integrator. SHAKE itself has many variants, such as RATTLE [5], which substitutes the Verlet algorithm with Velocity Verlet [71] and employs a second set of Lagrange multipliers.

**Coarse-grained models [32]**

Coarse-grain modeling is a form of surrogate modeling and addresses efficiency by reducing simulation size. These models are particularly useful for
complex biomolecules such as proteins. The reduced representations come in part by using “pseudoo-atoms” to represent groups of atoms. Note that parameters must be adjusted to properly account for several atoms now being represented as a single “atom”.

Having given a brief introduction to MD simulation, important computable quantities, and MD approximations, we are ready to introduce our large-scale dynamics approximation. We will use this approximation as a basis for various error analyses later on.

**Acknowledgement**: Chapter 1 in part is currently being prepared for submission for publication. This work is coauthored with Holst, Michael; Lindblom, Lee.
Chapter 2

Large-scale molecular dynamics

In this chapter, we develop our large-scale molecular dynamics approximation. This approximation takes advantage of the two-scale nature of MD behavior, which consists of large-scale overall rotations and translations of the molecule, and small-scale internal movements of the individual atoms. In particular, the small-scale movements of the atoms are those subject to approximation. We will later perform error analysis in Section 2.4 and Chapter 3 using the derived equations from this section. Overall, this chapter provides the theory and development of this large-scale approximation and lays the groundwork for the numerical results seen in the next chapter.

2.1 Preliminaries

We first introduce variables and equations relevant to our study of large-scale dynamics. Primarily, we write the atom positions in a molecule in a way that allows us to easily separate the multi-scale behavior seen in MD.

Let \( \vec{x}_A(t) \) denote the locations of the atoms in a single molecule. For the reminder of this discussion we drop the index \( i \), which denoted the \( i \)th molecule, for readability purposes. Thus, we are considering only a single molecule for the ensuing derivation, but we remark that it can easily be extended to the case where there are multiple molecules. As discussed in Chapter 1, the classical MD method
numerically solves Newton’s equation of motion. Here, we write this as:

\[
m_A \frac{d^2 \vec{x}_A}{dt^2} = - \frac{\partial U(\vec{x}_C)}{\partial \vec{x}_A}, \tag{2.1}
\]

where \(m_A\) is the mass of atom \(A\) and \(U(\vec{x}_A)\) is the inter-atomic energy potential. The center of mass \(\vec{x}_{CM}(t)\) of this molecule is given by

\[
\vec{x}_{CM}(t) = \frac{\sum_A m_A \vec{x}_A(t)}{M}, \tag{2.2}
\]

where \(M = \sum_A m_A\) is the total mass of the molecule. As a first step towards a large-scale dynamics construction, let \(\Delta \vec{X}_A(t)\) denote the location of atom \(A\) relative to the center of mass of the molecule:

\[
\Delta \vec{X}_A(t) = \vec{x}_A(t) - \vec{x}_{CM}(t). \tag{2.3}
\]

We use a time dependent rotation matrix \(R(t)\) to describe the spatial orientation of the molecule. We then separate the large-scale and small-scale interactions by representing the relative positions \(\Delta \vec{X}_A\) of these atoms as this rotation matrix acting on the sum of the equilibrium positions of the atoms, \(\vec{x}_{0A}\), plus deviations \(\delta \vec{x}_A(t)\)—which may or may not be small—away from these equilibrium positions:

\[
\Delta \vec{X}_A(t) = R(t) \cdot [\vec{x}_{0A} + \delta \vec{x}_A(t)]. \tag{2.4}
\]

From the definitions of \(\vec{x}_{CM}\) in Eq. (2.2) and \(\Delta \vec{X}_A\) in Eq. (2.3), it follows that

\[
0 = \sum_A m_A \Delta \vec{X}_A = R \cdot \sum_A m_A (\vec{x}_{0A} + \delta \vec{x}_A). \tag{2.5}
\]

Without loss of generality, we can express the equilibrium positions of the atoms \(\vec{x}_{0A}\) in a frame located at the center of mass; therefore,

\[
0 = \sum_A m_A \vec{x}_{0A} = \sum_A m_A \delta \vec{x}_A. \tag{2.6}
\]

The angular velocity of the molecule, which we denote as \(\vec{\Omega}\), can be defined as the time derivative of the rotation matrix \(R(t)\) in the following way. First, the
angular velocity matrix, \( \mathbf{\Omega} \), is given by the time derivative of \( \mathbf{R}(t) \) multiplied by the inverse of the rotation:

\[
\mathbf{\Omega} = -\frac{d\mathbf{R}}{dt} \cdot \mathbf{R}^{-1}.
\]  

(2.7)

Since \( \mathbf{\Omega} \) is anti-symmetric, it can be written as the dual of a vector:

\[
\mathbf{\Omega}_{ij} = \epsilon_{ijk} \Omega^k,
\]  

(2.8)

where \( \epsilon_{ijk} \) is the totally anti-symmetric tensor with \( \epsilon_{xyz} = 1 \) in Cartesian coordinates. Eqs. (2.7) and (2.8) therefore define the angular velocity \( \mathbf{\Omega} \). These expressions can then be used to obtain an expression for \( \mathbf{x}_A \):

\[
\mathbf{x}_A(t) = \mathbf{x}_{CM}(t) + \mathbf{R}(t) \cdot [\mathbf{x}_0 + \delta \mathbf{x}_A(t)],
\]  

(2.9)

and the velocities of the atoms in a single molecule, \( d\mathbf{x}_A/dt \):

\[
\frac{d\mathbf{x}_A}{dt} = \frac{d\mathbf{x}_{CM}}{dt} + \mathbf{R} \cdot \frac{d\delta \mathbf{x}_A}{dt} + \mathbf{\Omega} \times \Delta \mathbf{X}_A.
\]  

(2.10)

Writing the atom positions \( \mathbf{x}_A \) as in Eq. (2.9) is beneficial for error quantification purposes. Not only is it easier to identify error arising from vibrational, rotational, and translational motion, but it allows one to separate and exploit the two-scale structure of MD behavior. Here, the variables \( \mathbf{x}_{CM}, \mathbf{R}, \) and \( \mathbf{\Omega} \) represent large-scale behavior and the variable \( \delta \mathbf{x}_A \) describes the behavior of the internal deviations of the molecule. We can then use these variables to study error in large-scale MD approximation schemes in Chapter 3.

2.2 Representation of \( \delta \mathbf{x}_A \) using normal mode analysis

Now that we have an appropriate way to represent the two-scale behavior of MD, we discuss representing the internal deviations, \( \delta \mathbf{x}_A \), with normal modes. Normal modes afford us a simple way to approximate the internal, small-scale motion of the atoms while still numerically evolving the bulk rotation and translation of the molecule. We note that normal mode analysis for MD has been previously used by Wilson Jr. [74], Levitt et al. [36], Dauber-Osguthorpe et al. [20], and others.
The vectors $\delta \vec{x}_A$ measure the displacements of the atoms in a molecule away from their ground state equilibrium locations. These displacements, in an isolated molecule that has been perturbed slightly away from its equilibrium state, can most succinctly be described as a linear combination of the normal modes of the molecule. We use the notation

$$\delta \vec{x}_A = \sum_{\mu} A_0^\mu \sin(\omega_\mu t + \phi_\mu) \vec{e}_A^\mu, \quad (2.11)$$

to describe this expansion of $\delta \vec{x}_A$. A single mode, labeled by the index $\mu$, has frequency $\omega_\mu$, amplitude $A_0^\mu$, and phase $\phi_\mu$. These quantities are real constants for these modes. The vectors $\vec{e}_A^\mu$ describe the relative displacements of the atoms in this mode. These normal modes are solutions of the classical MD equations of motion, under the assumption that the mode amplitudes $A_0^\mu$ are small, and the angular velocity $\Omega$ is much smaller than the mode frequencies $\omega_\mu$. In this case, Newton’s equation, Eq. (2.1), reduces to

$$0 = -m_A \omega_\mu^2 \vec{e}_A^\mu + \sum_B \frac{\partial^2 U}{\partial \vec{x}_B \partial \vec{x}_A} \cdot \vec{e}_B^\mu, \quad (2.12)$$

where the Hessian matrix of the potential $U(\vec{x}_A)$, $\partial^2 U/\partial \vec{x}_B \partial \vec{x}_A$, is to be evaluated at the equilibrium state of the molecule where $\vec{x}_A = \vec{x}_{0A}$. Eq. (2.12) is the eigenvalue problem for this Hessian matrix. Since this matrix is real and symmetric, the eigenvalues $\omega_\mu^2$ and eigenvectors $\vec{e}_A^\mu$ will also be real. The eigenvalues, $\omega_\mu^2$, will be non-negative for any stable molecule. Except for the six zero-frequency modes that correspond to rigid rotations and translations of the molecule, these eigenvalues will also be strictly positive. Since the Hessian matrix is symmetric, the eigenvectors $\vec{e}_A^\mu$ form a complete basis for the $\delta \vec{x}_A$. It is easy to show that these eigenvectors satisfy (or in the case of degenerate eigenvalues, can be chosen to satisfy – see Appendix B) the following orthogonality and normalization condition:

$$\delta^{\mu \nu} = \sum_A \frac{m_A}{M} \vec{e}_A^\mu \cdot \vec{e}_A^\nu, \quad (2.13)$$

where $\delta^{\mu \nu}$ is the Kronecker delta. It follows that general $\delta \vec{x}_A$ can be expressed as a linear combination of these eigenvectors:

$$\delta \vec{x}_A = \sum_{\mu} A_\mu(t) \vec{e}_A^\mu, \quad (2.14)$$
where $A_\mu(t)$ is the general time-dependent amplitude for the $\mu$th normal mode. Note that since the eigenvectors form a complete basis, any $\delta \vec{x}_A$ can be expressed using Eq. (2.16), even when the mode amplitudes are not small. The amplitudes of the zero-frequency modes associated with the translation and rotation eigenvectors vanish identically, see Appendix B. Thus, we can exclude these zero-frequency modes when we expand $\delta \vec{x}_A$ as sums of normal modes, as in Eq. (2.16), since the degrees of freedom associated with these modes are already included in the variables $\vec{x}_{CM}$, $\mathbf{R}$, and $\vec{\Omega}$.

Now that we can express $\delta \vec{x}_A$ in terms of normal modes, we can rewrite Eq. (2.9) for the position variable $\vec{x}_A$ as:

$$\vec{x}_A = \vec{x}_{CM} + \mathbf{R} \cdot \left( \vec{x}_{0A} + \sum_\mu A_\mu \vec{e}_A^\mu \right). \quad (2.15)$$

Here, an important issue concerns how to approximate these general mode amplitudes $A_\mu$. In the next section, we address this and continue developing our approximation.

### 2.3 Evolution equations for large-scale dynamics

Continuing with our description of the large-scale dynamics approximation, we now turn our attention to the equations needed to evolve this system in time. The discussion proceeds similarly to the development of the large-scale scheme. We begin briefly with our original derivation, based on a normal mode basis representation, which evolves certain normal mode amplitudes using an exact representation and approximates the highest-frequency modes. The primary goal of this approximation is to increase computational efficiency (by analytically approximating the high-frequency modes) while maintaining high levels of accuracy (by numerically evolving modes deemed important to the molecule’s motion). As we will see, this goal could not be achieved due to the sensitivity of these $n$-body systems, as approximating even just a single mode analytically resulted in similar results as approximating all of the modes. Searching for fixes to these issues leads us to the current approximation scheme we use, called the large-scale dynamics approximation. In light of the
performance of the normal mode basis representation approximation, all modes are evolved analytically with our scheme. In particular, the approximation for mode amplitude evolution can be chosen by the user; as such this approximation can be viewed as a class of schemes. We fully derive the evolution equations for macroscopic degrees of freedom $\vec{x}_{CM}$, $\vec{R}$, and $\vec{\Omega}$, as well as discuss some possible options for the evolution of an approximation of the mode amplitudes $A_\mu$. The approximate mode amplitudes in turn describe the intrinsic degrees of freedom $\delta \vec{x}_A$. We end with basic analytical bounds for the variables in this scheme.

2.3.1 Approximation using a normal mode basis representation

Here, we describe our first attempt at an approximation scheme, which we will modify when constructing our final large-scale dynamics approximation scheme in the next section. This first approximation scheme was conceived after noting that the simplest version of MD approximation simulates the macroscopic degrees of freedom of molecules, but approximates or neglects completely the dynamics associated with the internal motions of the individual atoms $\delta \vec{x}_A$ relative to their equilibrium positions. The main benefit to using these approximation methods is that numerical evolution of the high-frequency modes force the simulation timestep to be very small. However, when neglecting these degrees of freedom, accuracy is affected. The driving questions behind this method were:

- Can we develop a method and bound that determines which modes to evolve analytically – and which to evolve numerically – such that a best-case scenario of high efficiency and accuracy are achieved?
- Can this bound also be used for error quantification purposes?

As this discussion is simply a lead-in and motivator to our primary, more successful result – the large-scale dynamics approximation – our discussion of this original scheme is brief, and our focus here is on approximation of the mode amplitudes. We refer the interested reader to Appendix C which contains the full derivation of
the normal mode basis representation, including the evolution equations for the macroscopic variables.

**Mode amplitude approximation**

Recall Eq. (2.16), which gives the internal atom deviations $\delta \vec{x}_A$ as the following linear combination:

$$\delta \vec{x}_A = \sum_{\mu} A_{\mu}(t) \vec{e}_{A}^{\mu},$$

(2.16)

where $A_{\mu}(t)$ are the general mode amplitudes. The defining characteristic of this normal mode basis approximation is the numerical evolution of certain modes using an exact representation, and analytical evolution of the rest. The approximate analytical evolution of the mode amplitudes is given by

$$\frac{d^2 A_{\mu}}{dt^2} = -\omega_{\mu}^2 A_{\mu},$$

(2.17)

which has a simple sinusoidal solution, c.f. Eq. (2.11). In contrast, the exact representation is complicated, given by

$$\frac{d^2 A_{\mu}}{dt^2} + \sum_{\nu} S^{\mu\nu} \frac{dA_{\nu}}{dt} + \sum_{\nu} T^{\mu\nu} A_{\nu} = F^\mu,$$

(2.18)

where $S^{\mu\nu}, T^{\mu\nu},$ and $F^\mu$ are coefficients (see Appendix C for details). We can rewrite the above expression as

$$\frac{d^2 A_{\mu}}{dt^2} = -\omega_{\mu}^2 A_{\mu} + \mathcal{E}^\mu,$$

(2.19)

where $\mathcal{E}^\mu$ is the mode force error measure:

$$\mathcal{E}^\mu = \omega_{\mu}^2 A_{\mu} + F^\mu - \sum_{\nu} S^{\mu\nu} \frac{dA_{\nu}}{dt} - \sum_{\nu} T^{\mu\nu} A_{\nu}.$$  

(2.20)

By monitoring this error measure, we can determine when the approximate amplitudes yield a sufficiently reasonable approximation, and thus evolve these analytically during simulation. Similarly, we can determine when a full numerical evolution is needed.
Upon testing this algorithm, it became clear the effects of the mode amplitudes were not what we had originally predicted. The idea that we could approximate several of the mode amplitudes while still maintaining accuracy failed as we found that even approximating a single mode gave similar results as approximating all modes. In particular, Fig. 2.1 shows energy and momenta during a 400ps run for both choices using an error tolerance of 1E-7. (details of our simulation setup will be discussed in Section 3.1). With this, we see there is little difference in varying the number of analytically-evolved mode amplitudes. This behavior is due in part by the chaotic nature of MD simulations (see Section 3.2.5 and in particular, Fig. 3.11). The sensitivity of MD simulations renders the idea of this scheme – that we can successfully approximate any mode amplitudes and still have very high accuracy – unattainable.

2.3.2 Large-scale dynamics approximation

Overall, the above results of the normal mode basis representation suggested that we should aim to derive a method that was more simple. This leads us to the development of the large-scale dynamics approximation. Here, we approximate
all the mode amplitudes, which simplifies computation while maintaining similar error thresholds as other approximation schemes. The evolution equations for the macroscopic variables, and the mode amplitudes, are provided in detail below.

Evolution equations for macroscopic variables

We begin with the evolution equations for the center of mass $\vec{x}_{CM}$, the angular velocity $\vec{\Omega}$, and the spatial orientation $\mathbf{R}$. We do so in a way which conserves momentum, as this feature is important to maintaining the integrity of most MD simulations. The total momentum $\vec{P}$ and angular momentum $\vec{J}$ of this molecule are defined as

\[
\vec{P} = \sum_A m_A \frac{d\vec{x}_A}{dt} = M \frac{d\vec{x}_{CM}}{dt},
\]

\[
\vec{J} = \sum_A m_A \Delta \vec{X}_A \times \frac{d\Delta \vec{X}_A}{dt}.
\]

Their time derivatives are given by

\[
\frac{d\vec{P}}{dt} = \sum_A m_A \frac{d^2\vec{x}_A}{dt^2},
\]

\[
\frac{d\vec{J}}{dt} = \sum_A m_A \Delta \vec{X}_A \times \frac{d^2\Delta \vec{X}_A}{dt^2}.
\]

Now, differentiating Eq. (2.10) once more, the accelerations of the atoms in a single molecule are

\[
\frac{d^2\vec{x}_A}{dt^2} = \frac{d^2\vec{x}_{CM}}{dt^2} + -\Delta \vec{X}_A \times \frac{d\vec{\Omega}}{dt} + \vec{B}_A,
\]

where $\vec{B}_A$ is given by

\[
\vec{B}_A = \sum_{\mu} \frac{d^2 A^\mu_A}{dt^2} \mathbf{R} \cdot \vec{c}_A^\mu + 2\vec{\Omega} \times \left( \mathbf{R} \cdot \frac{d\delta \vec{x}_A}{dt} \right)
\]

\[
+ \left( \vec{\Omega} \otimes \vec{\Omega} - \vec{\Omega} \cdot \vec{\Omega} \mathbf{I} \right) \cdot \Delta \vec{X}_A.
\]

We point out that $d\vec{x}_A/dt$ and $\vec{B}_A$ depend on the large scale variables $d\vec{x}_{CM}/dt$, $\mathbf{R}$, and $\vec{\Omega}$, but not on their time derivatives. Substituting these expressions into the equations for $d\vec{P}/dt$ and $d\vec{J}/dt$, it is straightforward to show that:

\[
\frac{d\vec{P}}{dt} = M \frac{d^2\vec{x}_{CM}}{dt^2},
\]

\[
\frac{d\vec{J}}{dt} = \sum_A m_A \Delta \vec{X}_A \times \frac{d^2\Delta \vec{X}_A}{dt^2}.
\]
\[
\frac{d\vec{J}}{dt} = \vec{J}_\Omega \cdot \frac{d\vec{\Omega}}{dt} + \vec{J}, \tag{2.28}
\]

where the matrix \(\vec{J}_\Omega\) and the vector \(\vec{J}\) are given by

\[
\vec{J}_\Omega = \sum_A m_A \left( \Delta \vec{X}_A \cdot \Delta \vec{X}_A \mathbf{I} - \Delta \vec{X}_A \otimes \Delta \vec{X}_A \right), \tag{2.29}
\]

\[
\vec{J} = \sum_A m_A \Delta \vec{X}_A \times \vec{B}_A. \tag{2.30}
\]

We note that \(\vec{J}_\Omega\) and \(\vec{J}\) depend on the large scale variables \(d\vec{x}_{CM}/dt\), \(\mathbf{R}\), and \(\vec{\Omega}\), but not on their time derivatives.

With momentum conservation in mind, we obtain the evolution equation for the center-of-mass motion of each molecule by setting the rate of change of the linear momentum equal to the total external force acting on the molecule:

\[
\frac{d\vec{P}}{dt} = -\sum_A \frac{\partial U}{\partial \vec{x}_A} = M \frac{d^2\vec{x}_{CM}}{dt^2}. \tag{2.31}
\]

Similarly we obtain equations for \(d\vec{\Omega}/dt\) by setting \(d\vec{J}/dt\) to its rate of change due to torque:

\[
\frac{d\vec{J}}{dt} = -\sum_A \Delta \vec{X}_A \times \frac{\partial U}{\partial \vec{x}_A} = \vec{J}_\Omega \cdot \frac{d\vec{\Omega}}{dt} + \vec{J}. \tag{2.32}
\]

The resulting equation for \(d\vec{\Omega}/dt\) can then be written as:

\[
\frac{d\vec{\Omega}}{dt} = -\left(\vec{J}_\Omega\right)^{-1} \cdot \left[ \vec{J} + \sum_A \Delta \vec{X}_A \times \frac{\partial U}{\partial \vec{x}_A} \right]. \tag{2.33}
\]

We remark that the potential terms in Eqs. (2.31) and (2.33) are evaluated at the current positions of the atoms using Eq. (2.15), which uses a normal mode approximation for the \(\delta \vec{x}_A\).

We use unit quaternions to numerically evolve rotation as opposed to \(3 \times 3\) rotation matrices. Using quaternions in lieu of rotation matrices has several advantages. Not only are quaternions easier to represent (4 numbers versus 9), but they can be easily constructed given an axis and angle of rotation. Furthermore, due to roundoff error, using matrices to compose rotations numerically does not guarantee the final matrix is orthogonal. In contrast, quaternions can simply be
renormalized to represent the rotation, whereas it can be difficult to convert a non-orthogonal matrix back to an orthogonal one.

So, we will describe rotations using unit quaternions $q(t) = q_0(t) + q_1(t)i + q_2(t)j + q_3(t)k$, where

$$0 = C \equiv q_0^2 + q_1^2 + q_2^2 + q_3^2 - 1. \tag{2.34}$$

The evolution equation using quaternions is simple:

$$\frac{dq(t)}{dt} = \frac{1}{2} \Omega(t)q(t), \tag{2.35}$$

where $\Omega(t)q(t)$ can be multiplied out using the Hamilton product. Note that the result, $dq(t)/dt$, will usually need to be normalized. For simplicity, we can make these evolution equations for $q_i$ automatically preserve the constraint, Eq. (2.34), by modifying the above evolution system in the following way:

$$\frac{dq(t)}{dt} = \frac{1}{2} \Omega(t)q(t) - \frac{1}{8} \eta q C, \tag{2.36}$$

where $\eta > 0$. Multiplying this out, we have the following:

$$\frac{dq_0(t)}{dt} = -\frac{1}{2}(\Omega_x q_1 + \Omega_y q_2 + \Omega_z q_3) - \frac{1}{8} \eta q_0 C, \tag{2.37}$$

$$\frac{dq_1(t)}{dt} = \frac{1}{2}(\Omega_x q_0 + \Omega_y q_3 - \Omega_z q_2) - \frac{1}{8} \eta q_1 C, \tag{2.38}$$

$$\frac{dq_2(t)}{dt} = \frac{1}{2}(-\Omega_x q_3 + \Omega_y q_0 + \Omega_z q_1) - \frac{1}{8} \eta q_2 C, \tag{2.39}$$

$$\frac{dq_3(t)}{dt} = \frac{1}{2}(\Omega_x q_2 - \Omega_y q_1 + \Omega_z q_0) - \frac{1}{8} \eta q_3 C. \tag{2.40}$$

This form of the equations implies the following evolution equation for the constraint:

$$\frac{dC}{dt} = -\eta(C + 1)C. \tag{2.41}$$

This version of the evolution equations will ensure that small constraint violations will be exponentially suppressed as the system evolves. If the constraint becomes negative, this suppression mechanism becomes somewhat less efficient. However, since this constraint is bounded below by $-1$, the constraints will always be driven back toward zero for any value of the quaternions $q_i$.

For more information regarding quaternions and rotations, see Appendix D or [33].
Evolution equations for the mode amplitude variables

We conclude our derivation of the evolution equations with a discussion of the evolution of the mode amplitudes $A_\mu(t)$. The following is a derivation for mode amplitude evolution using approximation. Once more, recall from Section 2.2 that the $\delta \vec{x}_A$ can be described as the following linear combination:

$$\delta \vec{x}_A = \sum_\mu A_\mu(t) \vec{e}_A^\mu,$$  \hspace{1cm} (2.42)

where $A_\mu(t)$ is the general time-dependent amplitude for the $\mu$-th normal mode. As noted, the approximation of $A_\mu(t)$ can vary. One choice is the following simple sinusoidal behavior, c.f. Eq. (2.11),

$$A_\mu(t) = A_0^\mu \sin(\omega_\mu t + \varphi_\mu).$$ \hspace{1cm} (2.43)

In particular, and as we will see in the numerical results, simply setting the mode amplitudes to zero, or, using Eq. (2.43), setting $A_0^\mu = 0$, yields the most accurate results. In essence, this means we are fixing the mode amplitudes to 0 for the entire evolution, thus the evolution of $\delta \vec{x}_A$ is also fixed to 0. However, as this large-scale scheme is really a class of approximations, we also test this method using nonzero $A_0^\mu$. For this, the simple sinusoidal motion still allows us to easily derive the following identities, which yield the evolution of the mode amplitudes, and thus, the atom deviations $\delta \vec{x}_A$, for this particular scheme:

$$\frac{dA_\mu}{dt} = \omega_\mu A_0^\mu \cos(\omega_\mu t + \varphi_\mu),$$ \hspace{1cm} (2.44)

$$\frac{d^2A_\mu}{dt^2} = -\omega_\mu^2 A_\mu,$$ \hspace{1cm} (2.45)

$$\frac{d\delta \vec{x}_A}{dt} = \sum_\mu \frac{dA_\mu}{dt} \vec{e}_A^\mu,$$ \hspace{1cm} (2.46)

$$\frac{d^2\delta \vec{x}_A}{dt^2} = \sum_\mu \frac{d^2A_\mu}{dt^2} \vec{e}_A^\mu.$$ \hspace{1cm} (2.47)

Both choices of this large-scale dynamics approximation will be studied in Chapter 3 on numerical results, although it is emphasized that other approximations for $A_\mu$, and thus $\delta \vec{x}_A$, could be easily implemented with this type of scheme.
**Summary**

Below, we provide a brief recap of our large-scale dynamics approximation. We express the position of an atom, $\vec{x}_A$, in the molecule in the following way:

$$
\vec{x}_A = \vec{x}_{CM} + R \cdot \left( \vec{x}_{0A} + \sum_{\mu} A_{\mu} \vec{e}_A^\mu \right),
$$

(2.48)

where $\vec{x}_{CM}$ is the position of the center of mass of the molecule, $R$ is a rotation matrix that describes the spatial orientation of the molecule, $\vec{x}_{0A}$ is the equilibrium position of the atom labeled by the index $A$, $\vec{e}_A^\mu$ is one of the normal mode eigenvectors of the molecule, and $A_{\mu}$ is the amplitude of this $\mu$-th mode. The system of equations that determine the evolution of the dynamical variables can be written in the form:

$$
\frac{d^2 \vec{x}_{CM}}{dt^2} = -\sum_A \frac{1}{M} \frac{\partial U(\vec{x}_C)}{\partial \vec{x}_A},
$$

(2.49)

$$
\frac{dq(t)}{dt} = \frac{1}{2} \Omega(t) q(t) - \frac{1}{8} q qC,
$$

(2.50)

$$
\frac{d\vec{\Omega}}{dt} = -\left( \tilde{J}_\Omega \right)^{-1} \cdot \left[ \vec{J} + \sum_A \Delta \vec{X}_A \times \frac{\partial U}{\partial \vec{x}_A} \right],
$$

(2.51)

where the quantities $\tilde{J}_\Omega$ and $\vec{J}$ are defined in Eqs. (2.29) and (2.30) respectively. The approximation for the mode amplitudes $A_{\mu}$ are user-chosen. For a molecule with one equilibrium state, the normal mode eigenvectors, $\vec{e}_A^\mu$ and eigenvalues, $\omega_{\mu}^2$, that appear in these equations are determined once and for all by solving the eigenvalue problem given in Eq. (2.12) for the Hessian matrix of the inter-atomic energy potential.

### 2.4 Error quantification using large-scale dynamics approximation

Now that we have established the equations used for our large-scale dynamics approximation, we turn to exploiting them for the benefit of error quantification. In the next chapter, we will show how our approximation quantifies various sources of error through numerical simulation. Here, we briefly discuss analytical error bounds for the macroscopic variables and normal mode amplitudes.
In the previous sections, we carefully chose variables to describe atom position so as to allow for natural separation of molecular movement into the large overall rotations and translations, and the small inner deviations of the individual atoms. In classical MD simulations, atom position is taken into account as a single variable, which makes it difficult to extrapolate the complicated effects of rotational, translational, and vibrational motion. Instead, Eq. (2.15), which more clearly separates these, is better suited to studying some error effects. This falls under algorithmic, or numerical uncertainty, which tackles error stemming from numerical methods and approximations. More precisely, we will evaluate the differences between the full, classical molecular dynamics simulations, and those based on our large-scale approximation.

Our large-scale model uses exact representations to numerically evolve the large-scale variables the center of mass $\vec{x}_{CM}$, rotation $\vec{R}$, and angular velocity $\vec{\Omega}$. Below, we will derive error bounds for the position and velocity of the center of mass ($\vec{x}_{CM}$ and $d\vec{x}_{CM}/dt$), rotation, and angular velocity. We note that while these bounds are technically correct, they are a fairly poor indicator for the actual numerical error and have room for improvement.

For our center of mass error bound, denote the error between classical MD and large-scale solutions as $D_{\vec{x}_{CM}}$. We know from Eq. (2.31) that

$$\frac{d^2D_{\vec{x}_{CM}}}{dt^2} = -\sum_A \frac{1}{M} \mathcal{D}\left(\frac{\partial U}{\partial \vec{x}_A}\right), \quad (2.52)$$

where $\mathcal{D}(\partial U/\partial \vec{x}_A)$ similarly denotes the error in the force $\partial U/\partial \vec{x}_A$. We know there exists some time $0 \leq t_1 \leq t$ such that

$$\left|\sum_A \mathcal{D}\left(\frac{\partial U}{\partial \vec{x}_A(t)}\right)\right| \leq \max_{t_1} \left|\sum_A \mathcal{D}\left(\frac{\partial U}{\partial \vec{x}_A(t)}\right)\right| = \left|\sum_A \mathcal{D}\left(\frac{\partial U}{\partial \vec{x}_A(t_1)}\right)\right|. \quad (2.53)$$

Thus,

$$\left|\frac{d^2D_{\vec{x}_{CM}}}{dt^2}\right| \leq \frac{1}{M} \left|\sum_A \mathcal{D}\left(\frac{\partial U}{\partial \vec{x}_A(t_1)}\right)\right| = \hat{C}(t). \quad (2.54)$$
Integrating, we then have error bounds for the velocity of the center of mass
\[
\left| \frac{d\vec{x}_{CM}}{dt} \right| \leq \hat{C}(t), \quad (2.55)
\]
and its position,
\[
|\vec{D}_{\vec{x}_{CM}}| \leq \frac{1}{2} \hat{C}(t)t^2. \quad (2.56)
\]
The error in our spatial orientation variables are as follows. Denote the error in our quaternion values as
\[
\vec{D}q(t) = q_c(t) - q_\ell(t),
\]
where \(q_c(t)\) is the quaternion value in a classical MD system (or a system using exact representations), and \(q_\ell(t)\) is the quaternion value in our large-scale system. Then,
\[
\left\| \vec{D}q(t) \right\| = \left\| q_c(t) - q_\ell(t) \right\| \leq \left\| q_c(t) \right\| + \left\| q_\ell(t) \right\| < 2 (1 + \epsilon), \quad (2.57)
\]
where \(\left\| \cdot \right\|^2 := \frac{1}{t} \int_0^t (\cdot)^2 dt\), and the last inequality comes from the fact that our quaternion norms are forced to be within some \(\epsilon > 0\) of 1 during the entire evolution by Eq. (2.41).

As for the error in angular velocity, we similarly denote it as
\[
\vec{D}\Omega = \Omega_c - \Omega_\ell,
\]
where \(\Omega_c\) is the classical MD angular velocity and \(\Omega_\ell\) is the angular velocity of the large-scale system. Using Eq. (D.11) and applying norms, we get
\[
\left\| \vec{D}\Omega(t) \right\| = \left\| dq_c \right\|_2 (1 + \epsilon) + \left\| dq_\ell \right\|_2 (1 + \epsilon), \quad (2.58)
\]
where \(q^{-1}(t) = q_0(t) - q_1(t)i - q_2(t)j - q_3(t)k\), so \(\left\| q_c^{-1} \right\|\) and \(\left\| q_\ell^{-1} \right\|\) are also within \(\epsilon\) of 1. Now, there exists some \(t_1, t_2\) such that the max norms of the quaternion’s derivatives are achieved at \(0 \leq t_1, t_2 \leq t\), respectively. Thus,
\[
\left\| \vec{D}\Omega(t) \right\| < 2 (1 + \epsilon) \left( \left\| \frac{dq_c}{dt} \right\|_{t=t_1} + \left\| \frac{dq_\ell}{dt} \right\|_{t=t_2} \right). \quad (2.59)
\]
As the mode amplitudes \(A_\mu\) consist of the sole approximation in this scheme, in the following we establish a bound on the growth in the error in our analytical
solutions for the mode amplitudes. In order to do so, we need to use an exact normal mode basis representation – one that does not use an approximate formulation of the internal deviations $\delta \vec{x}_A$ – and thus produces the same dynamics as the classical MD equations. We provide the formulation in Appendix C, although the details are not necessary for this section. We can thus compare this exact representation with one using an approximation of the atoms, producing a way to quantify error when the other variables use exact evolution equations. We write the exact evolution of $\delta \vec{x}_A$ as

$$\frac{d^2 \delta \vec{x}_A}{dt^2} = \sum_{\mu} \frac{d^2 A_\mu}{dt^2} \vec{e}_\mu^A,$$  \hspace{1cm} (2.60)

where, in this case, the acceleration of the mode amplitudes, $\frac{d^2 A_\mu}{dt^2}$, are given by

$$\frac{d^2 A_\mu}{dt^2} = -\omega_\mu^2 A_\mu + \mathcal{E}^\mu(A_\mu, \frac{dA_\mu}{dt}, R, \vec{\Omega}, \frac{d\vec{\Omega}}{dt}),$$  \hspace{1cm} (2.61)

where $\mathcal{E}^\mu$ is the mode force error measure consisting of several terms (see Appendix C for details). This is in comparison with our analytic mode evolution given in the previous section, which behaves according to the ODE,

$$\frac{d^2 A_\mu}{dt^2} = -\omega_\mu^2 A_\mu.$$  \hspace{1cm} (2.62)

Letting $\mathcal{D}A_\mu$ be the error in our analytical mode solutions, we begin by rearranging Eq. (2.61) and multiplying by $d\mathcal{D}A_\mu/dt$:

$$d\mathcal{D}A_\mu \left( \frac{d^2 \mathcal{D}A_\mu}{dt^2} + \omega_\mu^2 \mathcal{D}A_\mu = \mathcal{E}^\mu \right).$$  \hspace{1cm} (2.63)

We obtain an exact derivative on the left-hand side:

$$\frac{1}{2} \frac{d}{dt} \left[ \left( \frac{d\mathcal{D}A_\mu}{dt} \right)^2 + \omega_\mu^2 \mathcal{D}A_\mu^2 \right] = \mathcal{E}^\mu \frac{d\mathcal{D}A_\mu}{dt}.$$  \hspace{1cm} (2.64)

Integrating, then invoking Cauchy-Schwarz yields

$$\left( \frac{d\mathcal{D}A_\mu}{dt} \right)^2 + \omega_\mu^2 \mathcal{D}A_\mu^2 = 2 \int_0^t \mathcal{E}^\mu \left| \frac{d\mathcal{D}A_\mu}{dt} \right| dt \leq 2t \| \mathcal{E}^\mu \| \left| \frac{d\mathcal{D}A_\mu}{dt} \right|.$$  \hspace{1cm} (2.65)
Next, we simplify things further. There exists some time \( 0 \leq t_1 \leq t \) such that
\[
\| \mathcal{E}^\mu(t) \| \leq \max \| \mathcal{E}^\mu(t) \| = \| \mathcal{E}^\mu(t_1) \|. \tag{2.66}
\]
Also noting that \( \left\| \frac{d \mathcal{D} A^\mu}{dt} \right\| \leq \max \left| \frac{d \mathcal{D} A^\mu}{dt} \right| \), there will exist some time \( 0 < t_2 \leq t \) such that
\[
\left| \frac{d \mathcal{D} A^\mu}{dt} \right|_{t=t_2} = \max \left| \frac{d \mathcal{D} A^\mu}{dt} \right| . \tag{2.67}
\]
Starting with Eq. (2.65), we have
\[
\left( \frac{d \mathcal{D} A^\mu}{dt} \right)^2 \leq 2t \| \mathcal{E}^\mu \| \left\| \frac{d \mathcal{D} A^\mu}{dt} \right\|. \tag{2.68}
\]
Using Eq. (2.66):
\[
\left( \frac{d \mathcal{D} A^\mu}{dt} \right)^2 \leq 2t \| \mathcal{E}^\mu(t_1) \| \left\| \frac{d \mathcal{D} A^\mu}{dt} \right\|, \tag{2.69}
\]
then evaluating at \( t_2 \) and using Eq. (2.67):
\[
\left( \frac{d \mathcal{D} A^\mu}{dt} \right)^2 \bigg|_{t=t_2} \leq 2t_2 \| \mathcal{E}^\mu(t_1) \| \left\| \frac{d \mathcal{D} A^\mu}{dt} \right\|_{t=t_2} \leq 2t \| \mathcal{E}^\mu(t_1) \| \left\| \frac{d \mathcal{D} A^\mu}{dt} \right\|_{t=t_2}. \tag{2.70}
\]
It follows that
\[
\left| \frac{d \mathcal{D} A^\mu}{dt} \right|_{t=t_2} \leq 2t \| \mathcal{E}^\mu(t_1) \|, \tag{2.71}
\]
and we can bound \( \mathcal{D} A^\mu \) by
\[
| \mathcal{D} A^\mu(t)| \leq | \mathcal{D} A^\mu(0)| + t^2 \| \mathcal{E}^\mu(t_1) \| = t^2 \| \mathcal{E}^\mu(t_1) \|, \tag{2.72}
\]
given the initial condition \( \mathcal{D} A^\mu(0) = 0 \).

Similarly, referring back to Eq. (2.65), we also have
\[
\omega^2 \mathcal{D} A^2 \leq 2t \| \mathcal{E}^\mu \| \left\| \frac{d \mathcal{D} A^\mu}{dt} \right\|. \tag{2.73}
\]
Using Eqs. (2.66) and (2.67):
\[
\omega^2 \mathcal{D} A^2 \leq 2t \| \mathcal{E}^\mu(t_1) \| \left\| \frac{d \mathcal{D} A^\mu}{dt} \right\|_{t=t_2}. \tag{2.74}
\]
Now, by the second inequality in (2.71),
\[ \omega_\mu^2 D A_\mu^2 \leq 2t \| \mathcal{E}^\mu(t_1) \| (2t \| \mathcal{E}^\mu(t_1) \|) \leq 4t^2 \| \mathcal{E}^\mu(t_1) \|^2, \] (2.75)
and we can also bound \( D A_\mu \) by
\[ |D A_\mu(t)| \leq \frac{2}{\omega_\mu} t \| \mathcal{E}^\mu(t_1) \|. \] (2.76)
As \( \| \mathcal{E}^\mu(t_1) \| \) is a quantity we can measure, Eqs. (2.72) and (2.76) give two possible bounds on \( D A_\mu \). Namely, these bounds measure how fast the error in our analytical solutions can grow. We see Eq. (2.72) is a stronger bound when \( t \leq 2/\omega_\mu \), and Eq. (2.76) is better for longer time lengths. In summary,
\[ |D A_\mu(t)| \leq \begin{cases} t^2 \| \mathcal{E}^\mu(t_1) \| & \text{if } t \leq 2/\omega_\mu, \\ \frac{2t}{\omega_\mu} \| \mathcal{E}^\mu(t_1) \| & \text{otherwise.} \end{cases} \] (2.77)

With this bound we have shown that we can explicitly quantify the error between a constrained MD system that approximates internal atom deviations (such as our large-scale dynamics approximation) and a full system with an exact representation, such as the classical MD equations. Numerical results in Section 3.2.2, will show that this particular bound does not provide a good approximation for realistic simulation runtimes, and as such, has ample room for improvement. However, such bounds may still have some utility. As previously noted, the large-scale approximation is more efficient as high-frequency modes no longer have to be evaluated numerically, although apart from being momentum-conserving, errors in the mode amplitudes can affect other statistical averages. Thus, there is a trade-off between efficiency and accuracy. Overall, an explicit bound can therefore be used to decide whether or not determining mode amplitudes analytically yields results within an acceptable error threshold, or a full numerical evolution is needed.

Acknowledgement: Chapter 2 in part is currently being prepared for submission for publication. This work is coauthored with Holst, Michael; Lindblom, Lee.
Chapter 3

Numerical results for the large-scale dynamics approximation

In this chapter, we present numerical results concerning our large-scale approximation. These results focus on error quantification between our approximation and classical MD. Noting that the large-scale dynamics approximation is really a class of schemes, we include comparisons between two particular choices of mode amplitude approximation: \( A_\mu = 0 \) and \( A_\mu = A_\mu^0 \sin(\omega_\mu t + \phi_\mu) e_\mu^A \) where \( A_\mu^0 \neq 0 \). Runtime efficiency and a baseline comparison to classical MD simulation are also discussed.

Our simulations use one of four Buckminsterfullerene molecules, which are hollow spherical molecules composed solely of carbon atoms. As the following results are mainly for the purposes of highlighting error quantification through our large-scale approximation, using these single fullerene molecules is sufficient. The following section gives the topologies of the four molecules and the potential used.

3.1 Setup

For our simulations, we use single \( C_{20}, C_{26}, C_{60}, \) and \( C_{70} \) molecules, see Fig. 3.1. The simple interaction potential energy function used for these examples
Figure 3.1: The $C_{20}$, $C_{26}$, $C_{60}$, and $C_{70}$ (left to right) molecules used in our molecular dynamics example simulations.

has the form:

$$U = \frac{1}{2} \kappa_b \sum_{\text{bonds}(CD)} (r_{CD} - L_b)^2 + \frac{1}{2} \kappa_\theta \sum_{\text{angles}(CDE)} (\theta_{CDE} - \theta_b)^2,$$

with the following carbon-carbon bond parameters: $\kappa_b = 305$ kcal/Å²/mol, $\kappa_\theta = 305$ kcal/rad²/mol, $L_b = 1.375$ Å, and $\theta_b = 120$ degrees, where $\kappa_b$ and $\kappa_\theta$ are the bond length and angle force constants, and $L_b$ and $\theta_b$ are the lengths and angles of equilibrium molecular bonds in this simple model. The values for $\kappa_b$, $L_b$, and $\theta_b$ are consistent with the CHARMM27 force field parameters for these carbon-carbon parameters [39]. Without the torsion angle bond forces, the Fullerene molecules become unstable using the CHARMM27 value for the bond angle force constant $\kappa_\theta = 40$ kcal/rad²/mol. The molecules were stable once this constant was increased to $\kappa_\theta = 305$ kcal/rad²/mol. Below, $r_{CD}$ and $\theta_{CDE}$ represent the distance between atoms $C$ and $D$ and the angle formed by the bonds between atom $D$ with atoms $C$ and $E$, respectively, where

$$r_{CD}^2 = (\vec{x}_C - \vec{x}_D) \cdot (\vec{x}_C - \vec{x}_D),$$
$$\cos \theta_{CDE} = \frac{(\vec{x}_C - \vec{x}_D) \cdot (\vec{x}_E - \vec{x}_D)}{r_{CD} r_{DE}}.$$

The sums in Eq. (3.1) are taken over bonds$(CD)$ and over angles$(CDE)$. These represent the collection of bonds between the pairs of atoms $C$ and $D$, and the collection of angles formed by the bond between atoms $C$ and $D$ and the bond between atoms $D$ and $E$, respectively. The force acting on the atom located at $\vec{x}_A$ is therefore given by:

$$\vec{F}(\vec{x}_A) = -\frac{\partial U}{\partial \vec{x}_A} = -\kappa_b \sum_{\text{bonds}(CD)} (r_{CD} - L_b) \Delta_{CD} \delta^A_{CD}$$
\[-\kappa_\theta \sum_{\text{angles}(CDE)} (\theta_{CDE} - \theta_b) \left( \vec{W}_{CDE} \delta^A_{DE} + \vec{W}_{EDC} \delta^A_{DC} \right), \quad (3.4)\]

where \(\vec{\Delta}_{CD}, \delta^A_{CD}\) and \(\vec{W}_{CDE}\) are given by

\[
\vec{\Delta}_{CD} = -\vec{\Delta}_{DC} = \vec{x}_C - \vec{x}_D, \quad (3.5)
\]

\[
\delta^A_{CD} = -\delta^A_{DC} = \delta_{AC} - \delta_{AD}, \quad (3.6)
\]

\[
\vec{W}_{CDE} = -\vec{\Delta}_{DC} - \cos \theta_{CDE} \vec{\Delta}_{DE} \rho_{DE} \sin \theta_{CDE}, \quad (3.7)
\]

The Hessian matrix needed to find the normal mode frequencies and eigenvectors for this simple potential energy function is given by:

\[
\frac{\partial^2 U}{\partial \vec{x}_A \partial \vec{x}_B} = \kappa_b \sum_{\text{bonds}(CD)} \left( \frac{L_b}{r_{CD}} \vec{\Delta}_{CD} \otimes \vec{\Delta}_{CD} + \frac{r_{CD} - L_b}{r_{CD}} \mathbf{I} \right) \delta^A_{CD} \delta^B_{CD}
\]

\[
+ \kappa_\theta \sum_{\text{angles}(CDE)} \left( \vec{W}_{EDC} \delta^A_{DC} + \vec{W}_{CDE} \delta^A_{DE} \right) \otimes \left( \vec{W}_{EDC} \delta^B_{DC} + \vec{W}_{CDE} \delta^B_{DE} \right)
\]

\[
- \kappa_\theta \sum_{\text{angles}(CDE)} \frac{\theta_{CDE} - \theta_b}{r_{CD}} \left[ \vec{W}_{EDC} \otimes \vec{\Delta}_{DC} + \vec{\Delta}_{DC} \otimes \vec{W}_{EDC} \right] + \frac{\cot \theta_{CDE}}{r_{DC}} \left( \vec{\Delta}_{DC} \otimes \vec{\Delta}_{DC} - \mathbf{I} \right)
\]

\[
+ \frac{\cot \theta_{CDE}}{r_{DC}} \vec{W}_{EDC} \otimes \vec{W}_{EDC} \delta^A_{CD} \delta^B_{CD}
\]

\[
- \kappa_\theta \sum_{\text{angles}(CDE)} \frac{\theta_{CDE} - \theta_b}{r_{CD}} \left[ \vec{W}_{CDE} \otimes \vec{\Delta}_{DE} + \vec{\Delta}_{DE} \otimes \vec{W}_{CDE} \right] + \frac{\cot \theta_{CDE}}{r_{DE}} \left( \vec{\Delta}_{DE} \otimes \vec{\Delta}_{DE} - \mathbf{I} \right)
\]

\[
+ \frac{\cot \theta_{CDE}}{r_{DE}} \vec{W}_{CDE} \otimes \vec{W}_{CDE} \delta^A_{DE} \delta^B_{DE}
\]

\[
- \kappa_\theta \sum_{\text{angles}(CDE)} \frac{\theta_{CDE} - \theta_b}{r_{DE}} \left[ \vec{\Delta}_{DE} \otimes \vec{W}_{EDC} + \frac{1}{r_{CD} \sin \theta_{CDE}} \left( \mathbf{I} - \vec{\Delta}_{DC} \otimes \vec{\Delta}_{DC} \right) \right]
\]

\[
+ \frac{\cot \theta_{CDE}}{r_{DE}} \vec{W}_{CDE} \otimes \vec{W}_{CDE} \delta^A_{DE} \delta^B_{DE}. \quad (3.8)
\]
3.2 Results

In this section we present the numerical results for our large-scale dynamics approximation. These results highlight our approximation and its effectiveness in quantifying various sources of error. After discussing error in momenta and energy conservation, we then compare error in the macroscopic variables in Chapter 2 between the classical MD method, which evolves the Cartesian components of each atom using Newton’s law, Eq. (2.1), and our scheme. Comparisons between runs where $A_\mu = 0$ and $A_\mu = A_\mu^0 \sin(\omega_\mu t + \phi_\mu) \vec{e}_\mu^0$ where $A_\mu^0 \neq 0$, are also discussed. We include runtime efficiency results as well as a baseline comparison using an exact scheme in which the normal mode amplitudes are not approximated.

All simulations are run using an 8th-order ODE time integrator by Dormand and Prince, see Hairer et al. [29] for detailed description of the method. The simulations are run at temperature $T = 300$ K. The initial mode amplitudes have $kT$ energy in each mode (unless the mode amplitudes are set to 0), where $k = 1.9872 \cdot 10^{-3}$ kcal/(mol K) is the Boltzmann constant. The initial center of mass velocity is given $kT/2$ overall translational kinetic energy, and the initial angular velocity is given $kT/2$ overall rotational kinetic energy.

Recall that the 8th order Dormand and Prince integrator allows for control of both absolute error, given by the parameter $\text{atol}$, and relative error, given by the parameter $\text{rtol}$. Given a function $\vec{y}$, $\text{rtol}$ keeps the local error of $y(i)$ below $\text{rtol} \ast |y(i)| + \text{atol}$. In our simulations, $\text{rtol}$ and $\text{atol}$ are the same value, so we will refer to it simply as $\text{tol}$.

3.2.1 Error in thermodynamic averages using our large-scale dynamics approximation

In this section we measure $L_2$ fractional error during an evolution using large-scale dynamics with respect to three averages: energy, linear momentum, and angular momentum. The total energy of the system is the sum of the potential
and kinetic energy, i.e.

\[ E = U + \frac{1}{2} \sum_A m_A \frac{d\vec{x}_A}{dt} \cdot \frac{d\vec{x}_A}{dt} \]  

(3.9)

Specifically, the energy conservation error is computed as

\[ \left\| \frac{E(t) - E(t_0)}{E(t_0)} \right\|_2 \]  

(3.10)

where \( E(t) \) is the total energy function and \( t_0 = 0 \). We compute these values for both the classical MD simulation the large-scale dynamics approximation. Similarly, the conservation errors for the momentum, \( \vec{P}(t) \) and \( \vec{J}(t) \), are

\[ \left\| \frac{\vec{P}(t) - \vec{P}(t_0)}{\vec{P}(t_0)} \right\|_2 \text{ and } \left\| \frac{\vec{J}(t) - \vec{J}(t_0)}{\vec{J}(t_0)} \right\|_2 \]  

(3.11)

respectively.

We implement the large-scale dynamics scheme with \( A_{\mu} = 0 \) (Figs. 3.2 and 3.3) and \( A_{\mu} = A^0_{\mu} \sin(\omega_{\mu} t + \phi_{\mu}) \vec{e}^A_{\mu} \) where \( A^0_{\mu} \neq 0 \) (Figs. 3.2 and 3.4). As shown in Section 2.3, our approximation conserves momentum, and indeed we see convergence with respect to both linear and angular momentum in Figs. 3.3 and 3.4. Energy in this approximate evolution is not set to be conserved, but energy, regardless of the number of atoms used, is conserved well when setting mode amplitudes to 0, and remains bounded when evolving amplitudes sinusoidally (see Fig. 3.2). Furthermore, the fractional change in energy remains small. For the purposes of clarity, we take the average error for each 10 picosecond interval, and plot this average at the endpoint of that interval.

### 3.2.2 Error in macroscopic quantities and mode amplitudes using our large-scale dynamics approximation

Eq. (2.15), which separates atom position into rotations, translations, and vibrations, not only allows us to examine error in traditional averages like momentum and energy, but further calculate error in specific variables that affect translation and rotation. The following result examines the \( L_2 \) fractional error in the macroscopic...
Figure 3.2: $L_2$ energy error using large-scale dynamics in $C_{20}$, $C_{26}$, $C_{60}$, and $C_{70}$ molecules. Left: $A_\mu = 0$. Right: sinusoidal evolution of $A_\mu$.

quantities: center of mass $\vec{x}_{CM}$, velocity of the center of mass $\vec{v}_{CM}$, angular velocity $\vec{\Omega}$, and spatial orientation $q$, where $q$ is a quaternion of unit length. These error measures allow us to evaluate the differences between a classical MD simulation and the large-scale approximation. In the following two graphs, we present the fractional error arising from 400 ps runs using an error tolerance of 1E-7. The errors for $\vec{v}_{CM}$ and $\vec{\Omega}$ are normalized through division by their initial values, i.e.

$$
\left\| \frac{\vec{v}_{CMc}(t) - \vec{v}_{CM\ell}(t)}{\vec{v}_{CM}(t_0)} \right\|_2 \quad \text{and} \quad \left\| \frac{\vec{\Omega}_c(t) - \vec{\Omega}_\ell(t)}{\vec{\Omega}(t_0)} \right\|_2,
$$

where variables with the subscript ‘$c$’ are values taken from a classical MD run, and variables with the subscript ‘$\ell$’ are values obtained from runs using our large-scale approximation. Since the position of the center of mass is initialized at the origin, it is normalized through the bond length parameter $L_b$:

$$
\left\| \frac{\vec{x}_{CMc}(t) - \vec{x}_{CM\ell}(t)}{L_b} \right\|_2.
$$

Note that since $q$ is a unit quaternion, no normalization is needed.

Again, we implement the large-scale dynamics scheme with $A_\mu = 0$ (Fig. 3.5) and $A_\mu = A^0_\mu \sin(\omega_\mu t + \phi_\mu) \hat{e}^{\mu}_A$ where $A^0_\mu \neq 0$ (Fig. 3.6). We see the error in the large-scale quantities remains bounded throughout evolution. In particular, in
Figure 3.3: $L_2$ error in linear and angular momentum setting $A_\mu = 0$. Top row: $C_{26}$. Bottom row: $C_{70}$.

Fig. 3.5, we see all macroscopic variables are approximated quite well with this scheme. As seen in Fig. 3.6, when $A_\mu$ is evolved sinusoidally, we see the position variables are approximated quite well, whereas the spatial variables are not as accurately approximated with this scheme.

The following figure compares our analytical error bound, Eq. (2.77), with the numerically calculated $L_2$ error between mode amplitudes calculated using an exact representation, i.e. the solution of Eq. (2.61), and approximate mode amplitudes, Eq. (2.43). We see Eq. (2.77) gives a relatively strict bound on the numerical error for the first 10 picoseconds, but for longer times does not approximate it as well. This shows the error bound has room for improvement. However, due to the chaos
Figure 3.4: $L_2$ error in linear and angular momentum setting $A_{\mu} = A_{\mu 0}^0 \sin(\omega_{\mu} t + \phi_{\mu}) \vec{c}_A$ where $A_{\mu 0}^0 \neq 0$. Top row: $C_{26}$. Bottom row: $C_{70}$.

occurring in MD simulation, this may be difficult to find.

3.2.3 Comparison within the class of large-scale approximations

We now directly compare the previous results for the two choices of mode amplitudes: $A_{\mu} = 0$ and $A_{\mu}$ being evolved sinusoidally. Similar to the previous section, we look at fractional error between a classical MD simulation versus each scheme. We run for 400ps using an error tolerance of 1E-7.

In Fig. 3.7, we compare energy and momenta conservation. Linear momen-
Figure 3.5: $L_2$ error in position and velocity variables and spatial orientation variables setting $A_\mu = 0$. Top row: $C_{26}$. Bottom row: $C_{70}$.

Energy and angular momentum, however, are better conserved when the mode amplitudes are simply set to zero. We note that angular momentum, which is set to be conserved in our large-scale scheme, can achieve similar levels of conservation with both choices of mode amplitude approximation if the error tolerance, which was set to 1E-7, is appropriately modified. As for the disparity in energy conservation, in the zero mode amplitude case the initial energy is on the order of $kT$, whereas in the sinusoidal mode amplitude case the initial energy is on the order of $kT + (3N - 6)kT$, where $N$ is the number of atoms. Thus the initial energy is much lower when setting the mode amplitudes to 0.
Fig. 3.6: $L_2$ error in position and velocity variables and spatial orientation variables setting $\mathbf{A}_\mu = A_\mu^0 \sin(\omega_\mu t + \phi_\mu) \mathbf{e}_A^\mu$ where $A_\mu^0 \neq 0$. Top row: $C_{26}$. Bottom row: $C_{70}$.

Fig. 3.8 presents the position and velocity error, as well as spatial orientation error for the two approximation schemes. Both schemes clearly evolve $\vec{x}_{CM}$ and $\vec{v}_{CM}$ accurately compared with a classical MD scheme. With spatial orientation, however, we see the scheme that sets the mode amplitudes equal to zero more accurately describes spatial orientation than the other scheme.

Overall, these tests suggest that if using an approximate MD method for a simple system, a simpler scheme, such as the zero mode amplitudes approximation above, may be better than using more complex schemes for evolving the vibrations. With a more complex system, either with more molecules or molecules with more
Figure 3.7: Energy and momenta comparison between $A_\mu$ being evolved sinusoidally (NM amps) versus zero mode amplitudes ($A_\mu = 0$). Top row: $C_{26}$. Bottom row: $C_{70}$.

complex structures, it is possible that sinusoidal evolution of the mode amplitudes will yield smaller error of the macroscopic variables.

3.2.4 Runtime efficiency

MD simulations can be limited in practical use as most require long duration times with respect to the timestep being used. Timesteps for classical MD simulations are typically taken on the order of the highest vibrational frequency in
the system, which is around $10^{-15}$ seconds. Constraint algorithms such as SHAKE [59] or RATTLE [5], which fix the fastest vibrational frequencies, or multiscale methods that allow for longer times between force updates [68], have been created to allow for larger timesteps. Using an analytical approximation for the normal modes provides some computational cost savings. The high frequency modes are the ones that force the numerical evolution to use very short time steps—around the order of the shortest oscillation. Evolving those high frequency modes analytically therefore allows longer time steps to be taken, as seen in Janežič et al.[31] and subsequent papers by Praprotnik and Janežič [50, 51]. Such methods make the numerical evolution more efficient.

Figure 3.8: Variable error comparison between $A_\mu$ being evolved sinusoidally (NM amps) versus zero mode amplitudes ($A_\mu = 0$). Top row: $C_{26}$. Bottom row: $C_{70}$. 
Figs. 3.9–3.10 present efficiency gains when using large-scale dynamics over a classical MD evolution. These figures compare runtimes for a classical MD simulation versus the class of large-scale approximations simulation. Fig. 3.9 shows runtimes for a $C_{20}$ molecule at varying levels of tolerance (1E-7–1E-12). All runs are for 400 ps. Fig. 3.10 provides runtimes for all four molecules ($C_{20}, C_{26}, C_{60}, C_{70}$) for a 400 ps length simulation at a single level of tolerance (1E-7).

![C20 Runtimes](image1)

**Figure 3.9**: Runtimes for a $C_{20}$ molecule with a classical MD solver versus the large-scale dynamics approximation with varying degrees of tolerance.

![Runtimes at Error Tolerance 10^{-7}](image2)

**Figure 3.10**: Runtimes for a $C_{20}, C_{26}, C_{60},$ and $C_{70}$ molecule at tolerance level 1E-7.
3.2.5 Baseline comparison to classical MD evolution

We compare a classical MD simulation with an exact normal mode basis representation (exact NMBR) in which no approximations for the mode amplitudes have been made. The exact NMBR evolves all variables in Eq. (2.15), including the mode amplitudes, according to an evolution using exact representation. As mentioned, the detailed derivation of this exact version can be found in Appendix C. As no approximations or simplifications were taken, this exact normal mode representation should exhibit the exact dynamics as a classical MD evolution, allowing us to ensure veracity using normal mode analysis to represent the internal motions of the molecule.

Since $n$-body systems are chaotic [49], that is, identical systems with slight perturbations in initial conditions will have divergent trajectories, we cannot expect traditional convergence with our evolutions. Instead, we will look for convergence with respect to increasing tolerance levels for our ODE time integrator.

We present results comparing evolution using lower accuracy ($tol = 1E-7$ to $1E-11$) versus the highest accuracy ($tol = 1E-12$) for both the classical MD and exact NMBR algorithms. Each curve represents the absolute error in atom position evolution between the accuracy used for the ODE time integrator for a particular run versus the highest accuracy used out of all the evolution runs. Specifically, we are evaluating the $L_2$ norm $\|\vec{x}(tol) - \vec{x}(tol_{\text{max}})\|_2$, where $tol = 1E-7, \ldots, 1E-11$, and $tol_{\text{max}} = 1E-12$, for a classical MD simulation and then an exact NMBR simulation. We see in Fig. 3.11 that convergence trends for the exact normal mode basis representation are similar to the behavior seen using a classical, full MD algorithm.

Next, we present the $L_2$ fractional error with respect to three averages: energy, linear momentum, and angular momentum. Specifically, the energy error is computed as $\|E(t) - E(t_0)/E(t_0)\|_2$, where $E(t)$ is the total energy function and $t_0 = 0$. We compute these values for both the classical MD simulation the exact NMBR simulation. Similarly, the errors for the momentum, $\vec{P}(t)$ and $\vec{J}(t)$, are $\|\vec{P}(t) - \vec{P}(t_0)/\vec{P}(t_0)\|_2$ and $\|\vec{J}(t) - \vec{J}(t_0)/\vec{J}(t_0)\|_2$, respectively. The exact NMBR should conserve these thermodynamic quantities, which we see in the following
Figure 3.11: Differences in atom position in Angstroms (Å). Convergences for a classical MD simulation versus an exact normal mode basis representation. Solid curves are results from the exact normal mode variable algorithm, dashed curves are results from a classical MD algorithm. Left: $C_{26}$. Right: $C_{70}$.

Figure 3.12: $L_2$ energy error. Solid curves are results from the exact normal mode variable algorithm, dashed curves are results from a classical MD algorithm. Left: $C_{26}$. Right: $C_{70}$.

Figs. 3.12 and 3.13.

Acknowledgements: Chapter 3 in part is currently being prepared for submission for publication. This work is coauthored with Holst, Michael; Lindblom, Lee.
Figure 3.13: $L_2$ error in linear and angular momentum. For the angular momentum graphs, solid curves are results from the exact normal mode variable algorithm, dashed curves are results from a classical MD algorithm. Top row: $C_{26}$. Bottom row: $C_{70}$. 
Chapter 4

Conclusion to error quantification for large-scale dynamics

We have constructed a large-scale MD approximation scheme using the inherent two-scale nature of molecular motion. With this approximation, the position and orientation of a molecule are evolved using an exact representation whereas the normal modes that describe the atoms’ deviations are evolved analytically using normal mode analysis. In particular, the mode amplitudes can be approximated in various ways; thus the large-scale dynamics scheme generates a class of approximations. These equations were derived by carefully selecting variables so as to better separate the quantities involved in molecular movement. As such, we are better able to quantify their effect, which can be seen through numerical simulation. In particular, we tested our large-scale approximation on simple systems involving a single fullerene molecule.

When $A_\mu$ is evolved sinusoidally, numerical results for our test cases show that:

- linear and angular momenta are well-conserved.
- energy is fairly conserved.
- position and velocity variables are well-approximated.
- rotation and angular velocity variables are poorly-approximated.
When $\mathcal{A}_\mu = 0$, numerical results for our test cases show that:

- linear and angular momenta are well-conserved.
- energy is well-conserved.
- position and velocity variables are well-approximated.
- rotation and angular velocity variables are well-approximated.

Error quantification is particularly useful in the analysis of MD approximation schemes. The error estimates given with our large-scale approximation can be used as a baseline comparison for these approximations. For example, with the fullerene molecule tests, we found that the simpler approximation scheme, which sets the mode amplitudes to zero, performed better than when the amplitudes were evolved sinusoidally. Overall, our particular large-scale approximation provides a basis to compare the error between full, classical MD simulations and large-scale approximations in general.
Chapter 5

Introduction to level sets for detonation shock dynamics (DSD)

In the second half of this thesis, we study level set methods for detonation shock dynamics (DSD). Level set methods were introduced by Osher and Sethian in [46] to implicitly track dynamically evolving interfaces. The basis for these methods is the ability to embed an \( n \)-dimensional interface into an \( n + 1 \)-dimensional surface – called the level set function – and track the evolution of the interface by evolving the function instead. This approach arose as an alternative to tracking the interface through surface parameterization, where moving nodes, which can cross over themselves or separate into distinct regions, can lead to loss of accuracy and be difficult to implement algorithmically. In level set methods, the interface is represented as the zero-contour of the level set function, or the zero level set. Given the speed of the interface, it is then simple to describe movement of the front with a partial differential equation (PDE)-based advection equation [10, 44, 63].

The simplicity of the foundation behind level set methods lends itself to a number of applications [44, 45, 66], including image processing [47], fluid dynamics [6], and crystal growth [16]. Our interest is in the level sets’ ability to model high-explosive detonation fronts through detonation shock dynamics [9, 11, 13]. DSD is a theory describing the evolution of multidimensional detonation shocks using the velocity and curvature of the front. The modeling of high-explosive (HE) energy release (or burn) typically falls into two categories: (1) geometric methods in which
the times of energy release are pre-calculated based on the HE detonation velocity and calculated distances from a given detonation point, and (2) more sophisticated reactive chemistry models which directly compute the HE energy release based on solving reaction rate equations for the reactants and products [18, 34]. Here, we are concerned exclusively with geometric methods, and in particular, with DSD methods.

When using level sets for the purposes of DSD, given a level set function \( \phi(\vec{x}, t) \), the burn front is represented by the contour \( \phi = 0 \). The area where \( \phi < 0 \) corresponds to the products of the reaction, while the area where \( \phi > 0 \) corresponds to the unburned explosive. The goal of the DSD solver is to evolve (or propagate) the burn front \( \phi = 0 \) from some initial configuration to a final time where it has completely propagated through the region of interest. The resulting solution can be used to construct a so-called burn table, which contains the times the burn front arrives at particular points on the mesh.

In our approach, we spatially discretize the described PDE using high-order finite elements. Since we wish to address problems which may have complex geometries on unstructured meshes, using finite elements is more flexible over traditional finite difference methods. In particular, to address developing discontinuities in the dynamically evolving interface, we use the discontinuous Galerkin (DG) approach that allows us to handle discontinuities across elements. This approach allows the propagating front to easily handle complicated geometries and curved, highly-unstructured 2D and 3D meshes. After spatial discretization, we use standard explicit time integrators (typically matching the temporal order of the time integrator to the spatial discretization order) to evolve the resulting ODE.

The second half of this thesis contains four chapters and is organized as follows. In the remaining sections of this chapter we present the level set method as well as level set reinitialization, which is performed for stability purposes. We also highlight its application to DSD and HE burn. In Chapter 6, we present the details of our finite element spatial discretization and a description of the time integrator used for both level set advection and redistancing. Numerical results are presented in Chapter 7, which include typical level-set benchmarks as well as more
complex front propagation problems involving corner turning and shadow surfaces. We conclude in Chapter 8. We summarize the main results of this second project below.

- We design and implement a 2D/3D serial and parallel unstructured mesh level set method with arbitrary-order finite elements using the discontinuous Galerkin approach.

- Results show this approach leads to stable front propagation in 2D and 3D on most meshes.

  - We design and implement a modified reinitialization that makes the redistancing step in the level set method less ad hoc and improves stability.

- We apply this method to several problems, including modeling high-explosive burn on complex geometries through detonation shock dynamics.

5.1 Level set method

Level set methods are centered around evolving a scalar function \( \phi(\vec{x}, t) \) in time through advection:

\[
\frac{\partial \phi}{\partial t} + \vec{u} \cdot \nabla \phi = 0,
\]

where \( \vec{u} = u \frac{\nabla \phi}{|\nabla \phi|} \) is the velocity of the front, which we denote as the contour \( \phi = 0 \). Note that when well-defined, \( \nabla \phi / |\nabla \phi| \) is a unit vector field orthogonal to the level sets of \( \phi \). In this work, we will specify \( u \) as constant, but in general DSD applications, \( u = D_{CJ} - \alpha(\kappa) \), where \( D_{CJ} \) is the Chapman-Jouguet detonation velocity and \( \alpha(\kappa) \) is a function of curvature. Here, the curvature is the divergence of the front normal,

\[
\kappa = \nabla \cdot \vec{n}_f = \nabla \cdot \left( \frac{\nabla \phi}{|\nabla \phi|} \right).
\]

As opposed to surface parametrization methods mentioned above, it is easy to deal with topological changes when using level sets. A simple example of the
Figure 5.1: Example of level set evolution: Propagation of a circle, i.e. the zero contour of \( \phi \), on an unstructured mesh from \( t_0 \) (red) to \( t_{\text{final}} \) (black).

propagation of a circular interface according to the initialized level set function

\[
\phi = \sqrt{(x-x_0)^2 + (y-y_0)^2 - r},
\]

is shown in Fig. 5.1.

5.1.1 Reinitialization

In the next two subsections, we discuss approximation issues with the level set method and ways to mitigate them. During evolution, \( \phi \) may drift from its initial value, creating steep or shallow gradients leading to inaccurate approximation of the spatial derivatives. For better accuracy, it is standard to reinitialize \( \phi \), which can be done through the following reinitialization equation [70]:

\[
\frac{\partial \phi}{\partial \tau} + S(\phi)(\|\nabla \phi\| - 1) = 0,
\]

where \( \tau \) is a psuedotime variable, and \( S(\phi) \) corresponds to the sign function

\[
S(\phi) := \begin{cases} 
-1 & \text{if } \phi < 0, \\
1 & \text{if } \phi > 0, \\
0 & \text{otherwise.}
\end{cases}
\]
For stability purposes, a smoothed sign function is used, such as that in the reinitialization equation below (adapted from [70]):

$$\frac{\partial \phi}{\partial \tau} + \left( S(\phi) \frac{\nabla \phi}{|\nabla \phi|} \right) \cdot \nabla \phi = S(\phi), \quad \text{where} \quad S(\phi) = \frac{\phi}{\sqrt{\phi^2 + |\nabla \phi|^2 h(x)}}, \quad (5.6)$$

where $h(x)$ is a mesh size representing the resolution of the spatial discretization. Specifically, $h(x)$ is calculated by taking the smallest singular value of the Jacobian of the transformation from the perfect reference element at the center of said element. This is used instead of the more standard $\Delta x$ to accommodate for more general mesh geometries. We remark that Eq. (5.6) is only one of several variations of a reinitialization equation, see [44] for examples of others.

The above equation redistances the level set function to a signed distance function. See Fig. 6.1 for a 2D example and Fig. 5.2 for a 3D example. A signed distance function preserves the property $|\nabla \phi| = 1$. Although this method is widely used throughout the literature, its exact implementation remains somewhat ad hoc.
When and for how many iterations to run Eq. (5.6) varies between algorithms, and can cause stability to become problem-dependent. We use a modified redistancing equation:

\[
\frac{\partial \phi}{\partial \tau} + \left( S(\phi) \frac{\nabla \phi}{|\nabla \phi|} \right) \cdot \nabla \phi = S(\phi) + \mu(h(x)) \nabla \cdot \left[ \left( 1 - \frac{1}{|\nabla \phi|} \right) \nabla \phi \right],
\] (5.7)

where \( S(\phi) \) is the same as in Eq. (5.6) and the additional right-hand side term, adapted from [37], provides artificial viscosity for stability purposes. This term acts as a penalty: when \(|\nabla \phi|\) strays far from 1, the term \((1 - 1/|\nabla \phi|)\) is a diffusion rate aiding the level set function with redistancing. The diffusion coefficient \( \mu(h(x)) \) is mesh-dependent, and as shown in Section 7.6, it results in stable propagation on complex meshes. In theory, the level set is reinitialized until Eq. (5.6) reaches a steady-state. However, during numerical evolution the interface will move during reinitialization, so care must be taken not to redistance too much [64]. With this in mind, we use a variable constraint redistancing method, i.e. the amount of redistancing steps taken after each advection step will vary. We address this issue in more detail in the next chapter.

5.1.2 Other approaches

Reinitialization-free methods

Another option to addressing the cons in the classic redistancing algorithm is to use a reinitialization-free approach [15, 40, 77]. In a separate effort to address implementation issues seen in classical redistancing, we formulated and studied an approach which is a modified version of the equation used in [37]:

\[
\frac{\partial \phi}{\partial t} + \textbf{u} \cdot \nabla \phi = \mu(h(x)) \nabla \cdot \left[ \left( 1 - \frac{1}{|\nabla \phi|} \right) \nabla \phi \right],
\] (5.8)

where \( \mu(h(x)) \) is the mesh-dependent diffusion coefficient.

The above equation is the only equation needed, unlike the tandem advection and reinitialization equations used in this thesis. The right-hand side acts as a penalty term. When the \( \phi \) is a signed distance function – that is, \(|\nabla \phi| = 1\) – the right-hand side is zero and the equation becomes the standard advection equation,
Eq. (5.1). When $\phi$ does not satisfy this constraint, the right-hand side acts as a redistancer and corrects $\phi$.

Since only one equation is needed to advect and correct $\phi$, this method is simpler to implement than the previous one. However, due to the diffusion term, the Courant-Friedrichs-Lewy condition is more restrictive, depending on $h(x)^2$ as opposed to simply $h(x)$. Eq. (5.8) also failed in some problems tested to correctly adjust to the signed distance function, namely with meshes with circular holes, as in Section 7.7. This may be due to the fact that if $|\nabla \phi|$ is less than 1, the term

$$
\mu(h(x)) \nabla \cdot \left[ \left( 1 - \frac{1}{|\nabla \phi|} \right) \nabla \phi \right],
$$

becomes hyperbolic, affecting stability. Although not yet tested, proposed fixes for this are to replace this potentially problematic term $1 - 1/|\nabla \phi|$ with the term $\max(1 - |\nabla \phi|^2, 1 - 1/|\nabla \phi|)$, or to take $1 - 1/|\nabla \phi|$ outside the derivative and square it.

**Fast marching methods**

Fast marching methods, also developed by Sethian [62], is another numerical method for propagating interfaces. These methods also use PDEs to track an evolving interface, however, it solves a boundary value problem (instead of an initial value problem) called the Eikonal equation, i.e.

$$
|\nabla \phi(\bar{x})| = \frac{1}{f(\bar{x})}, \quad \bar{x} \in \Omega, \quad (5.10)
$$

$$
\phi(\bar{x}) = 0, \quad \bar{x} \in \partial \Omega, \quad (5.11)
$$

where $f(\bar{x})$ is the speed. Note that fast marching methods require $f > 0$, while level set methods do not. While level set methods denote the front as $\phi(\bar{x}, t) = 0$, fast marching methods instead denote the front as $\phi(\bar{x}) = t$. Since this method is not used in the results in this thesis, we refer the interested reader to Sethian [65] for further information. Sethian gives not only an overview, but also a comparison between both fast marching methods and level sets.
5.2 Application to DSD and high-explosive burn

In this section, we discuss using level sets for detonation shock dynamics (DSD) for the purposes of modeling high-explosive (HE) burn. High explosives characterize materials that detonate, i.e. during the reaction, the front moves faster through the region of interest than the speed of sound. DSD is a theory describing the evolution of multidimensional detonation shocks using the velocity and curvature of the front.

The ability to model a dynamically evolving front makes level sets a prime candidate for modeling HE burn [9]. In particular, using level sets for DSD has several advantages compared to other methods. Reactive chemistry, in which one concurrently solves rate equations for the reactants and products of the burn using detailed chemistry [18, 34], is fairly expensive. Another option is geometric burn, in which the burn front is pre-calculated, and we assume simple equations of state for the reactants and products. One method is Huygen’s construction [11], which is cheaper than reactive chemistry but requires geometric assumptions. While DSD, which falls under geometric burn, is more expensive than using Huygen’s construction, it works for general geometries and is less expensive than reactive chemistry, striking an appropriate balance. See Fig. 5.3 for a schematic.

Using level sets to model HE burn relies on the idea that the radius of the shock curvature is large compared to the one-dimensional, steady, reaction zone length. This relatively thin reaction zone can then be viewed as a front separating burnt from unburned explosive [12]. Using the language of level sets, we denote the zero-contour of $\phi$ as our detonation front. The material in front of $\phi = 0$, i.e. $\phi > 0$, are the reactants, while $\phi < 0$ designates the resulting products. The velocity coefficient used is the normal shock velocity $u = D_{CJ} - \alpha(\kappa)$, where $D_{CJ}$ is the Chapman-Jouguet detonation velocity, or the normal speed of the detonation, and $\alpha(\kappa)$ is a function of curvature. As previously mentioned, we consider only the case of $\alpha = 0$ in this work.

DSD also takes into account additional boundary conditions, namely

$$\cos \theta = \vec{n}_b \cdot \vec{n}_s,$$  \hspace{1cm} (5.12)
where $\theta$ is an experimentally-calculated angle, $\vec{n}_b$ is the boundary normal, and $\vec{n}_s$ is the local shock normal. When using level sets, we define the shock normal as

$$
\vec{n}_s = \frac{\nabla \phi}{|\nabla \phi|}.
$$

(5.13)

In particular, the boundary conditions require two constant boundary angles: the sonic angle, $\theta_s$, and the critical angle, $\theta_c$. These two angles dictate the shock dynamics; if the flow is supersonic ($\theta < \theta_s$), no additional boundary conditions are applied, and if the flow is sonic ($\theta = \theta_s$) or subsonic ($\theta > \theta_s$), $\theta$ is bounded above by $\theta_c$. See [9] for a detailed discussion on boundary angle conditions. In this work, we do not implement any angle conditions (see Future Work in Chapter 8); however the approach we describe is readily extendible to more general settings. With regard to boundary conditions we do use, the boundaries are set to the value of the level set function $\phi$ at the particular boundary point.

We can integrate DSD results with hydrodynamics simulations through creation of a burn table. The implementation of a burn table is straightforward. The solution of our advection equation is $\phi(\vec{x}, t)$. With this, we can locate the burn front at any time by finding the zero contour of the level set, $\phi = 0$. Recording
the time the burn front arrives at particular points on the mesh gives us our burn table. We can do this simply by seeing if $\phi$ has changed sign at a node after a timestep has been taken. Note that in doing this, we assume the velocity coefficient $u$ is greater than zero, as this means the shock will cross a node only once. Thus, although level set methods do not generally require $u > 0$, it will be nonnegative for the purposes of this thesis.

Acknowledgement: Chapter 5, in part is currently being prepared for submission for publication. This work is coauthored with Dobrev, Veselin; Kolev, Tzanko; Rieben, Robert; Tomov, Vladimir. A portion of this work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.
Chapter 6

Level sets using high-order finite elements (FE)

This chapter focuses on the numerical solution to our level set algorithm. First, we discuss the spatial discretization of Eqs. (5.1) and (5.7) using an arbitrary-order discontinuous Galerkin approach. We then give the time integration method used, along with an algorithm, Alg. 1, detailing the pseudocode for this method. We then test this algorithm on several problems in Chapter 7.

6.1 Finite element discretization

While level sets have several advantages over parametrization approaches, such as ease of representing the given interface, numerically solving the PDE in Eq. (5.1) can be difficult. In particular, using basic finite difference methods to spatially discretize the level set equation often are not stable. If using finite difference methods, high-order methods generally have to be used [44], and still may be non-optimal depending on the problem (see Section 7.2 for an example). Hybrid methods combining level sets with particle markers along the interface have been developed to improve stability and resolution along the front [24].

In our algorithm, we spatially discretize the level set equations using a discontinuous Galerkin method (DG) in space [19, 30, 54]. Inheriting features of the finite volume and finite element frameworks, DG methods are standard in nu-
merically evolving level set equations as they can capture the natural discontinuities that develop during advection.

To derive the spatial discretization for a fixed $t$, we start from a splitting of the computational domain into a mesh with elements (zones) $\{ z \}$ on which we introduce a piecewise polynomial discontinuous finite element space with basis functions $\psi_i(x,t)$. This is a finite-dimensional subspace of $L_2$. Since we are interested in high-order methods, the basis functions can be mapped to high-order polynomials, and the mesh elements are allowed to be curved (i.e. described by a high-order polynomial mapping).

We approximate the level set function by its expansion in the finite element basis,

$$\phi(x,t) = \sum_{i=1}^{N} \phi_i(t) \psi_i(x,t), \quad (6.1)$$

where $\phi_i(t)$ is the $i$th element of the discrete vector of degrees of freedom in the finite element space. Starting from the weak form of the advection equation tested with a finite element basis function $\psi_i \in L_2$, we integrate by parts to obtain

$$\int_{\Omega} \frac{\partial \phi}{\partial t} \psi = \int_{\Omega} (\vec{u} \cdot \nabla \psi) \phi - \int_{\partial \Omega} \phi (\vec{u} \cdot \vec{n}) \psi. \quad (6.2)$$

Define the jump operator $[ [ \psi ] ] := \psi^+ - \psi^-$, $\psi^\pm = \lim_{\epsilon \to 0^\pm} \psi(\vec{x} \pm \epsilon \vec{n}(\vec{x}))$, and the upwind operator

$$\psi_u := \begin{cases} 
\psi^+ & \text{if } \vec{u} \cdot \vec{n} > 0, \\
\psi^- & \text{if } \vec{u} \cdot \vec{n} < 0, \\
\psi & \text{if } \vec{u} \cdot \vec{n} = 0.
\end{cases} \quad (6.3)$$

We can then express Eq. (6.2) as

$$M \frac{d\phi}{dt} = K \phi, \quad (6.4)$$

where $M$ is the mass matrix and $K$ is the advection matrix for the finite element space:

$$M_{ij} = \sum_z \int_z \psi_i \psi_j, \quad K_{ij} = \sum_z \int_z (\vec{u} \cdot \nabla \psi_j) \psi_i - \sum_f \int_f (\vec{u} \cdot \vec{n}) [ [ \psi_j ] ] \psi_i. \quad (6.5)$$
Here, the integrals are over the zones $z$ and faces $f$ of the mesh. We perform a similar discretization of our reinitialization equation, Eq. (5.7). For simplicity, we rewrite Eq. (5.7) as

$$\frac{\partial \phi}{\partial \tau} + \mathbf{u} \cdot \nabla \phi = S(\phi) + \nabla \cdot [c \nabla \phi],$$  

(6.6)

where

$$\mathbf{u} = \left( S(\phi) \frac{\nabla \phi}{|\nabla \phi|} \right), \quad c = \mu(h(x)) \left( 1 - \frac{1}{|\nabla \phi|} \right).$$  

(6.7)

Define the following terms:

$$\{(c \nabla \phi) \cdot \mathbf{n}\} \psi := (\mathbf{n}^+ \cdot (c \nabla \phi)^+) \psi^+ + (\mathbf{n}^- \cdot (c \nabla \phi)^-) \psi^-, \quad \text{(6.8)}$$

$$\{c/h(x)\} := \frac{(c/h(x))^+ + (c/h(x))^-}{2}, \quad \text{(6.9)}$$

where $\mathbf{n}^\pm$ denotes the unit normals pointing outward from their respective element. The finite element discretization is then

$$M \frac{d\phi}{d\tau} = (K + D)\phi + \mathbf{f}, \quad \text{(6.10)}$$

where $M$ and $K$ are the mass and advection matrices from Eq. (6.4), and $D$ and $\mathbf{f}$ are given by

$$D_{ij} = \sum_z \int_z -c \nabla \psi_i \cdot \nabla \psi_j - \sum_f \int_f \{(c \nabla \psi_i) \cdot \mathbf{n}\} \psi_j \right] + \sigma \left[ \psi_i \right] \{(c \nabla \psi_j) \cdot \mathbf{n}\} + \kappa \left\{ \frac{c}{h(x)} \right\} \left[ \psi_i \right] \left[ \psi_j \right], \quad \text{(6.11)}$$

$$\mathbf{f}_i = \sum_z \int_z S(\phi) \psi_i, \quad \text{(6.12)}$$

where $\sigma$ and $\kappa$ are scalars. We choose $\sigma = -1$ and $\kappa > 0$ which yields a symmetric interior penalty Galerkin method [21, 27, 73].

### 6.2 Time integration

Once we have obtained the semi-discrete form Eqs. (6.4) and (6.10), we integrate both sets of equations using an explicit Runge-Kutta operator usually of order corresponding to the order of the finite elements space. For example, the use
of second order elements implies a time discretization using two-stage Runge-Kutta for the advection steps:

\[ \phi^{n+\frac{1}{2}} = \phi^n + \frac{\Delta t}{2} M^{-1} K \phi^n, \quad (6.13) \]
\[ \phi^{n+1} = \phi^n + \Delta t M^{-1} K \phi^{n+\frac{1}{2}}, \quad (6.14) \]

as well as the reinitialization steps:

\[ \phi^{n+\frac{1}{2}} = \phi^n + \frac{\Delta \tau}{2} M^{-1} \left[(K + D) \phi^n + \vec{f}\right], \quad (6.15) \]
\[ \phi^{n+1} = \phi^n + \Delta \tau M^{-1} \left[(K + D) \phi^{n+\frac{1}{2}} + \vec{f}\right]. \quad (6.16) \]

Note that although \( \tau \) is, in theory, supposed to act as a psuedotime variable, we found that changes in \( \tau \) greatly affected the success or failure of the reinitialization steps. With this in mind, we set \( \tau \) and \( \Delta \tau \) to be equal to \( t \) and \( \Delta t \), respectively. We reinitialize after every five iterations of the advection equation. In an effort to address consistency with how long to reinitialize, we use variable redistancing. Using the fact that a signed distance function stipulates \( |\nabla \phi| = 1 \), we reinitialize until \( |\nabla \phi| \) is within an acceptable order of 1, or the maximum number of iterations has been reached. This constraint prevents the level set from over-reinitializing. The detailed algorithm is presented in Alg. 1.

**Figure 6.1:** An initially distorted level set function (left) is reinitialized to a proper signed distance function (right) using \( Q_2 \) finite elements.

To illustrate the redistancing portion of the algorithm which solves Eq. (5.7), we briefly consider the test problem described in [69]. Specifically, we start with a
Algorithm 1: Level set solver

\[ t \leftarrow 0 \]

\[ \text{iter} \leftarrow 0 \]

while \( t \leq t_{\text{final}} \) do

\[ \phi^{n+\frac{1}{2}} \leftarrow \phi^n + \frac{\Delta t}{2} M^{-1} K \phi^n \]

\[ \phi^{n+1} \leftarrow \phi^n + \Delta t M^{-1} K \phi^{n+\frac{1}{2}} \]

\[ \text{iter} \leftarrow \text{iter} + 1 \]

if \( \text{iter} \bmod 5 = 0 \) then

\[ \tau \leftarrow 0 \]

while \( \tau \leq \tau_{\text{max}} \) or \( 0.9 \leq |\nabla \phi| \leq 1.1 \) do

\[ \phi^{n+\frac{1}{2}} \leftarrow \phi^n + \frac{\Delta t}{2} M^{-1} \left[ (K + D) \phi^n + \tilde{f} \right] \]

\[ \phi^{n+1} \leftarrow \phi^n + \Delta t M^{-1} \left[ (K + D) \phi^{n+\frac{1}{2}} + \tilde{f} \right] \]

\[ \tau \leftarrow \tau + \Delta \tau \]

end while

end if

\[ t \leftarrow t + \Delta t \]

end while

return \( \phi \)

distorted level set function:

\[ \phi = \left( (x - 1)^2 + (y - 1)^2 + 0.1 \right) \left( \sqrt{x^2 + y^2} - 1 \right), \quad (6.17) \]

which has steep gradients that will affect spatial accuracy if not corrected. After iterating solely through Eq. (5.6), the level set function should properly reinitialize to the signed distance function

\[ \phi = \sqrt{x^2 + y^2} - 1. \quad (6.18) \]

Results from application of the redistance solver using quadratic \((Q_2)\) elements on a \([0, 1] \times [0, 1]\) structured square mesh are shown in Fig. 6.1.

Acknowledgement: Chapter 6, in part is currently being prepared for submission for publication. This work is coauthored with Dobrev, Veselin; Kolev, Tzanio;
Rieben, Robert; Tomov, Vladimir. A portion of this work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.
Chapter 7

Numerical results for the level set method using high-order FE

In this chapter we test our level set method and present several numerical results. The level set algorithm with modified reinitialization was implemented in C++ using the MFEM finite element software library [1]. The solver runs in both serial and parallel; the latter being useful for simulation in 3D or problems with a large number of unknowns. Although not explicitly used in the following tests, the level set solver is also capable of implementing a reinitialization-free approach, Eq. (5.8), as well as two simple implicit ODE solvers: backward Euler and implicit midpoint.

Our choice of numerical tests is similar to those presented in [43]. The first two results are based on using a prescribed velocity field in order to test the stability and accuracy of the basic advection equation, Eq. (5.1). The remaining tests consider the more general case where the velocity of the front is determined by its time-evolving surface normal, which requires use of the full level set evolution plus redistance solver. The order of elements used in the following problems varies from one to four to show the solver’s ability to implement finite elements of arbitrary order. As previously mentioned, an explicit Runge-Kutta method is used with order equal to that of the spatial discretization. Simulation visuals were generated using the visualization tool VisIt [17].
7.1 Advection only test: sheared circle

Originating in [56], a circle is deformed according to the velocity field

$$\vec{u} = -\cos \left( \frac{\pi t}{8} \right) \left[ -\sin(\pi x) \cos(\pi y), \sin(\pi y) \cos(\pi x) \right]. \quad (7.1)$$

The circle deforms in one direction until \( t = 4 \), and then rotates in the other direction until \( t = 8 \). At \( t = 8 \) the circle should have the same shape and location as the initial circle. We use cubic \( Q_3 \) finite elements on a \([0, 1] \times [0, 1]\) structured square mesh with 4,096 elements. The circle itself has radius 0.2 and is initially centered at \((0.5, 0.75)\), i.e. the initial level set function is

$$\phi = \sqrt{(x - 0.5)^2 + (y - 0.75)^2} - 0.2. \quad (7.2)$$

Results are in Fig. 7.1, showing the circle returning to its original location and shape, without any need for adaptive mesh refinement (AMR).

7.2 Advection only test: Zalesak’s disk

This problem, proposed in [76], rotates a notched circle with velocity

$$\vec{u} = [-\pi y, \pi x]. \quad (7.3)$$

We use cubic \( Q_3 \) finite elements on a \([-1, 1] \times [-1, 1]\) structured square mesh with 9,216 elements. The notch’s corners should not deform as the circle rotates and should remain sharp. Smearing of the corners may be seen with certain finite difference methods if AMR is not used (see [24] for an example). Results are in Fig. 7.2. The solution exhibits numerical oscillations at the circle interface, which is expected behavior for the use of high-order elements without any special (non-linear) flux correction [7].
Figure 7.1: Deformation of circle from $t = 0$ (top left) to $t = 8$ (bottom right). In the third image, at $t = 4$, the circle reverses direction.
7.3 Expanding circle test

In the remaining problems we use a velocity based on the surface normal $\nabla \phi/|\nabla \phi|$, with a constant magnitude $u$.

This simple problem consists of tracking the expansion of an initial circular level set with a starting radius of $r_0$, expanding outward with constant speed $u = 1$. The analytic solution to the problem for any given time $t$ is simply an expanded circle of radius $r_0 + ut$. In this example we consider two different mesh types:

- a simple Cartesian mesh of $[0, 1] \times [0, 1]$. 

Figure 7.2: One clockwise rotation of notched disk (top left to bottom right). The notch maintains its integrity and does not deform as the disk rotates.
• an more complex unstructured, quadratically curved \((Q_2)\) mesh.

We show mesh convergence results for the expanding circle initialized according to the level set

\[
\phi = \sqrt{(x - 0.5)^2 + (y - 0.5)^2} - 0.1, \tag{7.4}
\]

on the simple Cartesian mesh in Fig. 7.3 using both \(h\) and \(p\) refinement. The convergence study, which plots the \(L_2\) error between the exact and calculated numerical solution, i.e. \(\|\phi_{\text{exact}} - \phi_{\text{num}}\|_2\), is performed using both the classic redistancing method and the modified redistancing method with coefficient \(\mu = 0.00025/p\), where \(p\) is the order of elements used. Convergence rates are provided in Tables 7.1 and 7.2. Note that the classic redistancing method achieves for the most part, the expected high-order convergence rates; although the \(p = 4\) case is not reaching 4th order. The modified redistancing method exhibits some degree of high-order convergence until the error reaches a \textit{floor} which is due to the addition of numerical diffusion characterized by the coefficient \(\mu = 0.00025/p\).

**Table 7.1**: Convergence Rates with Classic Redistancing

<table>
<thead>
<tr>
<th></th>
<th>(h=1/8)</th>
<th>(1/16)</th>
<th>(1/32)</th>
<th>(1/64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order 1</td>
<td>1.16</td>
<td>1.04</td>
<td>0.79</td>
<td>1.63</td>
</tr>
<tr>
<td>2</td>
<td>2.39</td>
<td>2.33</td>
<td>1.92</td>
<td>2.19</td>
</tr>
<tr>
<td>3</td>
<td>3.18</td>
<td>2.89</td>
<td>2.57</td>
<td>2.38</td>
</tr>
<tr>
<td>4</td>
<td>3.13</td>
<td>3.59</td>
<td>2.86</td>
<td>3.31</td>
</tr>
</tbody>
</table>

In Fig. 7.4 we show results of the level set \(\phi = \sqrt{x^2 + y^2} - 0.1\) propagating outwards using \(Q_2\) (quadratic) finite elements on the unstructured mesh. The algorithm is able to preserve the symmetry of the initial circular level set even on a highly unstructured, curved mesh.
Table 7.2: Convergence Rates with Modified Redistancing ($\mu = 0.00025/p$)

<table>
<thead>
<tr>
<th>Order</th>
<th>h=1/8</th>
<th>1/16</th>
<th>1/32</th>
<th>1/64</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.16</td>
<td>1.04</td>
<td>0.70</td>
<td>1.60</td>
</tr>
<tr>
<td>2</td>
<td>2.40</td>
<td>2.18</td>
<td>1.73</td>
<td>1.72</td>
</tr>
<tr>
<td>3</td>
<td>3.17</td>
<td>2.68</td>
<td>2.39</td>
<td>-0.62</td>
</tr>
<tr>
<td>4</td>
<td>3.12</td>
<td>3.96</td>
<td>0.95</td>
<td>-0.21</td>
</tr>
</tbody>
</table>

Figure 7.3: $L_2$ error over entire mesh for an expanding circle on a structured square. The floor reached with the modified redistancing is due to the addition of the diffusion coefficient $\mu$.

7.4 Rate stick

The rate stick experiment simulates a traveling high-explosive detonation front [9, 12]. Using $Q_2$ (quadratic) elements, we initialize the interface according to

$$\phi = \sqrt{x^2 + (y - 2.5)^2} - 0.5,$$

(7.5)
on a structured $[0, 25] \times [0, 5]$ stick with 500 elements. Once the front is propagated to the end of the stick from left to right, lighting times for hydrodynamics simulations can easily be calculated at some point $\hat{x}$ and time $t$ using the definition $T_{light} := \phi(\hat{x}, t)/u$. The front towards the end of its run is shown in Fig. 7.5.
Figure 7.4: A circle, initialized at the star’s center, stably propagated outwards (left to right).

### 7.5 3D–expanding sphere and merging spheres

Here, we consider two problems: the Expanding Sphere and Merging Spheres [43]. First, we expand a sphere using $Q_2$ (quadratic) elements on a structured $[0, 1] \times [0, 1] \times [0, 1]$ cube with 4,096 elements. The initial level set function $\phi$ is given by

$$
\phi = \sqrt{(x - 0.5)^2 + (y - 0.5)^2 + (z - 0.5)^2} - 0.2. 
$$

This problem represents propagation of an idealized detonation front. Results are in Fig. 7.6.

In the second problem of Merging Spheres, we initialize two spheres at two
corners of a cube according to the equation

\[
\phi = \begin{cases} 
\sqrt{x^2 + y^2 + z^2} - 0.1 & \text{if } x \leq 0.5, \\
\sqrt{(x-1)^2 + y^2 + z^2} - 0.1 & \text{if } x > 0.5.
\end{cases}
\] (7.7)

Similar to the propagation of 2D circles, the analytic solution to the problem for any given time \( t \) is simply an expanded sphere of radius \( 0.1 + ut \), where \( u = 1 \), centered at \([0,0,0]\) for points satisfying \( x \leq 0.5 \), and an expanded sphere of radius \( 0.1 + ut \) centered at \([1,0,0]\) for points satisfying \( x > 0.5 \). Like the expanding sphere, the merging spheres represents the propagation and subsequent collision of two idealized detonation fronts. This problem introduces an added complication, as certain finite difference methods cannot handle the discontinuous derivative along the merging fronts – see [43] for an example. Order two elements are used on the same mesh as the expanding sphere. We see smooth merging in Fig. 7.6.

### 7.6 Corner turning

The next two results test the ability of our solver to handle propagation around sharp and smooth corners. The corner turning problem initializes a circle using

\[
\phi = \sqrt{(x-1)^2 + (y-1)^2} - 0.2,
\] (7.8)

using \( Q_4 \) (4th order) finite elements on the upper right corner of an \([-1,1] \times [-7,1]\) \( L \)-shape mesh with 144 elements. The contours of the level set interface should properly turn the corner and form a ninety-degree angle with the boundary [8].
Figure 7.6: Top row: Simulation of an idealized detonation front with an expanding sphere. Bottom row: Simulation of merging detonation fronts with two expanding spheres.

In this mesh, classic reinitialization may have stability issues depending on order and refinement level. To remedy this, we use a modified reinitialization with $\mu = 0.0005$, which we see in Fig. 7.7 yields stable evolution of the front. In Fig. 7.7, we especially notice that upon initialization, the contours are not orthogonal to the wall, but with the redistancing steps are able to adjust to the desired angle. To better illustrate this, the analytical solution at final time is included for comparison purposes.
Figure 7.7: (a) Initialization and propagation of front on an L-shape mesh. (b) Reinitialization correctly adapts and propagates level set contours around the sharp corner and compares well to final exact solution (c) and (d).
7.7 Propagation around circular hole

Similar to [75], this problem of propagation around a circular hole, using $Q_1$ (linear) elements, initializes $\phi$ as

$$\phi = \sqrt{(x - 0.5)^2 + y^2} - 0.1,$$

which is centered on the bottom edge of a $[0, 1] \times [0, 1]$ NURBS square mesh with 4,096 elements with a hole of radius 0.2 in the middle. The contours of the level set interface should properly address the hole and exhibit appropriate bending after passing through the hole—thus maintaining the property of a signed distance function. We see our method yields stable and accurate approximation of the resulting level set field.

Note that the reinitialization-free diffusive penalty approach given by Eq. (5.8) will not properly adjust its signed distance function to the hole. Instead it will “ignore” the hole when passing through it, i.e. the contours will evolve exactly as if propagated on a square mesh without holes. It is possible that this occurs due to the nature of the right-hand penalty term. If $|\nabla \phi| < 1$, the right-hand side changes sign which can affect stability. The term on the right-hand side

$$\mu(h(x)) \nabla \cdot \left[ \left( 1 - \frac{1}{|\nabla \phi|} \right) \nabla \phi \right],$$

will also become hyperbolic if $|\nabla \phi| < 1$, negatively affecting stability. Proposed fixes for this are to replace this potentially problematic term $1 - 1/|\nabla \phi|$ with the term $\max(1 - |\nabla \phi|^2, 1 - 1/|\nabla \phi|)$, or to take the term $1 - 1/|\nabla \phi|$ outside the derivative and squaring it. Results are presented in Fig. 7.8.

7.8 Letter E

Here we initialize a circle at the bottom left of a paved E mesh according to

$$\phi = \sqrt{(x - 0.09)^2 + (y - 0.09)^2} - 0.5.$$  

We first use order one elements on a mesh size of 11,024, with a constant diffusion coefficient $\mu = 0.0001$. We see stable propagation, but the level set function does
of \( \mu = 0.0001 \). The level set contours correctly turn the corners initially, appearing to form a ninety-degree angle with the wall, but becomes unstable at later times. This is believed to happen in part due to the fact that, at the time of testing, the diffusion coefficient was set as constant and not mesh-dependent, and as such our modified reinitialization, Eq. (5.7), is not able to keep the evolution from going unstable. As earlier results using the modified reinitialization were on meshes with uniform element size, the value of \( \mu \) should be the same throughout the region of interest, and thus the issue was not noticed until run on this unstructured, paved
mesh. Implementation of a mesh-dependent $\mu$ would be a first step toward future work. Results for the evolutions are seen in Figs. 7.9 and 7.10.

**Figure 7.9:** Propagation using order one elements results in stable but inappropriate behavior. Top: The level set $\phi(t_0)$. Bottom: $\phi(t_{\text{final}})$. 
Figure 7.10: Propagation using order two elements results in appropriate behavior for earlier times but goes unstable. Top: The level set \( \phi(t_0) \). Bottom: \( \phi(t_{final}) \).
Acknowledgement: Chapter 7, in part is currently being prepared for submission for publication. This work is coauthored with Dobrev, Veselin; Kolev, Tzanio; Rieben, Robert; Tomov, Vladimir. A portion of this work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.
Chapter 8

Conclusion to level set methods for DSD using high-order FE

We present a level-set method using high-order finite elements using a discontinuous Galerkin method. This method leads to stable front propagation in 2D and 3D on high-order, curved, unstructured meshes. To maintain accuracy during spatial discretization, the level set is periodically reinitialized with a variable constraint using an amalgamation of the classic redistancing equation and a diffusion penalty term. This provides stability to front evolution. Although level sets are useful for a number of physical applications, our focus is on DSD, as correct propagation of a high-explosive burn front allows accurate determination of lighting times for hydrodynamics simulations. In addition, use of a high-order DG method allows for propagation over areas with complex geometries.

The level set algorithm with modified reinitialization was implemented in C++ using the MFEM finite element software library [1]. The solver runs in both serial and parallel; the latter being useful for simulation in 3D or problems with a large number of unknowns. The solver was tested against traditional level set benchmarks, which showed the solver correctly evolves the advection equation and performs proper redistancing using our reinitialization equation. We also successfully tested against DSD benchmarks such as the rate stick and collision of two idealized detonation fronts. Lastly, we tested on less traditional problems, such those that require the navigation of sharp corners and holes. Results show
that our level set solver is successful with corner turning on uniform meshes and propagating around circular holes. However, more adjustments are needed on the most complex case we tested on – a paved E mesh – that involves multiple smooth and sharp corners on an unstructured mesh.

The approach given in this work can be fine-tuned. In particular, further work towards application to DSD is needed:

- **Work towards robustness on all meshes.** As seen in the paved E mesh examples at the end of the previous chapter, the solver does not exhibit appropriate behavior on more complicated meshes. As noted, this is most likely at least partially due to the use of a constant diffusion coefficient $\mu$ as opposed to one that is mesh-dependent. As such, meshes like the paved E mesh, in which the elements are not uniform in size, will not yield stable evolution results. We hypothesize that making the diffusion coefficient mesh-dependent will greatly improve results on these unstructured meshes.

  Additionally, we noted in Section 5.1.2 that the term $1 - 1/|\nabla \phi|$ could be replaced with the term $\max(1 - |\nabla \phi|^2, 1 - 1/|\nabla \phi|)$ to avoid a potential sign change on the right-hand side of the redistancing equation. The term $1 - 1/|\nabla \phi|$ could also be taken outside the derivative and squared, thus preventing any scenarios in which the operator

  $$\mu(h(x))\nabla \cdot \left[ \left( 1 - \frac{1}{|\nabla \phi|} \right) \nabla \phi \right]$$

  becomes hyperbolic, negatively affecting stability.

- **Implement DSD velocity coefficient.** As mentioned in Section 5.2, the velocity coefficient $\bar{u}$ used in this work is constant. In DSD applications, however, $\bar{u} = (D_{CJ} + \alpha(\kappa))\bar{n}$, where $D_{CJ}$ is the Chapman-Jouget constant and $\alpha(\kappa)$ is a function of curvature (i.e. the velocity coefficient should really be dependent on the shock curvature). This should be implemented into the code so that simulations are run more in line with DSD theory.

- **Implement DSD angle boundary conditions.** Also as noted in Section 5.2,
DSD introduces additional boundary conditions on the wall angles, namely,

\[ \cos(\theta) = \vec{n}_b \cdot \vec{n}_s, \]

where \( \theta \) is an experimentally-calculated angle, \( \vec{n}_b \) is the boundary normal, and \( \vec{n}_s \) is the local shock normal. In the current algorithm seen in this thesis, no boundary angle conditions have been implemented. Instead, the boundaries are set to equal the value of the level set function \( \phi \) at the particular boundary point. The additional DSD boundary condition should also be implemented into the code to match better with DSD theory.

**Acknowledgement:** A portion of this work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.
Appendix A

Dormand and Prince function evaluations

We use an 8th-order Dormand and Prince integrator [22, 29]. This method uses coefficients $c_i$, $b_i$, and $a_{ij}$ that satisfy the following reduced system, with $s = 13$:

\[
\sum_{i=1}^{s} b_i c_i^{q-1} = 1/q, \quad q = 1, \ldots, 8 \quad (A.1)
\]

\[
\sum_{i=1}^{i-1} a_{ij} = c_i, \quad i = 1, \ldots, s \quad (A.2)
\]

\[
\sum_{i=1}^{i-1} a_{ij} c_j = c_i^2/2, \quad i = 3, \ldots, s \quad (A.3)
\]

\[
\sum_{i=1}^{i-1} a_{ij} c_j^2 = c_i^3/3, \quad i = 3, \ldots, s \quad (A.4)
\]

\[
\sum_{i=1}^{i-1} a_{ij} c_j^3 = c_i^4/4, \quad i = 6, \ldots, s \quad (A.5)
\]

\[
\sum_{i=1}^{i-1} a_{ij} c_j^4 = c_i^5/5, \quad i = 6, \ldots, s \quad (A.6)
\]

\[
b_2 = b_3 = b_4 = b_5 = 0 \quad (A.7)
\]

\[
a_{i2} = 0 \quad \text{for} \quad i \geq 4, \quad a_{i3} = 0 \quad \text{for} \quad i \geq 6 \quad (A.8)
\]

\[
\sum_{i=j+1}^{s} b_i a_{ij} = b_j (1 - c_j), \quad j = 4, 5, 10, 11, 12 \quad (A.9)
\]
\[
\sum_{i=j+1}^{s} b_i c_i a_{ij} = 0, \quad j = 4, 5 \quad (A.10)
\]
\[
\sum_{i=j+1}^{s} b_i c_i^2 a_{ij} = 0, \quad j = 4, 5 \quad (A.11)
\]
\[
\sum_{i=k+2}^{s} b_i c_i \sum_{j=k+1}^{i-1} a_{ij} a_{j_k} = 0, \quad k = 4, 5 \quad (A.12)
\]
\[
\sum_{i=1}^{s} b_i c_i \sum_{j=1}^{i-1} a_{ij} e_j^5 = 1/48, \quad (A.13)
\]
\[
\sum_{i=1}^{s} b_i c_i \sum_{j=1}^{i-1} a_{ij} e_j^5 = 1/48. \quad (A.14)
\]

The code is freely available at [28].
Appendix B

Zero frequency modes

The zero-frequency modes of a molecule are the eigenvectors of the Hessian matrix of the equilibrium potential energy function that have zero eigenvalues,

\[ 0 = \sum_B \frac{\partial^2 U(\vec{x}_0C)}{\partial \vec{x}_A \partial \vec{x}_B} \cdot \vec{e}_B^\mu. \] (B.1)

We assume that the equilibrium state of the molecule is invariant under rigid spatial translations and rotations (this assumption makes sense to ensure that the equilibrium state of interest to us is one where the molecule is isolated and does not interact with its large scale environment).

If the molecule is invariant under translations, then the forces acting on the individual atoms must also be invariant. In its equilibrium state, the total force acting on each atom must vanish, therefore the gradient of the potential energy function must vanish:

\[ 0 = \frac{\partial U(\vec{x}_0C)}{\partial \vec{x}_A} = \frac{\partial U(\vec{x}_0C + \lambda \vec{c})}{\partial \vec{x}_A}, \] (B.2)

where \( \vec{c} \) is an arbitrary vector that describes the same translation for all the atoms in the molecule, and where \( \lambda \) is an arbitrary parameter that determines the magnitude of the translation. We can express the forces acting on each atom of a molecule that has been translated by an infinitesimal amount using a Taylor series expansion:

\[ 0 = \frac{\partial U(\vec{x}_0C + \lambda \vec{c})}{\partial \vec{x}_A} = \frac{\partial U(\vec{x}_0C)}{\partial \vec{x}_A} + \lambda \sum_B \frac{\partial^2 U(\vec{x}_0C)}{\partial \vec{x}_A \partial \vec{x}_B} \cdot \vec{c} + \mathcal{O}(\lambda^2). \] (B.3)
The first term on the right side of Eq. (B.3) vanishes because of the equilibrium condition, Eq. (B.2). Therefore, the second term on the right side of Eq. (B.3) must also vanish for all values of $\lambda$. It follows that any vector $\vec{c}$ that is the same for all the atoms in a molecule is a zero-frequency eigenvector:

$$0 = \sum_B \frac{\partial^2 U(\vec{x}_{0C})}{\partial \vec{x}_A \partial \vec{x}_B} \cdot \vec{c}. \quad (B.4)$$

The argument for the rotational invariance of the equilibrium state of the molecule is similar (but slightly more involved). Let $\mathbf{R}(\lambda)$ denote a one parameter family of rotation matrices. We assume that $\lambda = 0$ corresponds to the identity rotation: $\mathbf{R}(0) = \mathbf{I}$. The rotational invariance of the equilibrium state of the molecule implies that

$$0 = \frac{\partial U[\vec{x}_{0C}]}{\partial \vec{x}_A} = \frac{\partial U[\mathbf{R}(\lambda) \cdot \vec{x}_{0C}]}{\partial \vec{x}_A}. \quad (B.5)$$

As before, we perform a Taylor expansion of the expression for the forces acting on an equilibrium molecule that has been rotated an infinitesimal amount:

$$0 = \frac{\partial U[\mathbf{R}(\lambda) \cdot \vec{x}_{0C}]}{\partial \vec{x}_A} = \frac{\partial U[\vec{x}_{0C}]}{\partial \vec{x}_A} + \lambda \sum_B \frac{\partial^2 U[\vec{x}_{0C}]}{\partial \vec{x}_A \partial \vec{x}_B} \cdot \left. \frac{d\mathbf{R}}{d\lambda} \right|_{\lambda=0} \cdot \vec{x}_{0B} + \mathcal{O}(\lambda^2). \quad (B.6)$$

As we have seen, cf. Eq. (2.7), the derivative of any rotation matrix is an antisymmetric matrix. In this case this matrix can be written as

$$\left. \frac{dR_{ij}}{d\lambda} \right|_{\lambda=0} = -\sum_k \epsilon_{ijk} v_k, \quad (B.7)$$

where the vector $\vec{v}$ determines the direction and magnitude of the infinitesimal rotation. It follows that Eq. (B.6) can be re-written as

$$0 = \frac{\partial U[\mathbf{R}(\lambda) \cdot \vec{x}_{0C}]}{\partial \vec{x}_A} = \frac{\partial U[\vec{x}_{0C}]}{\partial \vec{x}_A} + \lambda \sum_B \frac{\partial^2 U[\vec{x}_{0C}]}{\partial \vec{x}_A \partial \vec{x}_B} \cdot (\vec{v} \times \vec{x}_{0B}) + \mathcal{O}(\lambda^2). \quad (B.8)$$

The first term on the right side of Eq. (B.8) vanishes because of the equilibrium condition, Eq. (B.5). Therefore, the second term on the right side of Eq. (B.8) must vanish for all values of $\lambda$. It follows that any vector of the form $\vec{v} \times \vec{x}_{0A}$, where $\vec{v}$ is the same for all the atoms in a molecule, is a zero-frequency eigenvector:

$$0 = \sum_B \frac{\partial^2 U[\vec{x}_{0C}]}{\partial \vec{x}_A \partial \vec{x}_B} \cdot (\vec{v} \times \vec{x}_{0B}). \quad (B.9)$$
The eigenvectors of the zero frequency modes (for generic molecules) therefore consist of rigid translations

\[ \vec{c}_A^t = \vec{c}, \]  

where \( \vec{c} \) is a constant vector that determines the magnitude and direction of the translation, and rigid rotations,

\[ \vec{e}_A^r(\vec{v}) = \vec{v} \times \vec{x}_{0A}, \]

where \( \vec{v} \) is a constant vector that determines the axis and magnitude of the rotation. Any linear combination of these modes is also a zero frequency mode. So it is often convenient to choose an orthonormal set of eigenvectors under the orthogonality condition given in Eq. (2.13).

First consider the translations. Let \( \vec{e}_A^t(\vec{c}) = \vec{c} \) and \( \vec{e}_A^t(\vec{d}) = \vec{d} \) be two of the translation eigenvectors. Their eigenvector inner product is given by:

\[ \sum_A m_A \vec{e}_A^t(\vec{c}) \cdot \vec{e}_A^t(\vec{d}) = \vec{c} \cdot \vec{d}. \]

It follows that these vectors will be orthogonal to one another iff, \( \vec{c} \cdot \vec{d} = 0 \), and will be normalized properly iff, \( \vec{c} \cdot \vec{c} = \vec{d} \cdot \vec{d} = 1 \). Therefore, we can choose the translations along the Cartesian coordinate directions as a basis for the translation eigenvectors:

\[ \vec{e}_A^t(\vec{x}) = (1, 0, 0), \]

\[ \vec{e}_A^t(\vec{y}) = (0, 1, 0), \]

\[ \vec{e}_A^t(\vec{z}) = (0, 0, 1). \]

Next consider the rotations. Let \( \vec{e}_A^r(\vec{v}) = \vec{v} \times \vec{x}_{0A} \) and \( \vec{e}_A^r(\vec{w}) = \vec{w} \times \vec{x}_{0A} \) denote two of the rotation eigenvectors. Their eigenvector inner product is given by:

\[ \sum_A \frac{m_A}{M} \vec{e}_A^r(\vec{v}) \cdot \vec{e}_A^r(\vec{w}) = \vec{v} \cdot \left[ \sum_A \frac{m_A}{M} (\vec{x}_{0A} \cdot \vec{x}_{0A} - \vec{x}_{0A} \otimes \vec{x}_{0A}) \right] \cdot \vec{w} \]

\[ = \frac{1}{M} \vec{v} \cdot Q_0 \cdot \vec{w}, \]
where $\mathbf{Q}_0$ is the moment of inertia tensor of the unperturbed equilibrium molecule. Two rotations will be orthogonal in this sense iff $\vec{v} \cdot \mathbf{Q}_0 \cdot \vec{w} = 0$, and will be normalized properly iff $\vec{v} \cdot \mathbf{Q}_0 \cdot \vec{v} = \vec{w} \cdot \mathbf{Q}_0 \cdot \vec{w} = M$. Since $\mathbf{Q}_0$ is a positive definite symmetric matrix, it is straightforward to construct an orthonormal set of rotations using Gramm-Schmidt (for example). A more canonical set of eigenvectors could also be constructed by taking the suitably normalized eigenvectors of the moment of inertia tensor $\mathbf{Q}_0$.

We also need to consider the inner product between one of the translation eigenvectors $\vec{e}_A^t(\vec{c}) = \vec{c}$ and one of the rotation eigenvectors $\vec{e}_A^r(\vec{v}) = \vec{v} \times \vec{x}_{0A}$:

$$\sum_A \frac{m_A}{M} \vec{e}_A^t(\vec{c}) \cdot \vec{e}_A^r(\vec{v}) = (\vec{c} \times \vec{v}) \cdot \sum_A \frac{m_A}{M} \vec{x}_{0A} = 0. \quad (B.18)$$

These rotations and translations are orthogonal, therefore, because of the condition in Eq. (2.6).

Finally, we note that the translations $\vec{e}_A^t(\vec{c}) = \vec{c}$ and rotations $\vec{e}_A^r(\vec{v}) = \vec{v} \times \vec{x}_{0A}$ will be orthogonal to all of the other non-zero frequency mode eigenvectors $\vec{e}_A^\mu$. These orthogonality conditions are given by:

$$0 = \sum_A \frac{m_A}{M} \vec{e}_A^\mu \cdot \vec{e}_A^t(\vec{c}) = \sum_A \frac{m_A}{M} \vec{e}_A^\mu \cdot \vec{c}, \quad (B.19)$$

$$0 = \sum_A \frac{m_A}{M} \vec{e}_A^\mu \cdot \vec{e}_A^r(\vec{v}) = -\sum_A \frac{m_A}{M} (\vec{e}_A^\mu \times \vec{x}_{0A}) \cdot \vec{v}. \quad (B.20)$$

These orthogonality conditions will hold for arbitrary values of the vectors $\vec{c}$ and $\vec{v}$. Therefore the needed orthogonal conditions can also be written in the form

$$0 = \sum_A \frac{m_A}{M} \vec{e}_A^\mu = \sum_A \frac{m_A}{M} \vec{e}_A^\mu \times \vec{x}_{0A}. \quad (B.21)$$

These conditions will hold for each non-zero frequency mode $\mu$.

Appendix B, in full is currently being prepared for submission for publication. This work is coauthored with Holst, Michael; Lindblom, Lee.
Appendix C

Exact normal-mode basis representation

The following is a derivation of an exact normal-mode basis representation. Here, as opposed to the large-scale dynamics formulation in Chapter 2, no approximations or simplifications are made. This exact representation will be useful for error quantification.

Recall that the position $\vec{x}_A$ of each molecule can be written in the form

$$\vec{x}_A = \vec{x}_{CM} + \mathbf{R} \cdot (\vec{x}_0 + \delta \vec{x}_A) = \vec{x}_{CM} + \Delta \vec{X}_A.$$  (C.1)

The position of the center of mass, $\vec{x}_{CM}$, is determined by the evolution of the total momentum, $\vec{P}$, of the molecule:

$$M \frac{d^2 \vec{x}_{CM}}{dt^2} = \sum_A m_A \frac{d^2 \vec{x}_A}{dt^2} = \frac{d\vec{P}}{dt} = -\sum_A \frac{\partial U(\vec{x}_C)}{\partial \vec{x}_A}. \quad \text{(C.2)}$$

The rotational degrees of freedom of $\mathbf{R}$ are coupled to the degrees of freedom of the individual atoms, $\delta \vec{x}_A$. Substituting the second derivative of $\vec{x}_A$ into Newton’s law, Eq. (2.1), and using the expression from Eq. (C.2), gives us a constraint for both the rotational degrees of freedom, $\mathbf{R}(t)$ and $\vec{\Omega}$, and the $\delta \vec{x}_A$:

$$\mathbf{R} \cdot \frac{d^2 \delta \vec{x}_A}{dt^2} = \Delta \vec{X}_A \times \frac{d\vec{\Omega}}{dt} - 2\vec{\Omega} \times \left( \mathbf{R} \cdot \frac{d\delta \vec{x}_A}{dt} \right) + \left( \vec{\Omega} \cdot \vec{\Omega} \mathbf{I} - \vec{\Omega} \otimes \vec{\Omega} \right) \cdot \Delta \vec{X}_A$$
\[- \frac{1}{m_A} \frac{\partial U}{\partial \vec{x}_A} + \frac{1}{M} \sum_B \frac{\partial U}{\partial \vec{x}_B}. \quad (C.3)\]

The above gives us 3N equations, where N is the number of atoms. However, as seen in Chapter 2, the \( \delta \vec{x}_A \) comprise of 3N equations which include 3 degrees of freedom for rigid translations and 3 degrees of freedom for rigid rotations. Thus Eq. (C.3) is underdetermined and has no solution. We can remedy this issue by projecting the atomistic deviations \( \delta \vec{x}_A \) onto vector fields that represent a rigid rotation about the center of mass, and then having these deviations vanish on these fields. This constraint will give us viable equations with which to obtain a unique solution to our spatial and individual atom variables. Using the fact that we can represent rotations using cross products, we denote our rotation vector fields as

\[
\vec{e}_{rA}(\vec{v}) = \vec{v} \times \vec{x}_{0A}, \quad (C.4)
\]

where \( \vec{v} \) is an arbitrary vector determining the axis and the magnitude of the rotation. We then project the \( \delta \vec{x}_A \) onto this field

\[
0 = \sum_A m_A \vec{e}_{rA}(\vec{v}) \cdot \delta \vec{x}_A = \vec{v} \cdot \sum_A m_A \vec{x}_{0A} \times \delta \vec{x}_A. \quad (C.5)
\]

This rotational constraint, Eq. (C.5), along with its time derivatives, hold for any arbitrary vector \( \vec{v} \). Therefore, the other two constraints are

\[
0 = \sum_A m_A \vec{x}_{0A} \times \frac{d \delta \vec{x}_A}{dt} = \sum_A m_A \vec{x}_{0A} \times \frac{d^2 \delta \vec{x}_A}{dt^2}. \quad (C.6)
\]

As mentioned above, the \( \delta \vec{x}_A \) also may contain a translation that is redundant with the position of the center of mass. With this in mind, we also need a constraint that takes care of these rigid translations. Using the same method as above, we project the \( \delta \vec{x}_A \) onto vector fields that represent rigid translations. These have the form

\[
\vec{e}_{tA}(\vec{v}) = \vec{c}, \quad (C.7)
\]

where \( \vec{c} \) is some arbitrary constant vector that determines the direction and magnitude of the translation. Projecting, we have

\[
0 = \sum_A m_A \vec{e}_{tA}(\vec{v}) \cdot \delta \vec{x}_A = \sum_A m_A \vec{c} \cdot \delta \vec{x}_A. \quad (C.8)
\]
However, this is already enforced thanks to Eq. (2.6).

We are ready to derive the equations for the rotational variables. Using the right-hand projection Eq. (C.6) and plugging in Eq. (C.3), we obtain an equation for evolution of the angular velocity $\Omega$:

$$U \cdot R^{-1} \cdot \frac{d\vec{\Omega}}{dt} = \vec{V},$$

(C.9)

where the tensor $U$ and vector $\vec{V}$ are given by

$$U = \sum_A \left( \frac{m_A}{M} \right) \left[ I (\vec{x}_{0A} + \delta\vec{x}_A) \cdot \vec{x}_{0A} - (\vec{x}_{0A} + \delta\vec{x}_A) \otimes \vec{x}_{0A} \right],$$

(C.10)

$$\vec{V} = \sum_A \left( \frac{m_A}{M} \right) \left[ 2 \left( \vec{x}_{0A} \cdot R^{-1} \cdot \vec{\Omega} \right) \frac{d\delta\vec{x}_A}{dt} - 2 \left( \vec{x}_{0A} \cdot \frac{d\delta\vec{x}_A}{dt} \right) R^{-1} \cdot \vec{\Omega} - \left( \vec{\Omega} \cdot \Delta\vec{X}_A \right) \vec{x}_{0A} \times \left( R^{-1} \cdot \vec{\Omega} \right) \right] \frac{1}{m_A} \vec{x}_{0A} \times \left( R^{-1} \cdot \frac{\partial U}{\partial \vec{x}_A} \right).$$

(C.11)

So, we have Equation (C.9), the evolution equation that determines $\vec{\Omega}$, which in turn determines the rotation matrix $R$ through the following re-written form of Eq. (2.7),

$$\frac{dR}{dt} = -*\Omega \cdot R.$$

(C.12)

Equation (C.9), together with the evolution equations for the other dynamical fields, will determine $\vec{\Omega}$ so long as the tensor $U$ is invertible. If $\delta\vec{x}_A = 0$, then $U$ becomes $Q_0$, the moment of inertia tensor for the equilibrium state of the molecule. The tensor $Q_0$ is positive definite (unless all the atoms in the molecule line along a line), so in the generic case $U$ is also invertible for arbitrary but sufficiently small $\delta\vec{x}_A$.

We also know that $U$ is singular in the non-physical case where the positions of all the perturbed atoms $\vec{x}_{0A} + \delta\vec{x}_A$ lie along a single line. Although it is possible that there are other situations where $U$ is singular, we conjecture that $U$ is invertible except in nonphysical cases.

The overdetermined Eq. (C.3) also presents a problem for obtaining evolution equations for the $\delta\vec{x}_A$. We use a similar method of projecting the $\delta\vec{x}_A$ to obtain an evolution equation that yields a unique solution.
Recall that general $\delta \bar{x}_A$ can be expressed as a linear combination of these eigenvectors:

$$\delta \bar{x}_A = \sum_{\mu} A_\mu(t) \bar{e}_A^\mu,$$  \hspace{1cm} \text{(C.13)}

where we exclude the $\mu$ such that $\omega_\mu = 0$. Recall also that the displacements $\delta \bar{x}_A$ can be projected onto the normal mode eigenvectors, even if the amplitudes $A_\mu$ are not small, since the eigenvectors form a complete basis:

$$\sum_A \frac{m_A}{M} \bar{e}_A^\mu \cdot \delta \bar{x}_A = A_\mu(t).$$  \hspace{1cm} \text{(C.14)}

Similarly, we can project the full molecular dynamics equations of motion, Eq. (2.1), onto the normal mode eigenvector basis:

$$\sum_A \frac{m_A}{M} \bar{e}_A^\mu \cdot \mathbf{R}^{-1} \cdot \frac{d^2 \bar{x}_A}{dt^2} = -\sum_A \frac{1}{M} \bar{e}_A^\mu \cdot \mathbf{R}^{-1} \cdot \frac{\partial U(\bar{x}_C)}{\partial \bar{x}_A}.$$  \hspace{1cm} \text{(C.15)}

The left side of this expression can be expanded by plugging in for $d^2 \bar{x}_A/dt^2$:

$$\sum_A \frac{m_A}{M} \bar{e}_A^\mu \cdot \mathbf{R}^{-1} \cdot \frac{d^2 \bar{x}_A}{dt^2} = \sum_A \frac{m_A}{M} \bar{e}_A^\mu \left\{ \frac{d^2 \delta \bar{x}_A}{dt^2} - 2 \mathbf{R}^{-1} \cdot \mathbf{R} \cdot \frac{\partial U(\bar{x}_C)}{\partial \bar{x}_A} \right. \hspace{1cm}
$$

$$+ \mathbf{R}^{-1} \cdot \left[ ** \mathbf{\Omega} \cdot \mathbf{\Omega} - \frac{d** \mathbf{\Omega}}{dt} \right] \cdot \Delta \bar{x}_A \hspace{1cm}
$$

$$- \frac{1}{M} \mathbf{R}^{-1} \cdot \mathbf{R} \cdot \left( \sum_B \frac{\partial U(\bar{x}_C)}{\partial \bar{x}_B} \right) \right\},$$

$$= \frac{d^2 A_\mu}{dt^2} + \sum_\nu S^{\mu\nu} \frac{dA_\nu}{dt} + \sum_\nu T^{\mu\nu} A_\nu$$

$$- \sum_A \frac{m_A}{M} \left( \mathbf{R} \cdot \bar{e}_A^\mu \right) \left[ \left( \mathbf{R} \cdot \bar{x}_A \right) \times \frac{d\mathbf{\Omega}}{dt} \right]$$

$$+ \sum_A \frac{m_A}{M} \left\{ \left[ \left( \mathbf{R} \cdot \bar{e}_A^\mu \right) \cdot \mathbf{\Omega} \right] \left[ \left( \mathbf{R} \cdot \bar{x}_A \right) \cdot \mathbf{\Omega} \right] \right. \hspace{1cm}
$$

$$- \left( \bar{e}_A^\mu \cdot \bar{x}_A \right) \left[ \mathbf{\Omega} \cdot \mathbf{\Omega} \right] \right\},$$  \hspace{1cm} \text{(C.16)}

where $S^{\mu\nu}$ and $T^{\mu\nu}$ are defined as

$$S^{\mu\nu} = -2 \sum_A \frac{m_A}{M} \left( \mathbf{R} \cdot \bar{e}_A^\mu \right) \cdot \left[ \left( \mathbf{R} \cdot \bar{e}_A^\nu \right) \times \mathbf{\Omega} \right],$$  \hspace{1cm} \text{(C.17)}
\[ T^{\mu \nu} = - \sum_A \frac{m_A}{M} (\mathbf{R} \cdot \hat{e}_A^\mu) \cdot \left( (\mathbf{R} \cdot \hat{e}_A^\nu) \times \frac{d\tilde{\Omega}}{dt} \right) \]
\[ + \sum_A \frac{m_A}{M} (\mathbf{R} \cdot \hat{e}_A^\mu) \cdot \left[ (\mathbf{R} \cdot \hat{e}_A^\nu) \cdot \tilde{\Omega} \right] - \tilde{\Omega} \cdot \tilde{\Omega} \delta^{\mu \nu}. \]  

(C.18)

Similarly the right side of Eq. (C.15) together with the force terms from the right side of Eq. (C.16) can be used to define right-hand side, i.e. the force on the \( \mu \)th mode:

\[ \mathcal{F}^\mu = - \sum_A \frac{1}{M} (\mathbf{R} \cdot \hat{e}_A^\mu) \cdot \frac{\partial U(x_C)}{\partial x_A} \]
\[ + \sum_A \frac{m_A}{M} (\mathbf{R} \cdot \hat{e}_A^\mu) \cdot \left( (\mathbf{R} \cdot \bar{x}_{0A}) \times \frac{d\bar{\Omega}}{dt} \right) \]
\[ - \sum_A \frac{m_A}{M} \left\{ \left( (\mathbf{R} \cdot \hat{e}_A^\mu) \cdot \bar{\Omega} \right) \left[ (\mathbf{R} \cdot \bar{x}_{0A}) \cdot \bar{\Omega} \right] - (\hat{e}_A^\mu \cdot \bar{x}_{0A}) \left( \bar{\Omega} \cdot \bar{\Omega} \right) \right\}. \]  

(C.19)

Combining the above gives us the evolution equations for the individual mode amplitudes:

\[ \frac{d^2 A_\mu}{dt^2} + \sum_\nu S^{\mu \nu} \frac{dA_\nu}{dt} + \sum_\nu T^{\mu \nu} A_\nu = \mathcal{F}^\mu. \]  

(C.20)

As no approximations are simplifications were used in this derivation, integrating these equations produces the same dynamical evolution as the original equations of full, classical molecular dynamics, Eq. (2.1).

For error quantification on the mode amplitudes, performed in Section 2.4, we can determine which \( A_\mu \) are well described by the analytic normal mode solution, and which are not, by defining the following normal mode force error measure:

\[ \mathcal{E}^\mu = \omega_\mu^2 A_\mu + \mathcal{F}^\mu - \sum_\nu S^{\mu \nu} \frac{dA_\nu}{dt} - \sum_\nu T^{\mu \nu} A_\nu. \]  

(C.21)

Using this expression, we may re-write Eq. (C.20) in the following simplified way,

\[ \frac{d^2 A_\mu}{dt^2} = -\omega_\mu^2 A_\mu + \mathcal{E}^\mu. \]  

(C.22)

This equation is again an exact representation of the molecular dynamics evolution equations.

Appendix C is a version of unpublished notes by Lee Lindblom and the thesis author [38].
Appendix D

Calculating rotation evolution using quaternions

The following is a more detailed look at calculating rotation evolution using quaternions as opposed to the $3 \times 3$ rotation matrices used in Eq. (2.15). It includes a derivation of the quaternion evolution equation used and provides conversions between rotation matrix and quaternion, and vice versa.

When numerically evolving rotations, using quaternions has several advantages over $3 \times 3$ rotation matrices. Not only are they easier to represent (4 numbers as opposed to 9), but they can be easily constructed given an axis and angle of rotation. Furthermore, if using matrices when composing rotations numerically, due to roundoff error the final matrix may not be orthogonal. Quaternions can simply be normalized to represent the rotation, whereas it can be hard to convert a non-orthogonal matrix back to an orthogonal one. So instead, we will describe rotations using unit quaternions $q(t) = q_0(t) + q_1(t)i + q_2(t)j + q_3(t)k$, where

$$q_0^2(t) + q_1^2(t) + q_2^2(t) + q_3^2(t) = 1.$$  \hfill (D.1)

The following is a derivation of the evolution equation for quaternions.

Given a position vector at an initial time, $\vec{x}_A(t_0)$, we can rotate this vector by a quaternion $q(t)$ using conjugation:

$$\hat{\vec{x}}_A(t) = q(t)\vec{x}_A(t_0)q^{-1}(t),$$  \hfill (D.2)
where $\hat{x}_A(t)$ is the result of the rotation. Differentiating Eq. (D.2), we get
\[ \frac{d\hat{x}_A(t)}{dt} = \frac{dq(t)}{dt}A(t_0)q^{-1}(t) + q(t)\hat{x}_A(t_0)\frac{dq^{-1}(t)}{dt}. \]  
(D.3)

From Eqs. (D.2) and (D.3),
\[ \frac{d\hat{x}_A(t)}{dt} = \frac{dq(t)}{dt}q^{-1}(t)\hat{x}_A(t) + \hat{x}_A(t)q(t)\frac{dq^{-1}(t)}{dt}. \]  
(D.4)

Since $q(t)q^{-1}(t) = 1$, by differentiation:
\[ \frac{dq(t)}{dt}q^{-1}(t) + q(t)\frac{dq^{-1}(t)}{dt} = 0. \]  
(D.5)

Using Eqs. (D.4) and (D.5):
\[ \frac{d\hat{x}_A(t)}{dt} = \frac{dq(t)}{dt}q^{-1}(t)\hat{x}_A(t) - \hat{x}_A(t)\frac{dq(t)}{dt}q^{-1}(t). \]  
(D.6)

Let $p(t) = \frac{dq(t)}{dt}q^{-1}(t)$. We will show $p(t)$ is a vector. Multiplying $\frac{dq(t)}{dt}$ with $q^{-1}(t)$ using the Hamilton product, we get that the scalar part of $p(t)$ is
\[ \frac{dq_0(t)}{dt}q_0(t) + \frac{dq_1(t)}{dt}q_1(t) + \frac{dq_2(t)}{dt}q_2(t) + \frac{dq_3(t)}{dt}q_3(t). \]  
(D.7)

However, since $q(t)$ is a unit quaternion, by differentiating Eq. (D.1) we see Eq. (D.7) is zero. So $p(t)$ is a vector. Since $p(t)$ and $\hat{x}_A(t)$ are both vectors:
\[ p(t)\hat{x}_A(t) - \hat{x}_A(t)p(t) = 2p(t) \times \hat{x}_A(t). \]  
(D.8)

We also know that
\[ \frac{d\hat{x}_A(t)}{dt} = \Omega(t) \times \hat{x}_A(t), \]  
(D.9)

where $\Omega(t) = (\Omega_x, \Omega_y, \Omega_z)^T$ is our angular velocity.

Then, from Eqs. (D.6), (D.8), and (D.9),
\[ \Omega(t) = 2p(t) = 2\frac{dq(t)}{dt}q^{-1}(t), \]  
(D.10)

so
\[ \frac{dq(t)}{dt} = \frac{1}{2}\Omega(t)q(t). \]  
(D.11)

Multiplying this out using the Hamilton product:
\[ \frac{dq_0(t)}{dt} = -\frac{1}{2}(\Omega_xq_1 + \Omega_yq_2 + \Omega_zq_3) \]  
(D.12)
\[
\frac{dq_1(t)}{dt} = \frac{1}{2}(\Omega_x q_0 + \Omega_y q_3 - \Omega_z q_2) \quad \text{(D.13)}
\]
\[
\frac{dq_2(t)}{dt} = \frac{1}{2}(-\Omega_x q_3 + \Omega_y q_0 + \Omega_z q_1) \quad \text{(D.14)}
\]
\[
\frac{dq_3(t)}{dt} = \frac{1}{2}(\Omega_x q_2 - \Omega_y q_1 + \Omega_z q_0). \quad \text{(D.15)}
\]

Note that the result, \(dq(t)/dt\), will usually need to be normalized. This equation allows us to evolve the rotations in a numerically stable way.

We can make these evolution equations for \(q_i\) automatically preserve the constraint,
\[
0 = C \equiv q_0^2 + q_1^2 + q_2^2 + q_3^2 - 1, \quad \text{(D.16)}
\]
by modifying the above evolution system in the following way:
\[
\frac{dq_0(t)}{dt} = -\frac{1}{2} (\Omega_x q_1 + \Omega_y q_2 + \Omega_z q_3) - \frac{1}{8} \eta q_0 C \quad \text{(D.17)}
\]
\[
\frac{dq_1(t)}{dt} = \frac{1}{2} (\Omega_x q_0 + \Omega_y q_3 - \Omega_z q_2) - \frac{1}{8} \eta q_1 C \quad \text{(D.18)}
\]
\[
\frac{dq_2(t)}{dt} = \frac{1}{2} (-\Omega_x q_3 + \Omega_y q_0 + \Omega_z q_1) - \frac{1}{8} \eta q_2 C \quad \text{(D.19)}
\]
\[
\frac{dq_3(t)}{dt} = \frac{1}{2} (\Omega_x q_2 - \Omega_y q_1 + \Omega_z q_0) - \frac{1}{8} \eta q_3 C. \quad \text{(D.20)}
\]

This form of the equations implies the following evolution equation for the constraint:
\[
\frac{dC}{dt} = -\eta(C + 1)C. \quad \text{(D.21)}
\]

Therefore, if we choose \(\eta\) to be positive constant, \(\eta > 0\), this version of the evolution equations will ensure that small constraint violations will be exponentially suppressed as the system evolves. If the constraint becomes negative, this constraint suppression mechanism becomes somewhat less efficient. But since this constraint is bounded below by \(-1\), the constraints will always be driven back toward zero for any value of the quaternions \(q_i\).

Since our rotation matrix \(R\) and its time derivative \(dR/dt\) appear in several of our equations, we need to be able to convert back and forth from quaternion to rotation matrix. Given a quaternion \(q(t) = q_0(t) + q_1(t)i + q_2(t)j + q_3(t)k\), the
conversion to an orthogonal matrix $R$ corresponding to a rotation is:

$$R = \begin{pmatrix}
q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\
2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\
2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2
\end{pmatrix}. \quad (D.22)$$

Using the same matrix $R$ and unit quaternion $q(t)$ as above, let’s now convert back into a quaternion. It is straightforward to show that

$$q_0 = \frac{1}{2}\sqrt{1 + \text{tr}(R)}, \quad (D.23)$$

where $\text{tr}(R)$ is the trace of $R$. For ease of notation, denote the elements of $R$ as

$$R = \begin{pmatrix}
R_{11} & R_{12} & R_{13} \\
R_{21} & R_{22} & R_{23} \\
R_{31} & R_{32} & R_{33}
\end{pmatrix}. \quad (D.24)$$

We can then solve for the rest of the parameters, giving us $q(t)$:

$$q(t) = q_0 + \left( \frac{R_{32} - R_{23}}{4q_0} \right) i + \left( \frac{R_{12} - R_{21}}{4q_0} \right) j + \left( \frac{R_{13} - R_{31}}{4q_0} \right) k. \quad (D.25)$$

However, if the argument of the square root is negative or very small, this will result in numerical instability. Since we can similarly calculate the other $q_i(t)$ using the diagonal of $R$, we first find which $q_i(t)$ has the largest square root argument. Then, we proceed to solve for the remaining three parameters. If using the diagonal to solve for $q_1$:

$$q(t) = \left( \frac{R_{32} - R_{23}}{4q_1} \right) i + \left( \frac{1}{2}\sqrt{1 + R_{11} - R_{22} - R_{33}} \right) i + \left( \frac{R_{13} - R_{31}}{4q_1} \right) j + \left( \frac{R_{21} - R_{12}}{4q_1} \right) k. \quad (D.26)$$

If solving for $q_2$:

$$q(t) = \left( \frac{R_{13} - R_{31}}{4q_2} \right) i + \left( \frac{R_{12} + R_{21}}{4q_2} \right) i + \left( \frac{1}{2}\sqrt{1 - R_{11} + R_{22} - R_{33}} \right) j + \left( \frac{R_{23} + R_{32}}{4q_2} \right) k. \quad (D.27)$$

If solving for $q_3$:

$$q(t) = \left( \frac{R_{21} - R_{12}}{4q_3} \right) i + \left( \frac{R_{13} + R_{31}}{4q_3} \right) i.
\[ + \left( \frac{R_{23} + R_{32}}{4q_3} \right) j + \left( \frac{1}{2} \sqrt{1 - R_{11} - R_{22} + R_{33}} \right) k. \] (D.28)

Appendix D is a version of unpublished notes by Lee Lindblom and the thesis author [38].
Appendix E

Converting from Cartesian to mode variables

When the mode amplitudes $A_\mu$ and their time derivatives $dA_\mu/dt$ are known, we can convert to the Cartesian variables $\vec{x}_A$ and $d\vec{x}_A/dt$ using Eqs. (2.15) and (2.10), respectively. The following section highlights the equations needed to, given Cartesian variables, convert to the mode variables $A_\mu$ and $dA_\mu/dt$.

We start off by noting that the $\delta\vec{x}_A$ and $d\delta\vec{x}_A/dt$ are linear combinations of the mode eigenvectors $\vec{e}_A^\mu$. In Appendix B we showed the $\vec{e}_A^\mu$ are orthogonal to the translation and rotation eigenvectors; thus the $\delta\vec{x}_A$ are also orthogonal to these eigenvectors. Now, if we dot a rotation eigenvector into our bulk rotation matrix $R$, the result is simply another rotation. Therefore, $R \cdot \delta\vec{x}_A$ as well as its time derivative counterpart will be orthogonal to the rotation eigenvectors, i.e.

$$\sum_A m_A (R \cdot \delta\vec{x}_A) \cdot \vec{e}_A^r = 0,$$

(E.1)

and

$$\sum_A m_A (R \cdot \frac{d\delta\vec{x}_A}{dt}) \cdot \vec{e}_A^r = 0.$$  

(E.2)

Multiplying the known variables $\vec{x}_A$ and $\vec{x}_{CM}$ by all rotation eigenvectors $\vec{e}_A^r = \vec{v} \times \vec{x}_{0A}$, we get,

$$\sum_A m_A (\vec{x}_A - \vec{x}_{CM}) \cdot \vec{e}_A^r = \sum_A m_A (\vec{x}_A - \vec{x}_{CM}) \cdot (\vec{v} \times \vec{x}_{0A}).$$  

(E.3)
Using Eq. (C.1) and Eq. (E.1),
\[
\sum_A m_A (\vec{x}_A - \vec{x}_{CM}) \cdot (\vec{v} \times \vec{x}_0A) = \sum_A m_A [R \cdot (\vec{x}_{0A} + \delta \vec{x}_A)] \cdot (\vec{v} \times \vec{x}_{0A})
\]
\[
= \sum_A m_A (R \cdot \vec{x}_{0A}) \cdot (\vec{v} \times \vec{x}_{0A}). \tag{E.4}
\]

Rewriting, we have
\[
\vec{v} \cdot \sum_A m_A (\vec{x}_A - \vec{x}_{CM}) \times \vec{x}_{0A} = \vec{v} \cdot \sum_A m_A (R \cdot \vec{x}_{0A}) \times \vec{x}_{0A}. \tag{E.5}
\]

This holds for any \( \vec{v} \), so
\[
\sum_A m_A (\vec{x}_A - \vec{x}_{CM}) \times \vec{x}_{0A} = \sum_A m_A (R \cdot \vec{x}_{0A}) \times \vec{x}_{0A}, \tag{E.6}
\]
where \( \vec{x}_A, \vec{x}_{CM}, \) and \( \vec{x}_{0A} \) are known. Recall that \( R \) is uniquely determined by the four parameters in our quaternion \( q(t) = q_0(t) + q_1(t)i + q_2(t)j + q_3(t)k \) (Eq. (D.22)). Eq. (E.6) plus the unit quaternion constraint Eq. (D.1) provide us with the four equations needed to find \( R \).

Once \( R \) is known, we can use a similar projection onto all of the mode eigenvectors to determine the mode amplitudes \( A_\mu \):
\[
\sum_A m_A [R^{-1} \cdot (\vec{x}_A - \vec{x}_{CM})] \cdot \vec{\epsilon}_A^\mu = \sum_A m_A (\vec{x}_{0A} + \delta \vec{x}_A) \cdot \vec{\epsilon}_A^\mu
\]
\[
= \sum_A m_A \vec{x}_{0A} \cdot \vec{\epsilon}_A^\mu + \sum_A m_A \delta \vec{x}_A \cdot \vec{\epsilon}_A^\mu. \tag{E.7}
\]

Using Eq. (C.14),
\[
\sum_A m_A [R^{-1} \cdot (\vec{x}_A - \vec{x}_{CM})] \cdot \vec{\epsilon}_A^\mu = \sum_A m_A \vec{x}_{0A} \cdot \vec{\epsilon}_A^\mu + M A_\mu. \tag{E.8}
\]

Therefore,
\[
A_\mu = \sum_A \frac{m_A}{M} [R^{-1} \cdot (\vec{x}_A - \vec{x}_{CM})] \cdot \vec{\epsilon}_A^\mu. \tag{E.9}
\]

Similarly, we can multiply the known variables \( d\vec{x}_A/dt \) and \( d\vec{x}_{CM}/dt \) by all rotation eigenvectors \( \vec{\epsilon}_A^r \) to determine \( \vec{\Omega} \). Using Eq. (2.10),
\[
\sum_A m_A \left( \frac{d\vec{x}_A}{dt} - \frac{d\vec{x}_{CM}}{dt} \right) \cdot \vec{\epsilon}_A^r = \sum_A m_A \left( \frac{d\vec{x}_A}{dt} - \frac{d\vec{x}_{CM}}{dt} \right) \cdot (\vec{v} \times \vec{x}_{0A})
\]
\[
= \sum_A m_A \left( R \cdot \frac{d\delta x_A}{dt} + \vec{\Omega} \times \Delta \vec{X}_A \right) \cdot (\vec{v} \times x_{0A}).
\] (E.10)

Using Eq. (E.2) and rearranging,
\[
\vec{v} \cdot \sum_A m_A \left( \frac{d\vec{x}_A}{dt} - \frac{d\vec{x}_{CM}}{dt} \right) \times \vec{x}_{0A} = \vec{v} \cdot \sum_A m_A (\vec{\Omega} \times \Delta \vec{X}_A) \times \vec{x}_{0A},
\] (E.11)
so
\[
\sum_A m_A \left( \frac{d\vec{x}_A}{dt} - \frac{d\vec{x}_{CM}}{dt} \right) \times \vec{x}_{0A} = \sum_A m_A (\vec{\Omega} \times \Delta \vec{X}_A) \times \vec{x}_{0A},
\] (E.12)
where \(d\vec{x}_A/dt, d\vec{x}_{CM}/dt, \vec{x}_{0A}, \) and \(\Delta \vec{X}_A\) are known.

Once \(\vec{\Omega}\) is known, we can determine the derivatives of the mode amplitudes, \(dA_{\mu}/dt:\)
\[
\sum_A m_A \left[ R^{-1} \cdot \left( \frac{d\vec{x}_A}{dt} - \frac{d\vec{x}_{CM}}{dt} \right) \right] \cdot \vec{e}_A^\mu = \sum_A m_A \left[ R^{-1} \cdot \left( R \cdot \frac{d\delta x_A}{dt} \right. \right.
\]
\[+ \vec{\Omega} \times \Delta \vec{X}_A \left. \right) \right] \cdot \vec{e}_A^\mu
\]
\[= \sum_A m_A \frac{d\delta x_A}{dt} \cdot \vec{e}_A^\mu + \sum_A m_A [R^{-1} \cdot (\vec{\Omega} \times \Delta \vec{X}_A)] \cdot \vec{e}_A^\mu.
\] (E.13)

Given \(d\delta \vec{x}_A/dt = \sum_A (dA_{\mu}/dt)e_A^\mu,\) we can use the derivative analog to Eq. (C.14) to simplify the first term on the right:
\[
\sum_A m_A \left[ R^{-1} \cdot \left( \frac{d\vec{x}_A}{dt} - \frac{d\vec{x}_{CM}}{dt} \right) \right] \cdot \vec{e}_A^\mu = M \frac{dA_{\mu}}{dt} + \sum_A m_A [R^{-1} \cdot (\vec{\Omega} \times \Delta \vec{X}_A)] \cdot \vec{e}_A^\mu.
\] (E.14)

This yields the equation for \(dA_{\mu}/dt:\)
\[
\frac{dA_{\mu}}{dt} = \sum_A \frac{m_A}{M} R^{-1} \cdot \left[ \left( \frac{d\vec{x}_A}{dt} - \frac{d\vec{x}_{CM}}{dt} \right) - (\vec{\Omega} \times \Delta \vec{X}_A) \right] \cdot \vec{e}_A^\mu.
\] (E.15)

To solve for \(\vec{\Omega},\) Eq. (E.12) produces a linear system with unknown \(\vec{\Omega} := (\vec{\Omega}(1), \vec{\Omega}(2), \vec{\Omega}(3))^T.\) We can write the right-hand side of Eq. (E.12) as a matrix \(P_A\) comprised of known variables \(\Delta \vec{X}_A := (\Delta \vec{X}_A(1), \Delta \vec{X}_A(2), \Delta \vec{X}_A(3))^T\) and \(\vec{x}_{0A} := (\vec{x}_{0A}(1), \vec{x}_{0A}(2), \vec{x}_{0A}(3))^T\) times \(\vec{\Omega}.\) Eq. (E.12) then becomes:
\[
\sum_A m_A \left( \frac{d\vec{x}_A}{dt} - \frac{d\vec{x}_{CM}}{dt} \right) \times \vec{x}_{0A} = \sum_A m_A P_A \cdot \vec{\Omega},
\] (E.16)
where $P_A$ is the matrix determined by the following entries:

$$P_A(i, i) = -\bar{x}_{0A}(j)\Delta \bar{X}_{A}(j) - \bar{x}_{0A}(k)\Delta \bar{X}_{A}(k), \quad (E.17)$$

$$P_A(i, j), i \neq j = \bar{x}_{0A}(j)\Delta \bar{X}_{A}(i). \quad (E.18)$$

Letting $P := \sum_A m_A P_A$ and $\bar{b}$ denote the left-hand side of Eq. (E.12), we can solve for $\bar{\Omega}$ with the following equation

$$\bar{\Omega} = P^{-1} \cdot \bar{b}. \quad (E.19)$$

Appendix E is a version of the unpublished notes by Lee Lindblom and the thesis author [38].
Bibliography


[38] Lee Lindblom and Francesca Grogan. Notes on Molecular Dynamics. 2016.


