A Stochastic Finite Element Model for the Dynamics of Globular Proteins

Robin Carl Oliver

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

December 2013

The University of Leeds

Department of Physics and Astronomy

Intellectual Property and Publication Statements

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly authored publications that have been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Chapters 2, 3 and 4 are based on the jointly authored paper: A stochastic finite element model for the dynamics of globular macromolecules[1]. All the work in these chapters and in the cited paper are directly attributable to the candidate and was completed under the supervision of Sarah Harris, Daniel Read and Oliver Harlen.

Acknowledgments

I would like to extend my thanks and appreciation to Dr Sarah Harris, Dr Daniel Read and Dr Oliver Harlen for giving me the opportunity to perform this work and making the the process so enjoyable. Their supervision, help and motivation have been invaluable in developing the theoretical model in this thesis and applying it to new systems. All their individual skills and expertise have been required to make this project a success from Daniel's expertise in thermal noise, Oliver's in finite element analysis and Sarah's in computational physics. I have thoroughly enjoyed their company during the course of my PhD and am saddened at the prospect of having to leave to work elsewhere at the completion of this project.

My collaborators at the University of Leeds have also proved invaluable, especially Dr Stan Burgess. I thank Stan for allowing me access to his experimental data on the molecular motor dynein that has proved vital to the work of this thesis and for being such an interesting physical and biological problem. This system proved to be almost an ideal test case for the model in this thesis and yielded quite interesting results and showed the applicability of the model to very large biological systems.

I had the great privilege to work at the Hauptman Woodward Medical Institute in Buffalo New York with Edward Snell, Joseph Luft, Thomas Grant. The time spent at the Synchrotron facility at Stanford was an eye opening experience for a theoretical physicist made especially enjoyable by the company. The data that was provided from this trip again provided a test case for the model in this thesis and I would especially like to thank Thomas for producing a Small Angle X-Ray Scattering envelope for me to work on.

This work was made possible by the funding provided by the EPSRC as well as the Collaborative Computational Project for Biomolecular Simulation (CCPBioSim) that provided the funding to go Stanford and Buffalo. I thank these bodies deeply for funding and Dr Charles Laughton for selecting my application for funding and the support provided.

Abstract

This thesis concerns a novel coarse graining method for the simulation of the globular proteins. Conventional simulation methods such as molecular dynamics cannot generate sufficient times to reach the important timescales that govern the biology of large biological systems like molecular motors. To address this, I have developed a new coarse graining algorithm drawing on the techniques of continuum mechanics and finite element analysis to build a new simulation technique. The novel part of this algorithm is that the fluctuation dissipation relation for the system can be derived and solved locally. This avoids the need to invert a global resistance matrix to solve for the thermal noise component of the system and reduces the computational expense of the algorithm per time step. I have validated this coarse grained model by performing a variety of tests on the numerical code including spatial and temporal convergence tests using Fourier analysis and beam bending theory. In addition compliance with the equipartition theorem has been confirmed.

One key advantage of this method over atomistic techniques is that the coarse grained method does not require any atomic information ab intio. Thus, this method can interface with low resolution imaging techniques such as Small Angle X-Ray Scattering and Cryo-Electron Microscopy. In this thesis, I show how to construct a finite element mesh from both of these sources and run simulations to replicate the results from Small Angle X-Ray Scattering and Cryo-Electron Microscopy experiments. In more detail, I have taken a structure obtained using Small Angle X-ray Scattering, ran simulations and checked that the dynamics do not affect the average X-Ray scattering curve. Furthermore, using experimentally obtained structures and dynamics of the molecular motor dynein I have run simulations to find the elastic parameters that match the experimental data to map the overall dynamics of the dynein motor.

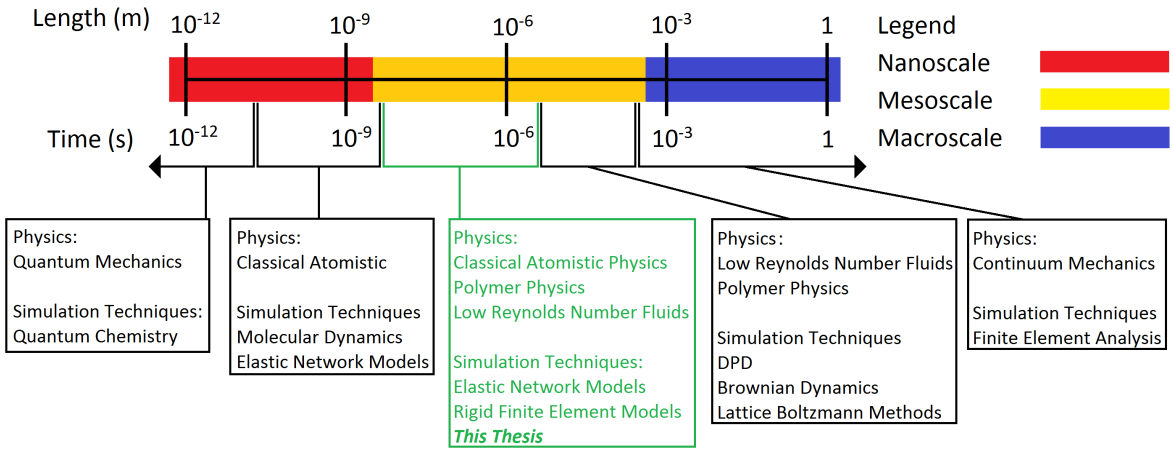
Contents

Chapter 1

Introduction

In this chapter, I will introduce the state of the art of biological simulations across a variety of length and time scales. In doing so, I highlight a length and time scale gap in the current state of the art. There are well established simulation techniques for small biomolecules of sizes up to 10Å that can reach biologically relevant time scales, of order $1ms$, such as molecular dynamics (MD)[2]. At long length and time scales, where thermal fluctuations are negligible, there are macroscopic simulation techniques such as finite volume and finite element methods[3]. However, there are relatively few techniques that are capable of simulating the thermal fluctuations of biomolecules with sizes greater than $10nm$ such as molecular motors and large protein assemblies like the ribosome[4] for long periods of time on the scale of $1\mu s$[5]. The purpose of this thesis is to address this mesoscale simulation gap by employing a new simulation strategy that extends traditional continuum mechanics and finite element analysis down into the mesoscale. I will present a brief summary of the important physics at different length scales and a summary of the relevant simulation techniques in Section 1.1 before discussing the Langevin equation in Section 1.2 and finite element analysis in Section 1.3.

Figure 1.1: The current state of the art in biological simulation showing the length and time scale gap in green.



1.1 The Physics of Biology

Biology exploits different length and time scales as well as different physics ranging from quantum mechanics[6], thermodynamics[7] and fluid mechanics[8] to achieve functionality. In order to explain these phenomena different computational techniques have been developed. In this section, I will discuss the different physics and simulation methods used in the nanoscale in Sections (1.1.1) and (1.1.2), the macroscale in Sections (1.1.3) and (1.1.4) and the mesoscale in Sections (1.1.5) and (1.1.6). In order, to try to understand these length and time scales, Figure 1.1 approximately gives the appropriate ranges of different regimes of physics and the computational techniques used to study them.

1.1.1 The Nanoscale

The nanoscale loosely refers to systems with typical length scales between an angstrom and a nanometer or with time scales less than a nanosecond. In this regime, the properties of individual atoms are important and can be described either by quantum mechanics[9] or classical atomistic physics[10]. This regime contains a variety of physics vital to life including enzyme catalysis[11], molecular recognition[12] and pro-

tein folding[13]. Of these systems, enzyme catalysis is a truly quantum problem as it involves the transfer of electrons whereas the remaining problems can be tackled using classical physics.

1.1.2 Nanoscale Simulation Techniques

In order to understand problems in the nanoscale, a variety of simulation techniques have been developed. These include Molecular Dynamics (MD)[14] and Quantum Chemistry[15]. However, these methods are both computationally expensive and it is often impractical to simulate large macromolecules[16]. As a consequence, coarse grained methods have been developed such as Go-Models[17] and Elastic Network Models[18] to extend the range of lengths and times accessible to simulation.

Quantum Chemistry

Quantum chemical methods look at electronic structure within small fragments of biomolecules or crystal structures. These techniques look at the smallest length and time scales in biophysics. Quantum mechanics in biology has been shown to be relevant in the study of photosynthesis[19] and stacking interactions in DNA[20]. There are a variety of software that apply the techniques of density functional theory[21] to solve these sort of problems such as Gaussian[15]. These computational techniques are also used to parametrise molecular dynamics forcefields[22].

Molecular Dynamics

Molecular Dynamics is a classical technique that resolves the forces on every atom in a system. The forces on each atom are calculated through the use of a forcefield that includes forces related to the bond angles and lengths between specific atoms, coulombic interactions between charged particles and the Lennard-Jones potential[23]. The solvent that a protein is immersed in can be treated explicitly using the TIP3P[24] water model that includes both electrostatic interactions and the Lennard-Jones potential or

implicitly where the external water is treated as a continuum[25].

While molecular dynamics provides a detailed view of biological matter, this technique is computationally expensive for large molecules[16]. This is a consequence of the range of time scales that molecular dynamics resolves from the bond frequency of hydrogen atoms with a period $1fs$[5] to protein folding that can be on the order of $1ms$[26]. As a consequence the time step for a molecular dynamics simulation has to be of order $1fs$. Molecular dynamics is also computationally expensive because the interactions between all possible pairings of atoms must be included. Thus, even with modern supercomputers it is not possible to simulate the longer biologically relevant time scales. An additional restriction is that MD requires the entire atomic structure of the molecule to be known.

Molecular dynamics has been used to study protein folding, DNA topology[27], drug design[28] and membrane proteins[29]. Today, molecular dynamics is a widely used as shown by the number of different molecular dynamics software packages such as Gromacs[30], Amber[31] and NAMD[32]. There are also a number of coarse grained molecular dynamics methods that I will discuss next.

Coarse Grained Molecular Dynamics

To extend the accessible length and time scales accessible to MD a variety of coarse grained methods have been developed. One approach is to coarse grain every individual atom in the system into a group of atoms to form virtual particles and resolve the forces between the virtual particles[33]. This method of coarse graining is exemplified by the Martini forcefield[33] for molecular dynamics. This reduces the total size of the computational loop and thus extends the system sizes that one can probe with molecular dynamics. This technique has been applied widely applied to lipid membranes[34].

Alternative schemes for coarse graining molecular dynamics also include manipulating the forcefield by placing a guiding potential into the forcefield. For example, proteins will have a native state and then a variety of different folded states. A protein

will take a long period of time to go from one folded state to another often well beyond the time scale accessible to conventional molecular dynamics. However, if one places an attractive potential that guides the dynamics in a particular direction then this reduces the necessary simulation time. This method is known as Go-Model[35] and has been widely used to study protein folding using fully atomistic molecular dynamics[36].

Elastic Network Models

Elastic network models[37] are divided into two broad categories Anisotropic Network Models (ANM) and Gaussian Network Models (GNM). Both methods utilise a series of nodes connected by springs, the difference being between the two methods being the potential chosen to represent the spring. The nodes are selected to be positioned at key residues within the structure of a protein such as the location of carbon atoms. In the ANM all nodes within a certain distance of one another are connected together by Hookean springs but with potentially different spring constants. In the GNM the nodes are positioned in the same manner as an ANM but with the spring potential that may be a function of direction as well as distance between the nodes.

In order to generate the overall motion of either an ANM or a GNM normal mode analysis can be performed in order to extract the bulk motions of network. Thus, elastic network models can eliminate the time scale problems associated with MD and obtain the longest time scale motions of a biomolecule without the need for extensive simulation. However, by removing all the physics except for the elastic potentials it is impossible for an elastic network to either simulate effects due to hydrophobicity, electrostatic interactions or multiple body systems.

Currently elastic network models have been applied to a wide variety of proteins and there are many online servers that will now predict and visualise the the motions of molecules within the Protein Data Bank (PDB)[38] on request[39] [40]. Elastic network models have also been used to steer molecular dynamics simulations[37].

1.1.3 The Macroscale

The macroscale refers to systems with length scales greater than a millimetre and time scales greater than a millisecond. The dominant physics in this regime is given by continuum mechanics as there are sufficiently many atoms that atomic detail can be ignored. The physics of the macroscale governs how entire organs and tissues work such as the heart pumping blood around the body.

1.1.4 Macroscale Simulation Methods

In the macroscale, materials are modelled using continuum models that average over the molecular detail and represent the solution in terms of continuously differentiable fields. The governing physics is incorporated into a series of partial differential equations that describe the evolution of the system called the continuum equations. There are a range of well established numerical algorithms for solving these systems, including finite difference, finite volume and finite element methods.

Finite Difference

The finite difference method discretises the domain of a continuum equation into a series of nodes and uses finite difference between values on these nodes to approximate derivatives. This method is easy to implement and typically uses a structured grid across the domain of the continuum equation[41]. A major disadvantage of this solution method is that the distances between nodes must be much smaller than the smallest length scale of the system being studied. When the continuum equation has a complex domain that evolves with time, as is the case with blood vessels or the human heart, or sharp changes in the fluid flow[42] the finite difference method can become computationally expensive as a new grid might have to be found to maintain the accuracy of the solution.

Finite Volume

The finite volume method[43] is related to the finite difference method in that the do-

main of the continuum equation is resolved on a grid. However, in the finite volume method the grid is not required to be structured. Around each node in the grid, a volume is defined and the continuum equation converted into a surface integral across the surfaces that connect the volumes in the grid. The solution can then be time stepped by computing the flux across each surface and updating the solution of the continuum equation on each node. The advantage of this method is that the solution is forced to conserve material flowing between elements and thus replicates fluid dynamics. Thus, this method can model fluid flow very efficiently in complicated geometries. This method has been applied to biological flows such as blood flow[44].

Finite Elements

The finite element method will be discussed in more detail in Section 1.3 as it forms the basis for the numerical algorithm presented in this thesis. At the macroscale, the finite element method has been applied to the study of blood flow in the human heart[3] and muscles[45].

1.1.5 The Mesoscale

For the purpose of this thesis, we shall consider the mesoscale to involve time scales greater than a nanosecond and less than a microsecond as well as length scales greater than a nanometer but less than a micrometer. The physics encountered in this region is diverse, ranging from low Reynolds number fluids[46], polymer physics[47], thermal physics[48] and atomistic physics[49]. The problem of the mesoscale is that in this regime, there is not one singular piece of physics that is important, this regime is driven by the connections between different realms of physics. For this reason, we will divide the mesoscale into two sections, upper and lower, as shown in Figure 1.1. The difference between these realms is length scale and its relation to the importance of thermal noise. The lower mesoscale includes systems such as cytoplasmic crowding[50], molecular motors and refers to systems with lengths on the order of nanometers, in

this regime thermal noise is important. The upper mesoscale refers to systems with lengths on the order one micrometer where thermal noise is not important describing such biological systems as individual cells and micro swimmers[51] like sperm. In this thesis, we are concerned primarily with the lower mesoscale and systems where thermal noise is important.

1.1.6 Mesoscale Simulation Techniques

One approach to simulation in the mesoscale is to extend the range of the continuum methods encountered in the macroscale into the mesoscale. In order to go to smaller length scales it is necessary to incorporate thermal noise into the model. Thermal noise can be incorporated into the continuum equations[52]. This replaces the system of continuum equations with a system of stochastic partial differential equations. This technique has been used to study the dynamics of particle suspensions[53], deformation of vesicles dragged by molecular motors[54] and the effect of shape on adhesion by including the thermal fluctuations as well as the mean flow in the background fluid[55].

An alternative to solutions of the continuum equations (using finite element or finite volume schemes) are methods that use simulations of systems of virtual particles to reproduce the continuum equations. Such methods include lattice Boltzmann methods[56], and lattice free methods such as dissipative particle dynamics[57], where the fluid motion is obtained from a simulation of collisions between soft particles. Even less sophisticated methods are available in Brownian Dynamics[58] where hydrodynamics is ignored entirely.

In these simulations the particles do not represent the underlying microstructure, but provide a means for transporting momentum. They are particularly suited to studies of suspensions where the suspended objects can be formed by joining together lattice sites of Dissipative Particle Dynamics (DPD) particles. This has led to their application to the simulation of blood flow in micro channels[59].

Lattice Boltzmann Method

The lattice Boltzmann method[56] is derived by consideration of the Boltzmann equation. The method uses a lattice wherein for each node of the lattice a velocity probability distribution function is defined. In each time step of the simulation, the probability distribution function on each node is updated using a collision operator that describes how the probability distributions of various nodes interact with one another. The collision operator conserves momentum and energy between time steps. Thus, at long length scales many times the distance between nodes on the lattice hydrodynamics is recovered. This technique has been applied to deformable particle suspensions such as blood cells, turbulence and microfluidics[56].

Dissipative Particle Dynamics

DPD[57] is a variant of the Brownian Dynamics method where momentum is conserved locally. In general, there is a lot of similarity between Brownian Dynamics and DPD. They are both particle based methods that might be connected by some conservative potential but the difference comes in the way the viscosity is calculated and the corresponding random force. The force on the i^{th} particle in dissipative particle dynamics is:

$$\mathbf{f_i} = \sum_{\mathbf{i \neq j}} (\mathbf{F^D_{ij}} + \mathbf{F^C_{ij}} + \mathbf{F^T_{ij}}). \tag{1.1}$$

where $\mathbf{F^D_{ij}}$ is the dissipative force, $\mathbf{F^C_{ij}}$ is the conservative force and $\mathbf{F^T_{ij}}$ is the thermal force.

What makes DPD unique is that all the forces $\mathbf{F^D_{ij}}$, $\mathbf{F^C_{ij}}$ and $\mathbf{F^T_{ij}}$ are all antisymmetric under permutation of i and j. Thus, the total force between two particles in the DPD system is always equal and opposite. As the forces are antisymmetric, momentum is conserved locally. To demonstrate this, we will consider the viscous force $\mathbf{F^D_{ij}}$ given by:

$$F^D_{ij} = -\gamma w(\hat{\mathbf{r}}_{ij} \cdot \mathbf{v_{ij}})\hat{\mathbf{r}}_{\mathbf{ij}}. \tag{1.2}$$

Equation (1.2) is antisymmetric under permutation of i and j because:

$$\mathbf{v_{ij}} = \mathbf{v_i} - \mathbf{v_j}. \tag{1.3}$$

Conservation of momentum locally is important because this gives rise to fluid mechanics without the need to solve the continuum fluid equations. Therefore, DPD does include hydrodynamic interactions, unlike Brownian Dynamics. DPD has been applied to red blood cells[60], polymers[61], surfactants[62] and suspensions of DNA under flow[63].

Brownian Dynamics

Brownian dynamics[58][64] is a simulation method wherein a series of point particles are subject to the Langevin equation (discussed in more detail in Section 1.2) plus an interaction potential such that the equation of motion for each particle is given by:

$$m\frac{d^2x}{dt^2} = -\lambda\frac{dx}{dt} + E(t) + f(t). \tag{1.4}$$

Where $E(t)$ is derived from a conservative interaction potential, a typical example being the Lennard Jones potential, $f(t)$ is a stochastic force vector with statistics chosen to conserve classical thermodynamics and λ is a drag coefficient.

The viscosity present in Brownian dynamics comes from the particle dragging against a viscous background medium, not from particle interaction. Thus, the momentum in a Brownian dynamics simulation is not conserved. The consequence of this is that there is no hydrodynamics in a Brownian dynamics model, the Brownian particles merely interact with one another through the conservative potential $E(t)$.

Brownian dynamics is ideal for the simulation of colloids wherein the important effect is the interaction between the colloids that might give rise to a phase transition or aggregation of the colloids[65]. More recently, McGuffee and Elcock[66] have applied Brownian dynamics to model cytoplasmic crowding. These simulations were performed

by using the crystal structures of proteins found in the bacterial cytoplasm and modelling them as rigid Brownian particles. To date this is the most accurate simulation ever produced of the cytoplasm.

1.1.7 Length Time Scale Gap

From this overview, in the mesoscale we can see that there are techniques that can tackle problems such as small scale blood flow problems or deformation of blood cells at the upper end of the mesoscale. Similarly, at the lower end of the mesoscale there are techniques such as elastic network models that can reveal the normal modes of large macromolecules. However, there are very few techniques that can tackle problems in the middle of the mesoscale where the workings of large proteins and cellular machinery such as organelles and molecular motors are important. This thesis addresses this issue by building a technique that can model the thermal fluctuations of large biomolecules for biologically relevant time scales by extending the functional range of the finite element method into the mesoscale. The closest competing method is that of the Immersed Finite Element Method[67] where thermal fluctuations are input into the model through a background fluid, but this has been used to model only rigid nano particles under the influence of flow[55] [68]. In the next two sections (1.2 and 1.3), I will develop the necessary background material for understanding this thesis by discussing the Langevin equation (Section 1.2) and the finite element method in (Section 1.3).

1.2 The Langevin Equation and Brownian Motion

Brownian motion is the random motion that microscopic particles undergo due to thermal collisions. The trajectory of a particle undergoing Brownian motion is described by a Langevin equation. This equation incorporates the physics underlying thermodynamic principles such as the Stokes-Einstein relation and the fluctuation dissipation relation (Section 1.2.1 and 1.2.2). These principles connect the properties of the thermal

noise in a non trivial manner to the dissipation of the system. This forms the theoretical basis for the statistical physics of the finite element model discussed in chapter 2. Therefore, it is prudent to discuss the physics of the Langevin equation as an introduction before moving onto the more complicated discussion. Thus, in this section I discuss the Langevin equation (Section 1.2.1), the derivation of fluctuation dissipation relation (Section 1.2.2) and the Einstein relation in (Section 1.2.3).

1.2.1 The Langevin Equation

The simplest form of Langevin equation[47] is for an isolated particle subject to thermal noise and viscous drag due to a background medium. Thus, by using Newton's second law the one dimensional Langevin equation is given by:

$$m\frac{d^2x}{dt^2} = -\lambda\frac{dx}{dt} + \nu(t), \tag{1.5}$$

where λ is a friction coefficient and ν is a stochastic force.

In this model of an isolated Brownian particle, hydrodynamic interactions are not considered, and for a spherical particle of radius R the friction coefficient is given by Stokes' law[69]:

$$\lambda = 6\pi\mu R, \tag{1.6}$$

where μ is the fluid viscosity.

The random force vector $\nu(t)$ applies random kicks to the particle. The statistics of the random force vector $\nu(t)$ must be selected so that the system equilibrates correctly. Specifically, every quadratic degree of freedom the particle possesses must possess an average kinetic energy of $\frac{k_BT}{2}$. This determines the statistics of $\nu(t)$. In one dimension, a Langevin particle cannot distinguish between left and right because the particle is equally likely to be hit from the left or the right by a thermal force, thus:

$$\langle \nu(t) \rangle = 0 \tag{1.7}$$

where $\langle ... \rangle$ denotes the ensemble average of a large number of realisations.

The second moment $\langle \nu(t)\nu(t') \rangle$ is non-zero and it will be shown in the next section that:

$$\langle \nu(t)\nu(t') \rangle = 2\lambda k_B T \delta(t - t'), \tag{1.8}$$

where $\delta(t)$ is the Dirac delta function.

Equation (1.8) is the fluctuation dissipation relation[70]. The fluctuation dissipation relation introduces the energy scale $k_B T$ and couples the thermal noise explicitly to the viscosity. In other words, the more viscous the medium, the stronger the kicks from the random force vector $\nu(t)$. One way in which we can interpret equation (1.8) is through energy balance. At equilibrium, the average energy in each degree of freedom will be $\frac{k_B T}{2}$, and therefore the effect of the thermal fluctuations is to restore the kinetic energy lost through viscosity.

1.2.2 Derivation of the Fluctuation Dissipation Relation

In chapter 2, we will be required to derive the fluctuation dissipation relation for a more complicated system of equations. In order to show how one can approach the derivation and the key principles to consider I will derive the fluctuation dissipation relation for equation (1.5). To begin, we shall consider the kinetic energy of a Langevin particle:

$$\mathcal{E} = \frac{mv^2}{2}. \tag{1.9}$$

Where $v = \frac{dx}{dt}$ is the particle velocity.

At thermal equilibrium, because the kinetic energy is a quadratic degree of freedom of the system, the time average of the kinetic energy must be:

$$\langle \mathcal{E} \rangle = \frac{k_B T}{2}. \tag{1.10}$$

To begin the derivation of the fluctuation dissipation relation, we shall define Q such that:

$$Q = \langle v(t)v(t) \rangle, \tag{1.11}$$

Then, from equation (1.10), it follows that:

$$Q = \frac{k_B T}{m}. \tag{1.12}$$

The derivation proceeds by considering fluctuations of the energy variable Q in a small time step Δt. The change in Q is given by:

$$\Delta Q = \langle v(t)\Delta v(t) + \Delta v(t)v(t) + \Delta v(t)\Delta v(t) \rangle = 0, \tag{1.13}$$

where $\Delta v = v(t + \Delta t) - v(t)$.

In this derivation, we assume that the noise is delta correlated such that $\langle \nu(t)\nu(t') \rangle = 0$. Such an assumption is reasonable because the collision times are short. More rigorously, the dissipation term is frequency-independent[70]. So there can be no frequency-dependence of the noise. This corresponds to white noise that is equivalent to a delta correlation in time.

The terms $v(t)\Delta v(t)$ and $\Delta v(t)\Delta v(t)$ can be evaluated by integrating equation (1.5) with respect to a finite time step Δt such that:

$$v(t)\Delta v(t) = \frac{\Delta t}{m}(-\lambda v(t)v(t) + v(t)\nu(t)), \tag{1.14}$$

and,

$$\Delta v(t)\Delta v(t) = \frac{\Delta t^2}{m^2}(\lambda^2 v(t)v(t) - \lambda v(t)\nu(t) - \lambda v(t)\nu(t) + \nu(t)\nu(t)). \tag{1.15}$$

We must now consider the effect of averaging equations (1.14), and (1.15) to order Δt on the understanding we will eventually take the limit Δt tending to 0. The thermal forces are delta correlated and so:

$$\langle v(t)\nu(t)\rangle = \langle v(t)\rangle\langle \nu(t)\rangle, \tag{1.16}$$

but since $\langle v(t)\rangle = 0$ from equation (1.15). So, equation (1.16) is simplified to:

$$\langle v(t)\nu(t)\rangle = 0. \tag{1.17}$$

Equations (1.14), and (1.15) are then reduced to:

$$\langle v(t)\Delta v(t)\rangle = \frac{-\lambda\Delta t}{m}\langle v(t)v(t)\rangle, \tag{1.18}$$

and,

$$\langle \Delta v(t)\Delta v(t)\rangle = \frac{\Delta t^2}{m^2}\langle \nu(t)\nu(t)\rangle. \tag{1.19}$$

Equation (1.19) holds because because the average of the first three terms of equation (1.15) are of order Δt^2 while the average of the noise correlation term is of order Δt. The reason for this is that the noise is delta correlated in time so that $\langle \nu(t)\nu(t')\rangle = \delta(t-t')$. In the discrete time framework, used throughout this derivation, this becomes $\langle \nu(t)\nu(t')\rangle = \frac{\delta_{tt'}}{\Delta t}$, where $\delta_{tt'}$ is the Kronecker delta function. Consequently, equation (1.19) is true to order Δt and in the limit Δt tending to 0 only the noise correlation term need be retained. We can then safely drop the second order terms in Δt as negligible.

We can now substitute equations (1.18) and (1.19) into (1.13) while using the delta time correlation property of the thermal noise to yield:

$$\langle \nu(t)\nu(t) \rangle = \frac{2\lambda k_B T}{\Delta t}. \tag{1.20}$$

We now reintroduce the delta time correlation into equation (1.20) so that:

$$\langle \nu(t)\nu(t') \rangle = \frac{2\lambda k_B T \delta_{tt'}}{\Delta t}. \tag{1.21}$$

We can now take the the limit of Δt tending to 0 such that:

$$\langle \nu(t)\nu(t') \rangle = 2\lambda k_B T \delta(t - t'). \tag{1.22}$$

This completes the derivation of the fluctuation dissipation relation[70]. We will return to this style of proof in chapter 2 in order to derive the fluctuation dissipation relation for the finite element model of a globular macromolecule.

1.2.3 Stokes Einstein Relation

Along with the fluctuation dissipation relation, the Stokes Einstein relation is the other important statistical result that arises from the Langevin equation. The Stokes Einstein relation was first reported in Einstein's 1905 paper[71] on Brownian motion and then independently discovered by both Smoluchowski[72] and by Sutherland[73] in the same year. The Stokes Einstein relation is as follows:

$$\langle \Delta x^2 \rangle = \frac{2k_B T}{\lambda} t, \tag{1.23}$$

where Δx is defined as $x = \Delta x + x(0)$ for an initial particle position $x(0)$ and t much longer than the collision time scale.

Equation (1.23) describes the diffusion of a particle in one dimension in this formulation. This result states that the diffusion of a Langevin particle is coupled to the drag coefficient λ and grows linearly in time. Thus, for a particle that is being hit randomly by forces that have statistical correlations described by the fluctuation dissipation re-

lation, there is a simple result that describes the average position of the particle due to the time evolution if the random force $\nu(t)$.

We will now turn our attention to the derivation of the Stokes Einstein relation using Langevin's approach[74]. This derivation begins by re-arranging the Langevin equation (1.5) as a function of $\frac{d(x^2)}{dt}$:

$$\frac{\lambda}{2}\frac{d(x^2)}{dt} = x\nu(t) - mx\frac{d^2x}{dt^2}. \tag{1.24}$$

The last term in equation (1.24) can be simplified by using the following relation:

$$x\frac{d^2x}{dt^2} = \frac{d}{dt}\left(x\frac{dx}{dt}\right) - \left(\frac{dx}{dt}\right)^2. \tag{1.25}$$

Thus, equation (1.24) can be simplified by substitution such that:

$$\frac{\lambda}{2}\frac{dx^2}{dt} = x\nu(t) - m\frac{d}{dt}\left(x\frac{dx}{dt}\right) + m\left(\frac{dx}{dt}\right)^2. \tag{1.26}$$

The final step is to take an ensemble average of equation (1.26). Since the position of the Langevin particle at time t is uncorrelated with both its velocity and the random force vector, it follows that both:

$$\langle x\nu\rangle = 0, \text{and} \tag{1.27}$$

$$\langle x\frac{dx}{dt}\rangle = 0. \tag{1.28}$$

We can then apply the equipartition theorem on the final term in equation (1.26) to result in the time derivative of the Stokes Einstein relation:

$$\frac{d\langle x^2\rangle}{dt} = \frac{2k_BT}{\lambda}. \tag{1.29}$$

The right hand side of equation (1.29) is a constant and thus the time integral is trivial so that:

$$\langle x^2 \rangle = \frac{2k_B T}{\lambda} t + \langle x^2(0) \rangle, \tag{1.30}$$

where we have also implemented the boundary condition at $t = 0$, $x(t) = x(0)$.

Equation (1.30) can then be simplified by considering the average of $x^2 = (\Delta x + x(0))^2$ given by:

$$\langle \Delta x^2 \rangle = \langle x^2 \rangle - \langle x(0)^2 \rangle. \tag{1.31}$$

Equation (1.31) is true because $\langle \Delta x \rangle = 0$ as there is no propensity for a one dimensional Brownian particle to prefer leftward or rightward drift. Hence we can now re-arrange equation (1.30) to yield:

$$\langle \Delta x^2 \rangle = \frac{2k_B T}{\lambda} t, \tag{1.32}$$

which is the Stokes Einstein relation.

The Stokes Einstein relation will be used in validating the implementation of the background drag in Chapter 2.

1.3 Finite Element Analysis

Finite element analysis is the solution scheme that I will use to solve the partial differential equation that describes the continuum model of a globular protein. In this section, I will introduce the mathematical concepts behind finite element analysis in order to demonstrate how the method works for a simple example system[75]. Finite element analysis has been widely used for structural mechanics and fluid mechanics and has more recently been applied to study macroscopic biological systems such as blood flow in the heart[3]. The key advantage of finite element analysis over finite difference methods is that finite element analysis is more easily able to handle complex geometries. The mathematical foundations of finite element analysis[75] are presented in Section

1.3.1, the Galerkin method[75][76] in Section 1.3.2, finite elements[75] in Section 1.3.3 and finally higher dimensions[75] in Section 1.3.3.

1.3.1 Weighted Integral Method and the Weak Formulation

Here we shall discuss the mathematical foundations of finite element analysis. In order to do this, I shall introduce a differential equation that we shall aim to solve using the finite element method[75]. From this, we shall see the general mechanism of the finite element approach[77] and how the finite element method builds an approximate solution to our partial differential equation. The example differential equation we seek a solution for is:

$$-\frac{d}{dx}\left(a\frac{du}{dx}\right) - cu + x^2 = 0. \tag{1.33}$$

Where the function u is defined between $0 < x < 1$ with the boundary conditions $u(0) = 0$ and $a\frac{du}{dx} = 1$ at $x = 1$. The finite element method works by finding an approximate solution of equation (1.33) within a restricted subspace of functions, typically piecewise polynomials. For any function u we define the residue function $R(u)$ as the result of applying the differential operator on the left hand side of equation (1.33) such that:

$$-\frac{d}{dx}\left(a\frac{du}{dx}\right) - cu + x^2 = R(u). \tag{1.34}$$

We must now define the space in which we will seek an approximate solution of (1.34). In general, we shall define the solution space as that spanned by a series of basis functions $\phi_1(x), ..., \phi_n(x)$. The functions $\phi_j(x)$ are chosen to satisfy the Dirichlet boundary conditions so that $\phi_j(0) = 0$. We define a function u of the form:

$$u = \sum_{i=1}^{n} c_i\phi_i, \tag{1.35}$$

where c_i are coefficients.

The next step is to find the function in the space defined by equation (1.35) that ensures the residue $R(u)$ is close to zero. Clearly, the residue $R(u)$ will not be 0 unless the space spanned by equation (1.35) includes the actual solution. Instead, we seek solutions for which the weighted integral:

$$\int_0^1 w(x)R(u)dx = 0. \tag{1.36}$$

Here $w(x)$ are some suitable set of functions that we choose. Equation (1.36) is an inner product so an interpretation of this process is that we seek to make the residue orthogonal to the weight functions $w(x)$. We now construct a weak form of equation (1.34) by substitution into equation (1.36) so that:

$$\int_0^1 \left(w\frac{d}{dx}\left(-a\frac{du}{dx}\right) - cuw + wx^2 \right) dx = 0. \tag{1.37}$$

Equation (1.37) can be simplified by integrating by parts and applying the boundary conditions to yield:

$$\int_0^1 \left(a\frac{dw}{dx}\frac{du}{dx} - cuw + wx^2 \right) dx - w(1) = 0. \tag{1.38}$$

Equation (1.38) is described as being the weak form of equation (1.33). Clearly, any solution of equation (1.33) must satisfy equation (1.38). Furthermore, it can be shown that if equation (1.38) is satisfied for all functions $w(x)$, then u satisfies equation (1.33). The proof is non-trivial and requires the use of Sobolev spaces and thus will be omitted[77] [76].

We must now choose a suitable solution space and corresponding choices for the weight functions $w(x)$. Since equation (1.37) involves first derivatives of both u and w, these functions should lie in the Sobolev space $H_0^1(0,1)$ (i.e. functions on the interval (0,1), whose first derivatives are square integrable on the interval (0,1)). Note that the original equation (1.33) requires second order derivatives of u to exist, but this is not required for solutions of the weak form. Furthermore if the solution space has dimension

n, then n independent weight functions are required to obtain a unique solution for the coefficients c_i.

1.3.2 The Galerkin Formulation

The Galerkin formulation[75][76] of finite element analysis provides a convenient means of choosing the n weight functions. We simply use the basis functions ϕ_i as the weight functions such that equation (1.36) becomes:

$$\int_0^1 \phi_i(x) R(u) dx = 0 \tag{1.39}$$

We can now substitute $u = \sum_{j=1}^{n} c_j \phi_j$ into equation (1.38) and set w to equal each of the basis functions ϕ_i in turn, to give:

$$\sum_{j=1}^{n} c_j \left(\int_0^1 a \frac{d\phi_i}{dx} \frac{d\phi_j}{dx} - c\phi_i \phi_j dx \right) = \phi_i(1) - \int_0^1 \phi_i x^2 dx. \tag{1.40}$$

Equation (1.40) is a set of n linear equations of the form:

$$A_{ij} c_j = f_i, \tag{1.41}$$

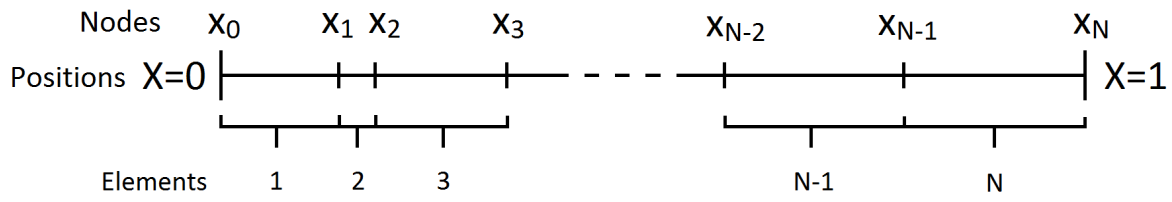
where A_{ij}, and f_i are defined as follows:

$$A_{ij} = \left(\int_0^1 a \frac{d\phi_i}{dx} \frac{d\phi_j}{dx} - c\phi_i \phi_j dx \right), \text{and} \tag{1.42}$$

$$f_i = \phi_i(1) - \int_0^1 \phi_i x^2 dx. \tag{1.43}$$

We have now reduced equation (1.33) to a set of linear equations that can be solved for the coefficients c_j.

Figure 1.2: Finite element discretisation of a 1D unit line.



1.3.3 Finite Elements

In the finite element method, the solution space is obtained by dividing the solution domain into elements called finite elements that are geometrically simple[75][78]. We then define the solution space from the union of simple functions ψ_i^e (typically polynomials) over each element so that:

$$\phi_i = \cup \psi_i^e. \tag{1.44}$$

In our simple one dimensional example, we divide the domain of the differential equation $0 < x < 1$ into sub domains such that each finite element describes an interval $[x_{i-1}, x_i]$, as shown in Figure 1.2. Since we require the functions to be in $H_0^1(1,0)$, we need the functions to be continuous between elements and so the simplest polynomial functions that can be used are linear functions as shown in Figure 1.3. Note that it is possible to use discontinuous functions, but this requires taking account of the discontinuities at element boundaries and is called the Discontinuous Galerkin method[79].

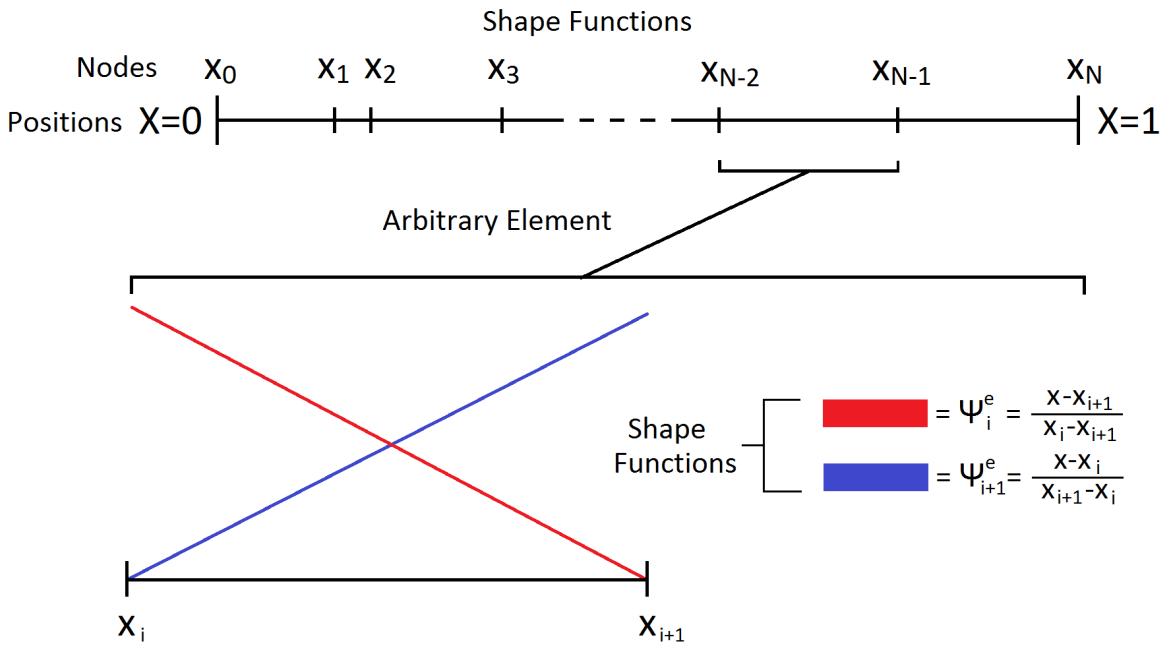
The basis of the solution space $\phi_i(x)$ can then, for example, be defined by piecewise linear functions $\phi_i(x_j)$ satisfying,

$$\phi_i(x_j) = \delta_{ij}. \tag{1.45}$$

ψ_i is zero except in the elements i and $i+1$ where it is given by:

$$\psi_i^e = \frac{x - x_{i-1}}{x_i - x_{i-1}}, \tag{1.46}$$

Figure 1.3: Shape functions for an element of a unit line.



where $x \in (x_{i-1}, x_i)$ and

$$\psi_i^e = \frac{x - x_{i+1}}{x_i - x_{i+1}}, \tag{1.47}$$

where $x \in (x_i, x_{i+1})$.

Thus, equation (1.40) becomes:

$$\sum_{j=1}^{n} c_j \left(\int_{x_{e-1}}^{x_e} a \frac{d\psi_i^e}{dx} \frac{d\psi_j^e}{dx} - c\psi_i^e \psi_j^e dx \right) = \psi_i^e(1) - \int_{x_{e-1}}^{x_e} \psi_i^e x^2 dx. \tag{1.48}$$

We then evaluate this integral over all elements and then sum the contribution from each element to produce the matrix A_{ij} and the vector f_i. We can then solve the linear system for the coefficients c_j that define the value of u on each of the nodes and thus complete the construction of the solution.

Thus, we have obtained the contributions from a general element to the matrix A_{ij} and vector f_i in terms of the shape functions ψ_i^e. To complete this problem one would evaluate these entities and then invert the matrix A_{ij} to solve the coefficients c_j. From

these coefficients the finite element solution to equation (1.33) is easy to calculate.

1.3.4 Higher dimensions

The detailed example that we have gone through up to now shows the mechanics of finite element analysis at its most basic. In general, the main advantages of this method are in higher dimensions[77] where the domain of our differential equation may be complicated, as in the case for a globular macromolecule. In chapter 2, we will be concerned with solving a 3 dimensional problem over a complex domain. In three dimensions, the volume of the solution domain is divided into polygons. Although other shapes can be used for 3D finite elements, the tetrahedron is the most versatile because it is generally straightforward to divide a volume into tetrahedra. A linear right angled tetrahedron is shown in Figure 1.4 with the nodes identified.

The space of linear functions on each element has four dimensions, therefore it is convenient to use four vertices as nodes with shape functions defined as:

$$\psi_1 = 1 - s - t - u, \tag{1.49}$$

$$\psi_2 = s, \tag{1.50}$$

$$\psi_3 = t, \text{and} \tag{1.51}$$

$$\psi_4 = u, \tag{1.52}$$

where s, t, and u are coordinates along the edges of the tetrahedron connected to node 1 with $0 \leq s, t, u \leq 1$. Linear elements are the simplest finite element for solving partial differential equations up to second order. However, such elements are only first order accurate in space, and so in order to provide a more accurate solution,

Figure 1.4: Right angled tetrahedron with sides of unit length.

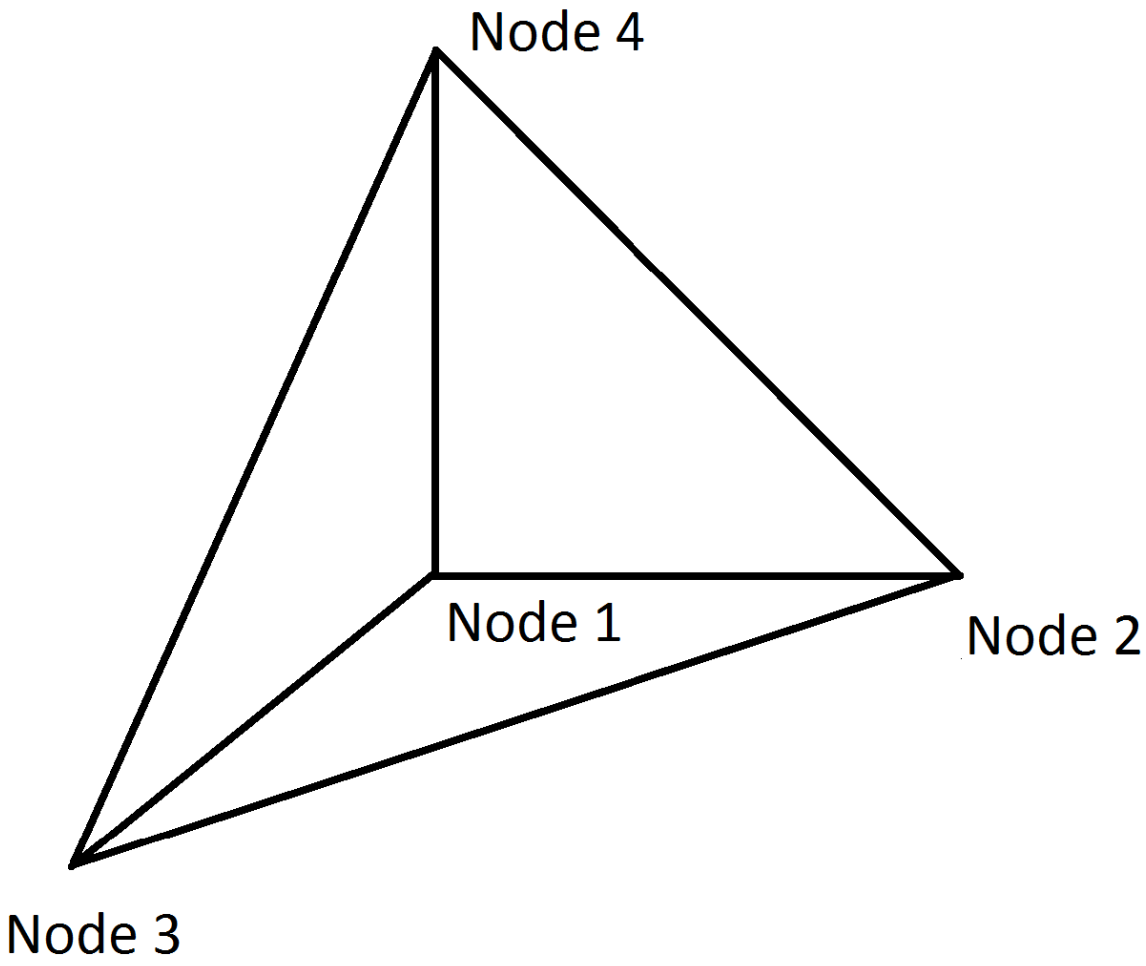
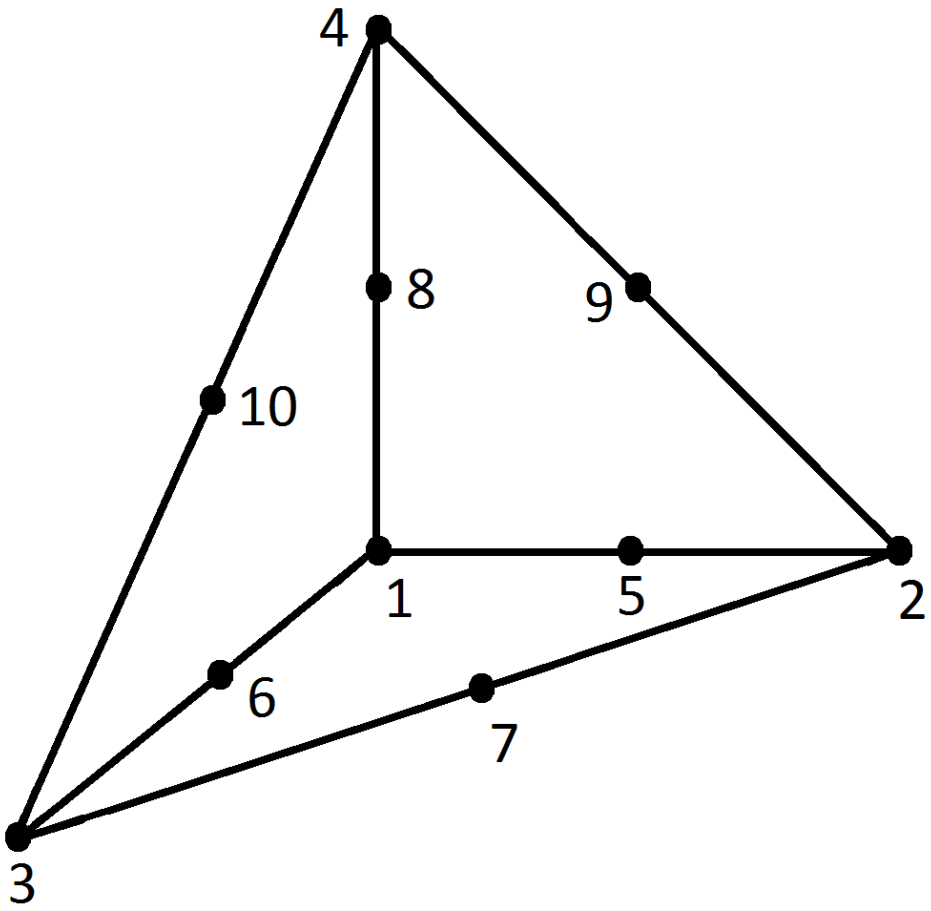


Figure 1.5: Second Order Tetrahedron with sides of unit length.



higher order polynomials are sometimes preferred. Quadratic interpolation requires ten shape functions[80] which can be accommodated by introducing additional nodes at the centres of each edge of the tetrahedron shown in Figure 1.5. Cubic and higher order interpolations can also be obtained in principle.

This completes our overview of finite element analysis; more details will be added in Chapter 2.

1.4 Conclusion

In this chapter, I have presented an overview of the current simulation methods for biomolecules. Currently, there is a lack of simulation capacity at the length and time scale that governs the functionality of the molecular machinery within our cells (Section 1.1). Consequently, there is a need to develop alternative techniques that are capable of exploring the biophysics within this regime. In order to develop such a technique, I will use the tools of statistical physics developed by exploring the Langevin equation (Section 1.2) and the Finite Element Method (Section 1.3). This simulation method will use continuum methods and not be based on the intrinsic atomistic structure of a particular biomolecule but instead the overall shape of the biomolecule. This will allow the method to interface with the wide array of low resolution structures available through Small Angle X-ray scattering of Cryo-EM. Currently, the EMDB[81] (Electron Microscopy Database) contains low resolution data of very large biological systems such as the axoneme[82]. A technique that could use this information to drive simulations would be a great asset and open up the lower mesoscale to simulation.

Chapter 2

Mathematical Background

2.1 The Continuum Model

In this thesis I will model a globular macromolecule as a continuous medium of density ρ subject to thermal noise, viscous dissipation and elasticity. I will focus first on describing the internal hydrodynamics, elasticity and thermal noise, with the effects of external solvent added to the problem in Section (2.4). The equation of motion connecting the velocity u_i to the stress σ_{ij} at all points in the material can be represented by continuum fields. Using index notation together with summation convention the equation of motion is then:

$$\rho\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right) = \frac{\partial \sigma_{ij}}{\partial x_j}, \tag{2.1}$$

where $\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right)$ is the total time derivative of the velocity vector field in the Lagrangian frame of the material. The stress σ_{ij} can be subdivided into three contributions:

$$\sigma_{ij} = \sigma^v_{ij} + \sigma^e_{ij} + \sigma^t_{ij} \tag{2.2}$$

σ^v_{ij}, σ^e_{ij} and σ^t_{ij} are the stresses due to non-conservative friction, conservative elastic forces and thermal fluctuations respectively. The material model for the macromolecule is therefore a Kelvin-Voigt[83] model in which the elastic stresses and viscous stresses act in parallel. In principle, different material models could be considered such as those with fading material memory, although this would not change the manner in which the derivations presented in this Section would proceed. A more complicated material model would only make the derivation more difficult due to the inclusion of new physics such as memory effects. The Kelvin-Voigt material model provides the simplest continuum model for which the thermal noise can be derived, as the stress in the system is dependent only upon the instantaneous deformation and the velocity field. The form of the three stress terms that are used are introduced in Section 2.1.1-2.1.3. The solution scheme employed and the nature of the finite element approximation is discussed in Section 2.2. In Section 2.3 additional details of the thermal noise term σ^t_{ij} are provided and demonstrate the compliance of the fluctuating finite element scheme with the fluctuation-dissipation theorem. Finally, in Section 2.4 an external solvent is added to the problem.

2.1.1 Viscous Stress

The material is assumed to have an linear viscous stress[84] σ^v_{ij}, which can be written as:

$$\sigma^v_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \lambda\frac{\partial u_m}{\partial x_m}\delta_{ij}, \tag{2.3}$$

where μ is the shear viscosity and λ is the second coefficient of viscosity, giving a bulk viscosity $\mu_{bulk} = \lambda + \frac{2}{3}\mu$.

2.1.2 Elastic Stress

In the model, we consider the simple case where material elasticity is hyperelastic, so that the elastic stress σ^e_{ij} can be derived from a strain energy density functional[85]. This is written in terms of the deformation gradient tensor F_{ij} (defined as $F_{ij} = \frac{\partial x_i}{\partial X_j}$ where $\mathbf{x}(\mathbf{X}, t)$) is the current position of material initially located at $\mathbf{X}$. Hence the local volume change is given by $\frac{V}{V_0} = det(F)$. We use a formulation which includes classical rubber elasticity and a volumetric spring that acts as a source of pressure. The strain energy density per unit current volume is written as:

$$\begin{aligned} W &= \frac{G}{2\det(F)}(F_{\alpha\beta}F_{\alpha\beta}) + \frac{B}{2\det(F)}\left(\det(F) - \alpha\right)^2 \\ &- \frac{3G}{2\det(F)} - \frac{B}{2\det(F)}\left(\frac{G}{B}\right)^2. \end{aligned} \tag{2.4}$$

Here G is the shear modulus, $K = B - \frac{G}{3}$ is the bulk modulus and α is a constant used to impose zero isotropic stress at zero deformation, requiring that $\alpha = 1 + \frac{G}{B}$. From the bulk and shear moduli, the Young's Modulus of the material is given by $E = \frac{9KG}{3K+G}$.

Equation (2.4) is effectively a second order expansion of the elastic energy in terms of the deformation gradient tensor F_{ij} about an elastic energy minimum. Thus, this expansion is only valid for small strains about this elastic energy minimum. For large strains additional terms in the expansion would have to be retained. Thus, the elasticity derived in this model cannot be used to model large conformational changes of a particular biomolecule as this would involve large strains. Instead, the elasticity used is only valid for the small scale thermal fluctuations experienced about an elastic energy miniumum or rest state of a protein. In general, proteins have many elastic energy minima and transitions between these minima cannot be modelled directly using this elastic energy expansion. However, one can model different minima independently as seen in Chapter 6.

The stress can be calculated by considering the the change in energy when a small

strain ϵ_{ij}, defined in terms of the deformation gradient tensor as $\widetilde{F}_{ij} = (\delta_{i\alpha} + \epsilon_{i\alpha})F_{\alpha j}$, is applied to a portion of material and taking the limit where $\epsilon_{ij} = 0$, giving the stress tensor:

$$\begin{aligned} \sigma^e_{ij} &= \frac{1}{\det(\widetilde{F})}\frac{\partial(W\det(\widetilde{F}))}{\partial\epsilon_{ij}} \\ &= \frac{G}{\det(F)}F_{ik}F^T{}_{kj} + B(\det(F) - \alpha)\delta_{ij}. \end{aligned} \tag{2.5}$$

We have assumed that the effect of the thermal noise on the elasticity of the material is small compared to the uncertainty in the known elastic moduli for biomaterials (see Section 4.2). In general, small length scale thermal fluctuations will affect the effective elasticity over larger length scales in the non-linear elastic regime. In principle, this effect should be accounted for when coarse-graining if the dimensions of the finite elements are significantly increased.

2.1.3 Thermal Stress

In particle based simulation, techniques such as Molecular Dynamics[86] or Brownian Dynamics[64] thermal fluctuations are included by adding a random force to each particle in the simulation. In the present method, thermal forces are introduced via a fluctuating stress tensor σ^t_{ij}. Unlike the elastic and viscous contributions, this thermal stress term is stochastic in both space and time, with statistics chosen to balance the viscous energy dissipation. The advantage of this approach is that within a finite element approximation the fluctuating stress can be calculated entirely locally for the viscous stress and still yield the correct thermal physics. In Section 2.3 we derive the fluctuation dissipation relation for this model and show that at equilibrium the input of energy into the system by the noise and the reduction of energy from the viscous terms balance appropriately, and consequently that the fluctuation dissipation theorem is satisfied.

2.2 Finite Element Approximation

In order to construct a finite element discretisation, we seek a weak form of Equation (2.1) by performing a volume integral with the weight function $w(\mathbf{x})$ to give:

$$\int_V \rho w(\mathbf{x}) \frac{\partial u_i}{\partial t} dV = -\int_V \frac{\partial w(\mathbf{x})}{\partial x_j} \sigma_{ij} dV + \int_S f_i w(\mathbf{x}) dS. \tag{2.6}$$

where f_i are the external surface forces. This corresponds to the standard application of the finite element method[76]. The second integral in Equation (2.6) can now be evaluated by substituting in the components of the stress.

$$\begin{aligned} \int_V \frac{\partial w(\mathbf{x})}{\partial x_j} \sigma_{ij} dV &= \int_V \left(\mu \frac{\partial w(\mathbf{x})}{\partial x_j} \frac{\partial u_i}{\partial x_j} + \mu \frac{\partial w(\mathbf{x})}{\partial x_j} \frac{\partial u_j}{\partial x_i} + \lambda \frac{\partial w(\mathbf{x})}{\partial x_i} \frac{\partial u_j}{\partial x_j} \right) dV \\ &\quad + \int_V \frac{\partial w(\mathbf{x})}{\partial x_j} \sigma^e_{ij} dV \\ &\quad + \int_V \frac{\partial w(\mathbf{x})}{\partial x_j} \sigma^t_{ij} dV. \end{aligned} \tag{2.7}$$

Equation (2.7) contains first order derivatives of both the velocity vector u_i and the the weight function $w(\mathbf{x})$. Thus, both the functions u_i and $w(\mathbf{x})$ must be differentiable over the domain of the differential equation and square integrable. Therefore, a suitable space is such that $u_i, w(\mathbf{x}) \in H^1_0(\omega)$ where ω is the domain of the differential equation in 3-space, while the tensors σ^e_{ij} and σ^t_{ij} can be defined as $\sigma^e_{ij}, \sigma^t_{ij} \in L_2$.

With the solution space now defined, we subdivide the domain ω of the differential equation into finite elements with nodes that are fixed in the Lagrangian frame of the material so that the velocity is expressed in the form $u_i = \sum_\alpha v_{i\alpha} \phi_\alpha$ where ϕ_α are base vectors that span the subspace of $H^1_0(\omega)$ so that,

$$\frac{Du_i}{Dt} = \sum_\alpha \frac{\partial v_{i\alpha}}{\partial t} \phi_\alpha. \tag{2.8}$$

Thus, with this discretisation Equation(2.6) becomes the following system of ordinary differential equations:

$$M_{pq}\frac{\partial v_q}{\partial t} + K_{pq}v_q = E_p + N_p, \tag{2.9}$$

where $M_{p(i,\beta)q(j,\alpha)}$, $K_{p(i,\beta)q(j,\alpha)}$, $E_{p(i,\beta)}$ and $N_{p(i,\beta)}$ are defined below. This treatment corresponds to the Galerkin formulation[76] of finite element analysis where the weight functions $w(\mathbf{x})$ are chosen to be the same as the basis functions ϕ_α. We have also introduced the indices p and q that run over the full dimension of the finite element system such that p can be written as $p(i,\beta)$ and q can be written as $q(j,\alpha)$. This gives:

$$M_{p(i,\beta)q(j,\alpha)} = \delta_{ij}\left(\int_V \rho\phi_\alpha\phi_\beta dV\right), \tag{2.10}$$

$$K_{p(i,\beta)q(j,\alpha)} = \int_V \mu\frac{\partial\phi_\beta}{\partial x_c}\frac{\partial\phi_\alpha}{\partial x_c}\delta_{ij} + \mu\frac{\partial\phi_\beta}{\partial x_j}\frac{\partial\phi_\alpha}{\partial x_i} + \lambda\frac{\partial\phi_\beta}{\partial x_i}\frac{\partial\phi_\alpha}{\partial x_j}dV, \tag{2.11}$$

$$E_{p(i,\beta)} = -\int_V \frac{\partial\phi_\beta}{\partial x_j}\sigma^e_{ij}dV, \tag{2.12}$$

$$N_{p(i,\beta)} = -\int_V \frac{\partial\phi_\beta}{\partial x_j}\sigma^t_{ij}dV. \tag{2.13}$$

Equation (2.9) describes a linear system of Langevin equations that can be solved for $\frac{\partial v_q}{\partial t}$ by inverting the the mass matrix M_{pq}. Physically, the different matrices presented in Equation (2.9) describe each of the particular processes that govern the behaviour of a macromolecule in the algorithm. The mass matrix M_{pq} (2.10) describes how mass is distributed throughout the finite elements, K_{pq} (2.11) describes how the model dissipates energy through viscosities, E_p (2.12) is an elastic force vector and N_p (2.13) is a thermal force vector.

2.3 Thermal Noise

The remaining undefined quantity in Equation (2.9) is the fluctuating stress tensor σ_{ij}^t. To derive the form of σ_{ij}^t we must first derive the fluctuation dissipation relation for this system under the assumption that the system as at constant temperature.

For the case of a Kelvin-Voigt material, the derivation of the fluctuation-dissipation theorem is simplified because the elastic stress in the model is derived from a strain energy that depends only upon the instantaneous deformation of the system. Consequently, the elastic terms in this model are conservative and the energy stored during a structural distortion is fully recovered when the material returns to equilibrium. By contrast, in material models with fading memory (such as the Maxwell model), the viscoelastic stress is dependent on the strain history of the material. In such cases, there will be additional dissipation of energy due to the fading memory within the material, and the derivation of the corresponding fluctuation-dissipation theorem is less straightforward.

The constant temperature assumption means that biological events that involve endothermic or exothermic reactions cannot be modeled explicitly. This is valid assumption for situations where biomolecules are in thermal equilibrium with their surrounding environment, for example proteins undergoing thermal fluctuations in a particular solvent. However, this approximation is not valid when biomolecules are undergoing chemical reactions with their environment. This issue could be addressed by associating the temperature with a particular finite element as oppose the the whole system and then modelling heat flow between the finite elements using a heat equation.

2.3.1 Fluctuation Dissipation Theorem

The overall equation of motion of the macromolecule comprises a linear system of Langevin equations. Deriving the fluctuation dissipation relation for this specific system is necessary to provide the statistics of N_p. We can re-write Equation (2.9) as:

$$M_{p\alpha}\frac{\partial v_\alpha}{\partial t} = N_p - K_{p\gamma}v_\gamma - \nabla_p U(\mathbf{x}), \tag{2.14}$$

where the elastic force vector E_p has been re-written in the form $E_p = -\nabla_p U(\mathbf{x})$, where the potential U is the strain energy. We have relabelled the dummy indices from q to α and γ for clarity.

The derivation of the fluctuation dissipation theorem first considers the total kinetic energy $\mathcal{E}$ of the discretised system. given by:

$$\mathcal{E} = \frac{v_\alpha M_{\alpha\beta} v_\beta}{2}. \tag{2.15}$$

Equation (2.15) is exact within the discretised finite element framework though only true since the mass matrix matrix is a constant. This corresponds to assuming that the total mass within each element is conserved for all possible deformations. In practise for 3D linear finite elements, discussed in Section 1.3.4, a constant mass matrix means that mass is distributed uniformly throughout a distinct finite element and that the density of that element changes as the volume of that finite element increases or decreases.

In general for globular proteins, the density at any particular point within the protein will be a function of both position and time. For example, spatially proteins are known to be inhomogenous and thus regions rich in secondary structure will have higher densities than surface regions. This introduces a limitation within the current model that density gradients within proteins are difficult to model and can only be approximated through the relative densities between neighboring finite elements. Though this could in principle be taken care of during coarse graining and parametrisation, it would introduce a maximum element size in order to preserve the density gradient to some tolerable level.

Temporal variation in density would correspond to binding of new molecules to a particular globular protein. While this can be modeled by recalculating the mass matrix with a new set of starting densities given there are no changes to the structure

of the globular protein and the new mass distribution does not introduce high density gradients. In general, one would have to recalculate the the finite element mesh to represent the new bound structure. An example of this sort of problem is given in Chapter 6 where the dynamics of two different biochemical states of the molecular motor dynein are modelled at thermal equilibrium. However, dynamic switching between the two states of the biomolecule would not be possible in the current method because the temperature of the system would change and the difficulty of relating the properties of the two finite element meshes.

From equation (2.15), the probability[87] of finding the system with a given given kinetic energy is therefore proportional to:

$$P \sim \exp(-\frac{v_\alpha M_{\alpha\beta} v_\beta}{2k_B T}). \tag{2.16}$$

Since this is a generalised normal distribution, it follows that the second moment average of the node velocities, at equilibrium, must be:

$$Q_{pq} = \langle v_p v_q \rangle = k_b T M^{-1}_{pq}, \tag{2.17}$$

which is the equipartition theorem for this system. Equation (2.17) is exact within the discretised finite element framework, so the fluctuation dissipation theorem derived from it is also exact. However, in practice numerical errors will occur due to the numerical integration of equation (2.9), and is discussed in Section 3.2.1 The derivation of the fluctuation dissipation relation for this system follows by analyzing fluctuations in the energy variable Q_{pq}, considering its change ΔQ_{pq} during a small time step Δt (that will become infinitesimally small) such that:

$$\Delta Q_{pq} = \langle \Delta v_p v_q + v_p \Delta v_q + \Delta v_p \Delta v_q \rangle = 0, \tag{2.18}$$

where from Equation (2.14):

$$\Delta v_p = \Delta t M_{p\alpha}^{-1}(N_\alpha - K_{\alpha\gamma}v_\gamma + \nabla_\alpha U(\mathbf{x})). \tag{2.19}$$

Thus, the terms on the right hand side of Equation (2.18) are given by:

$$\Delta v_p v_q = \Delta t M_{p\alpha}^{-1}(N_\alpha - K_{\alpha\gamma}v_\gamma + \nabla_\alpha U(\mathbf{x}))v_q, \tag{2.20}$$

$$v_p \Delta v_q = v_p \Delta t M_{q\delta}^{-1}(N_\delta - K_{\delta\epsilon}v_\epsilon + \nabla_\delta U(\mathbf{x})), \tag{2.21}$$

and finally:

$$\Delta v_p \Delta v_q = \Delta t^2 M_{p\alpha}^{-1}(N_\alpha - K_{\alpha\gamma}v_\gamma + \nabla_\alpha U(\mathbf{x}))M_{q\delta}^{-1}(N_\delta - K_{\delta\epsilon}v_\epsilon + \nabla_\delta U(\mathbf{x})). \tag{2.22}$$

To evaluate the ensemble averages of Equations (2.20), (2.21) and (2.22) to order Δt we note $\langle v_p \rangle = 0$, and that:

$$\langle M_{p\alpha}^{-1}\nabla_\alpha U v_q + M_{q\delta}^{-1}\nabla_\delta U v_p \rangle = 0. \tag{2.23}$$

Because at equilibrium the total energy of the system is simply the sum of the kinetic and potential energies, the probability of finding the system in any given microstate is also the product of the probability distributions describing the range of the potential and kinetic energies the system can adopt. Therefore, the potential and velocity terms are uncorrelated at equilibrium. Since the kinetic energy contains only terms quadratic in v_p, it follows that $\langle v_p \rangle = 0$, so Equation (2.23) must hold. Equations (2.20)-(2.22) then simplify to:

$$\begin{aligned} \langle \Delta v_p v_q \rangle &= -\Delta t M_{p\alpha}^{-1} K_{\alpha\gamma} \langle v_\gamma v_q \rangle \\ &= -\Delta t k_B T M_{p\alpha}^{-1} M_{\gamma q}^{-1} K_{\alpha\gamma}, \end{aligned} \tag{2.24}$$

$$\begin{aligned}\langle v_p \Delta v_q \rangle &= -\Delta t M^{-1}_{p\delta} K_{\delta\epsilon} \langle v_p v_\epsilon \rangle \\ &= -\Delta t k_B T M^{-1}_{q\delta} M^{-1}_{p\epsilon} K_{\delta\epsilon}, \end{aligned} \tag{2.25}$$

$$\langle \Delta v_p \Delta v_q \rangle = \Delta t^2 M^{-1}_{p\alpha} M^{-1}_{q\delta} \langle N_\alpha N_\delta \rangle. \tag{2.26}$$

Direct substitution of Equations (2.24)-(2.26) into (2.18) leads to the following,

$$\Delta t^2 \langle M^{-1}_{p\alpha} N_\alpha M^{-1}_{q\delta} N_\delta \rangle = \Delta t k_B T (M^{-1}_{p\alpha} K_{\alpha\gamma} M^{-1}_{\gamma q} + M^{-1}_{q\delta} K_{\delta\epsilon} M^{-1}_{p\epsilon}). \tag{2.27}$$

Multiplying through by the mass matrix gives:

$$\Delta t^2 \langle \delta_{p\alpha} N_\alpha \delta_{q\delta} N_\delta \rangle = \Delta t k_B T (\delta_{p\alpha} \delta_{\gamma q} K_{\alpha\gamma} + \delta_{q\delta} \delta_{p\epsilon} K_{\delta\epsilon}), \tag{2.28}$$

and so it follows that,

$$\langle N_p N_q \rangle = \frac{k_B T}{\Delta t} (K_{pq} + K_{qp}) \tag{2.29}$$

which is the fluctuation dissipation relation for the equation of motion in Equation (2.14).

As a further check on this result, we note that equation (2.14) can be cast as a stochastic differential equation:

$$dv_p = -M^{-1}_{p\alpha} K_{\alpha\beta} v_\beta dt - M^{-1}_{p\alpha} \nabla_\alpha U(\mathbf{x})\, dt + B_{p\alpha} dW_\alpha \tag{2.30}$$

where the dW_α are increments in a set of independent Wiener processes. Our derived fluctuation dissipation relation (2.29) is equivalent to a velocity space diffusivity of

$$D_{pq} = B_{p\alpha} B^{\mathrm{T}}_{\alpha q} = k_{\mathrm{B}} T M^{-1}_{p\alpha} M^{-1}_{q\beta} \left(K_{\alpha\beta} + K_{\beta\alpha} \right). \tag{2.31}$$

Furthermore, the node positions are coupled to the node velocities via

$$dx_p = v_p dt. \tag{2.32}$$

It can be shown from the general theory of stochastic equations ([88], equation 3.79) that the system of coupled stochastic differential equations (2.30) and (2.32) are equivalent to the Fokker-Planck equation in (x,v) space for evolution of the probability distribution $\psi(\mathbf{x}, \mathbf{v})$ of the position and velocity vectors:

$$\frac{\partial \psi}{\partial t} = \frac{\partial}{\partial v_p} \left[\left(M^{-1}_{p\alpha} K_{\alpha\beta} v_\beta + M^{-1}_{p\alpha} \frac{\partial U(\mathbf{x})}{\partial x_\alpha} \right) \psi \right] - \frac{\partial}{\partial x_p} \left(v_p \psi \right) + \frac{1}{2} \frac{\partial}{\partial v_p} \frac{\partial}{\partial v_q} \left(D_{pq} \psi \right). \tag{2.33}$$

It is a straightforward, if lengthy, exercise to verify that the steady state of equation (2.33) is

$$\psi(\mathbf{x}, \mathbf{v}) = A \exp\left(-\frac{U(\mathbf{x})}{k_{\mathrm{B}} T} - \frac{v_p M_{pq} v_q}{2 k_{\mathrm{B}} T} \right) \tag{2.34}$$

which is the expected Boltzmann distribution. For more detail on this derivation see Appendix 1.

2.3.2 Fluctuation Dissipation Relation for Linear Elements

In order to solve Equation (2.29) and derive the nature of the thermal stress tensor σ^t_{ij}, we must choose a set of basis functions ϕ_α. The simplest choice of basis functions are those of a linear tetrahedron (See Section 1.3.4). Equation (2.11) provides an explicit expression for the viscous matrix K_{pq}:

$$K_{pq} = \int_V \mu \frac{\partial \phi_\beta}{\partial x_c} \frac{\partial \phi_\alpha}{\partial x_c} \delta_{ij} + \mu \frac{\partial \phi_\beta}{\partial x_j} \frac{\partial \phi_\alpha}{\partial x_i} + \lambda \frac{\partial \phi_\beta}{\partial x_i} \frac{\partial \phi_\alpha}{\partial x_j} dV. \tag{2.35}$$

In the case of linear elements the derivatives of the basis functions are constants and Equation (2.29) can easily be simplified. In order to satisfy the fluctuation dissipation relation, we must assign an appropriate form to the fluctuating stress tensor σ_{ij}^t that is δ-correlated in space and time. Firstly, σ_{ij}^t must be symmetric such that $\sigma_{ij}^t = \sigma_{ji}^t$ and must consist of at least 6 independent stochastic processes as there are 6 degrees of freedom per element discussed in more detail in Chapter 3. The solution we have found has a total of 7 distinct stochastic processes and is of the following form:

$$\sigma_{ij}^t = \left(\frac{2k_B T}{V \Delta t} \right)^{\frac{1}{2}} \left(\mu^{\frac{1}{2}} X_{ij} + \lambda^{\frac{1}{2}} X^0 \delta_{ij} \right). \tag{2.36}$$

where X_{ij} is a stochastic tensor containing 6 independent stochastic processes such that $X_{ij} = X_{ji}$ and X^0 is a stochastic variable independent of any variable in X_{ij} such that:

$$\langle X_{ij} \rangle = 0, \tag{2.37}$$

$$\langle X^0 \rangle = 0, \tag{2.38}$$

$$\langle X_{ij} X_{kl} \rangle = \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}, \tag{2.39}$$

$$\langle X^0 X^0 \rangle = 1, \tag{2.40}$$

$$\langle X^0 X_{ij} \rangle = 0. \tag{2.41}$$

Note that the correlation function for the shear noise in Equation (2.39) is equivalent to that used by Sharma and Patankar[53].

The thermal stress tensor is δ-correlated in both space and time. The spatial δ-correlation is ensured by the finite element discretisation of the system, which guarantees that each element is independent of all the others. Since the viscous dissipation

within a single element depends only upon the instantaneous deformation rate, there is no dependence on history of deformation and the fluctuations must also be δ-correlated in time.

We now show that this choice satisfies Equation (2.29) and thus verify that for this Kelvin-Voigt material model the fluctuation dissipation theorem is obeyed. Note that we also convert from the p and q notation back to i, β and j, α so that the resulting matrices from Equation (2.29) can be directly compared.

$$\begin{aligned} \langle N_{p(i,\beta)} N_{q(j,\alpha)} \rangle &= \left\langle \int_V \frac{\partial \phi_\beta}{\partial x_c} \sigma^t_{ic} dV \int_V \frac{\partial \phi_\alpha}{\partial x_d} \sigma^t_{jd} dV \right\rangle && (2.42) \\ &= V^2 \frac{\partial \phi_\beta}{\partial x_c} \frac{\partial \phi_\alpha}{\partial x_d} \langle \sigma^t_{ic} \sigma^t_{jd} \rangle && (2.43) \end{aligned}$$

We now substitute Equation (2.36) into Equation (2.43), also using Equations (2.37)-(2.41), to yield the following:

$$\begin{aligned} \langle N_p N_q \rangle &= \left(\frac{2 k_B T V}{\Delta t} \right) \left(\mu \frac{\partial \phi_\beta}{\partial x_c} \frac{\partial \phi_\alpha}{\partial x_c} \delta_{ij} + \mu \frac{\partial \phi_\beta}{\partial x_j} \frac{\partial \phi_\alpha}{\partial x_i} + \lambda \frac{\partial \phi_\beta}{\partial x_i} \frac{\partial \phi_\alpha}{\partial x_j} \right) \\ &= \left(\frac{k_B T}{\Delta t} \right) \int_V 2\mu \frac{\partial \phi_\beta}{\partial x_c} \frac{\partial \phi_\alpha}{\partial x_c} \delta_{ij} + 2\mu \frac{\partial \phi_\beta}{\partial x_j} \frac{\partial \phi_\alpha}{\partial x_i} + 2\lambda \frac{\partial \phi_\beta}{\partial x_i} \frac{\partial \phi_\alpha}{\partial x_j} dV \\ &= \left(\frac{k_B T}{\Delta t} \right) (K_{pq} + K_{qp}) && (2.44) \end{aligned}$$

where the factor of two arises from the symmetry of the viscosity matrix K_{pq}. This simple solution for the thermal stress tensor is valid only for linear elements because it assumes that the compression across an element is uniform.

For second (and higher) order elements, it is significantly less straightforward to obtain fluctuating stress terms that satisfy the fluctuation dissipation relation. The difficulty arises because the derivatives of the basis functions are no longer constant, so the simple rearrangements in Equation (2.42) to (2.44), where the integrals are trivial, are no longer possible. In general, the fluctuating stress terms for second (and higher)

order elements depend in a non-trivial manner on the shape of the element, making them impractical for our computational scheme. In Section 3.2.7 below a simple scheme is presented which allows elastic contributions to the stress to be treated using second-order elements, whilst retaining a first-order scheme for viscous and thermal stresses.

2.4 External Fluid Dynamics

A proper treatment of the interaction between an external fluid and an immersed soft body involves solving the Navier Stokes equation in the external fluid with with continuity of velocity and surface forces at the boundary. This is a non-trivial problem due to the complexity of deriving the thermal noise terms, consistent with the dissipation of the Navier Stokes equation, where the boundary conditions are time dependent due to the motion and deformation of the soft immersed body. This would significantly increase the computational cost per timestep. We will consider a much simpler problem where the immersed soft body experiences friction due to the motion relative to a fixed background medium.

2.4.1 Mathematical Model of the External Fluid Dynamics

To develop a simple model for the external fluid dynamics, we shall consider that there is a sphere of a specific radius on each node. The radius of the sphere is defined by the length scale of the finite element mesh and in principle maybe different on each node. The drag force F_p on each node is assumed to be equal to the isolated Stokes' drag on its corresponding sphere:

$$F_p = -6\pi R(p)\mu^s v_p. \tag{2.45}$$

Here $R(p)$ is the radius of the sphere on a specific node p, μ^s the external solvent viscosity and v_p is the velocity on a specific node p. For convenience, we can re-express equation (2.45) in terms of a viscosity matrix ${K^e}_{pq}$ and a velocity vector v_p:

$$F_p = K^e{}_{p\alpha} v_\alpha, \tag{2.46}$$

where $K^e{}_{pq}$ is defined by

$$K^e{}_{pq} = \eta_p \delta_{pq}, \tag{2.47}$$

and the vector of friction constants is defined as

$$\eta_p = 6\pi R_p \mu^s. \tag{2.48}$$

This is the simplest form of interaction with an external fluid and is equivalent to that used in Brownian dynamics[58] simulations where hydrodynamic interactions are neglected. Although this model does not capture long range hydrodynamic interactions between different parts of a molecule that are moderated by the solvent, it does take some account of the influence of the external viscosity on the dynamics of the system. The new external fluid viscosity will alter the fluctuation dissipation relation that was derived in Section (2.3.1) and we are required to repeat the derivation and add an additional noise term to counterbalance this new viscosity.

2.4.2 Fluctuation Dissipation Relation with an External Fluid

In order to derive the fluctuation dissipation relation for the finite element system we must obtain the equation of motion for the deformable soft body with the external hydrodynamics. To the finite element equation of motion (2.14), we must now add two additional forces; the viscous force vector F_p in its matrix form defined in equation (2.47); and an additional noise vector N^e due to the external fluid. Thus, the new equation of motion is given by:

$$M_{p\alpha} \frac{\partial v_\alpha}{\partial t} = (N_p + N^e{}_p) - (K_{p\gamma} + K^e{}_{p\gamma}) v_\gamma - \nabla_p U(\mathbf{x}). \tag{2.49}$$

However by making two substitutions:

$$K'_{pq} = K^{e}{}_{pq} + K_{pq}, \tag{2.50}$$

$$N'_{pq} = N^{e}{}_{pq} + N_{pq}. \tag{2.51}$$

We can transform this back into the form the form of equation (2.14),

$$M_{p\alpha}\frac{\partial v_\alpha}{\partial t} = N'{}_p - K'_{p\gamma}v_\gamma - \nabla_p U(\mathbf{x}). \tag{2.52}$$

Furthermore, the derivation of the fluctuation dissipation relation is independent of the form of the viscosity matrix and so the form of the fluctuation dissipation relation must be the same. Therefore, the fluctuation dissipation relation for this system in terms of the noise vector N'_p and viscosity matrix K'_{pq} is:

$$\langle N'{}_p N'{}_q \rangle = \frac{k_B T}{\Delta t}(K'{}_{pq} + K'{}_{qp}). \tag{2.53}$$

We must now solve (2.53) for the exact form of the new noise vector $N^{e}{}_p$.

2.4.3 Solution for the External Fluid Noise

To solve for the form of the external fluid noise $N^{e}{}_p$ we note that N_p and $N^{e}{}_p$ are independent processes, so that $\langle N_p N^{e}{}_p \rangle = 0$. This is a consequence of the linear finite element solution. The solutions to both the internal dynamics and external dynamics are linear, and can be added together to form a new solution for the internal and external system. Thus, there are no correlations between the internal and external noise. Therefore equation (2.53) separates into internal and external components to give:

$$\langle N_p N_q \rangle + \langle N^{e}{}_p N^{e}{}_q \rangle = \frac{k_B T}{\Delta t}(K_{pq} + K_{qp}) + \frac{k_B T}{\Delta t}(K^{e}{}_{pq} + K^{e}{}_{qp}). \tag{2.54}$$

By substitution of the solution of the noise vector N_p in Section (2.3.2) into (2.54), we can eliminate the internal viscosity terms K_{pq} and reduce the problem to that of solving for the external noise vector ${N^e}_p$ in terms of the external viscosity matrix ${K^e}_{pq}$:

$$\langle {N^e}_p {N^e}_q \rangle = \frac{k_B T}{\Delta t}({K^e}_{pq} + {K^e}_{qp}). \tag{2.55}$$

Equation (2.55) and (2.54) show that the internal and external noise are not coupled and one can merely solve the two internal and external noise problems independently of one another.

Thus, in order to solve for the external noise vector ${N^e}_p$ we substitute in the form of the external viscosity matrix K_{pq} from equation (2.47) so that:

$$\langle {N^e}_p {N^e}_q \rangle = \frac{2k_B T \eta_p}{\Delta t}\delta_{pq}. \tag{2.56}$$

Equation (2.56) is the fluctuation dissipation relation for the external fluid viscosity and external noise alone. The external noise vector ${N^e}_p$ is then of the following form:

$${N^e}_p = \left(\frac{2k_B T \eta_p}{\Delta t}\right)^{\frac{1}{2}} {X^e}_p \tag{2.57}$$

Where ${X^e}_p$ is a stochastic vector with the following correlations:

$$\langle {X^e}_p \rangle = 0 \tag{2.58}$$

$$\langle {X^e}_p {X^e}_q \rangle = \delta_{pq} \tag{2.59}$$

2.5 Summary

In this chapter, I have introduced the material model that includes viscous damping, elasticity and thermal noise and I shown that within a finite element approximation the thermal noise for this system can be derived locally both with and without an external

solvent.

Chapter 3

The Numerical Method and Validation

The mathematical results of Chapter 2 form the basis for solving equation (2.1). In this chapter, I will describe the implementation of the algorithm and its validation via a series of test problems. Section 3.1 discusses the construction of the algorithm, Section 3.2 the numerical validation of only the continuum mechanical model in the absence of an external solvent and Section 3.3 the validation of the continuum model with an external solvent.

3.1 Numerical Method

As discussed in Chapter 2, the finite element discretisation results in a system of Langevin equations for the nodal velocities v_p:

$$M_{pq}\frac{\partial v_q}{\partial t} = -K_{pq}v_q + E_p + N_p. \tag{3.1}$$

Equation (3.1) can be numerically integrated using a standard time integrator such as Runge-Kutta[89] (RK), velocity Verlet[90] or Euler[91]. An example of a simple iterative loop to perform a time step is given below.

1. Characterise the initial conditions such as starting position and structure of the finite element mesh.

2. Calculate M_{pq}, K_{pq}, E_p, N_p and M_{pq}^{-1}.

3. Evaluate the new velocity vector and node positions using a time integrator.

The computational bottleneck in this loop is the calculation of the required matrices and vectors. However, this part of the algorithm can easily be parallelised. The matrices are symmetric which reduces the computational load. Since the the mass matrix and its inverse remain constant through the simulation, they only need to be calculated once at the start of the simulation.

3.2 Validation of the Continuum Model

In this Section, we consider the continuum model without the presence of an external solvent. To validate the results related to the continuum model derived in Chapter 2 and to demonstrate that the numerical algorithm reproduces the thermal physics of the system correctly, the following tests were performed on the numerical model:

1. The average kinetic and average potential energy converge to the correct values required by the classical equipartition theorem for sufficiently small integration timesteps (Section 3.2.1).

2. The distribution of the nodal velocities matches the theoretical Gaussian result (Section 3.2.2).

3. Used Euler Beam Theory to show that the average amount of potential energy found in the first two Fourier modes of a long beam also agrees with the equipartition value (Section 3.2.3-3.2.4).

4. The distribution of the Fourier amplitudes matches the theoretical distribution (Section 3.2.5).

5. The anisotropy of the inertia tensor converges as the mesh resolution is increased to demonstrate spatial convergence of the new method (Section 3.2.8).

3.2.1 Testing the Average Potential and Kinetic Energies

The average potential and kinetic energy depends on the number of degrees of freedom the of system. In this Section only internal forces are considered so there is no solid body rotation or translation, effectively freezing out six degrees of freedom. Thus if n is the number of nodes in the system, the total number of traditional degrees of freedom is $3n-6$ and there an equal number of velocity degrees of freedom. Therefore, from equipartition[92] the average kinetic energy is given by:

$$\langle \mathcal{E} \rangle = \frac{(3n-6)k_B T}{2}. \tag{3.2}$$

If deformations are small then only harmonic terms in the elastic energy are important and the average potential energy becomes:

$$\langle U \rangle = \frac{(3n-6)k_B T}{2}. \tag{3.3}$$

The kinetic energy and potential energy for the system are then defined as follows:

$$\mathcal{E} = \frac{v_p M_{pq} v_q}{2}, \tag{3.4}$$

$$\begin{aligned} U &= \sum_{\gamma} \int_{V_0} \frac{G}{2} tr(FF^T)^{\gamma} + \frac{B}{2} \left(\det(F^{\gamma}) - \alpha\right)^2 dV_0 \\ &\quad + \sum_{\gamma} \int_{V_0} -\frac{3G}{2} - \frac{B}{2} \left(\frac{G}{B}\right)^2 dV_0. \end{aligned} \tag{3.5}$$

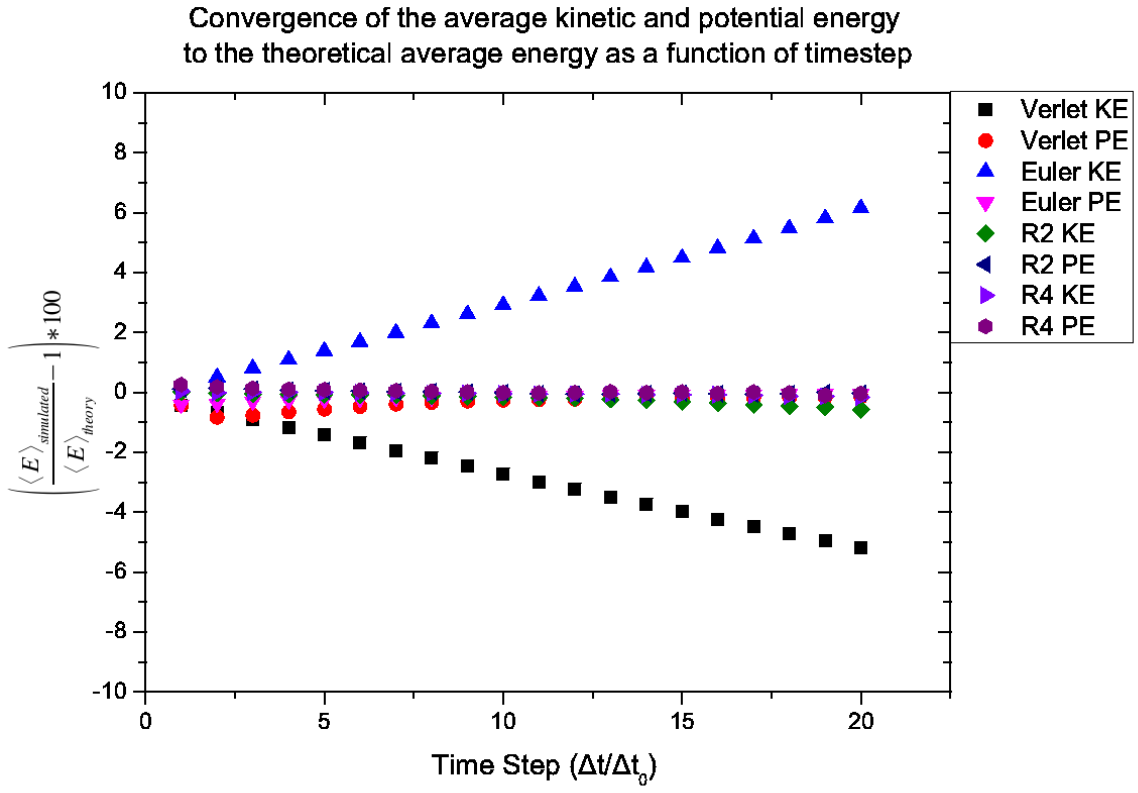
Here the sum over γ represents a sum over all the elements in the system and V_0 is the

rest volume of each individual element.

Dimensionless systems were considered, in which the density, viscosities, μ and λ and elastic moduli G and B were set to unity throughout, with $k_BT = 0.0001$ to ensure small deformations. To test that the kinetic and potential energies comply with equipartition, a simulation of a 54 element cylindrically meshed beam was performed, which was constructed using the GMSH package[93]. The system was first allowed to equilibrate for a million time steps and then the average kinetic and potential energies were calculated. The different time integrators; Euler[91], velocity Verlet[90], second order Runge-Kutta (R2) and fourth order Runge-Kutta[89] (R4) were tested. Figure 3.1 shows the percentage error in the energies as a function of the integration time step. The simulations were continued until the sampling errors in the average kinetic and potential energies were sufficiently small that the trends in Figure 3.1 could be clearly observed (this required 4 million timesteps for equilibration, and 20 million timesteps production run). The simulation performed with all four integrators gives the correct equipartition value for short integration timesteps, indicating that the inclusion of thermal fluctuations into FEA provides the expected equipartition values for the kinetic and potential energy associated with the thermal fluctuations of the mesoscale beam.

When larger time steps are used, the errors in the kinetic energy for the Euler and Verlet schemes grow linearly with time step. This is expected, as these schemes are both first order in time; whereas the errors in the higher order R2 and R4 schemes remain small over all time steps considered. All four integrators reproduce the potential energy to within 1%. Although R2 and R4 are more accurate than the Euler and Verlet algorithms, they also require more floating point operations per time step; to perform a single time step using R4 requires that the viscosity matrix K_{pq} and elasticity vector E_p be recalculated 4 times. Consequently in practice, the Euler integrator often offers the lowest computational expense, since an error of 1% is tolerable for most applications, and the stability limits on the maximum timestep are similar for all 4 integrators.

Figure 3.1: Convergence of the kinetic and potential energy averages as a function of the time step of a 54 element cylindrical mesh, where the unit time step is Δt_0=0.0001. This graph shows that as the time step decreases the error in the average energy in each quadratic degree of freedom tends to zero.



3.2.2 Nodal Velocity Distributions

As with comparing the average total kinetic energy we can also examine the distribution of kinetic energies. As discussed in Section (2.3.1) the distribution of the kinetic energy is a generalised Gaussian[87], as shown in Equation (2.16). Thus, the second moment average of the nodal velocities at equilibrium must be:

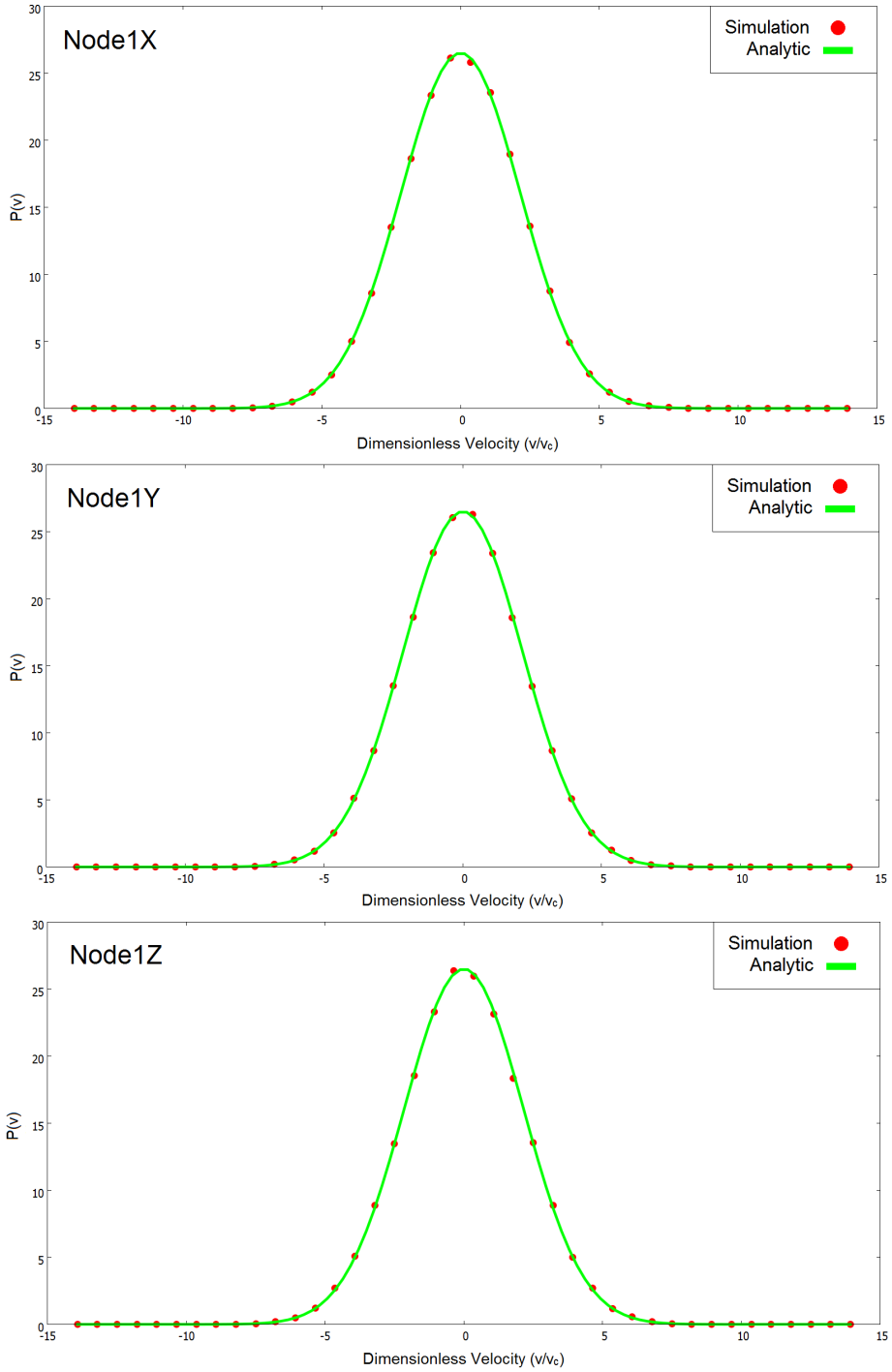
$$Q_{pq} = \langle v_p v_q \rangle = k_b T M^{-1}_{pq}. \tag{3.6}$$

From equations (2.16) and (3.6) the distribution of the velocity in each kinetic degree of freedom must be Gaussian with a variance given by Equation (3.6). In order to test this prediction, I ran long time scale simulations of a cylindrical beam, and measured the velocity distributions of nodes throughout the system. The results for a representative node are shown in Figure 3.2. The agreement between the theoretical and simulated results shows that the numerical solution to the equation of motion (2.1) preserves the correct statistical physics of the system.

3.2.3 Euler Beam Theory

The energy convergence tests in Section 3.2.1 and the distribution of the nodal velocities in Section 3.2.2 show that the fluctuation dissipation relation derived in Chapter 2 is obeyed and that the correct theoretical averages for the kinetic and potential energies are obtained as well as the correct nodal velocity distributions. However, these tests do not on their own show that the set of deformations predicted by the model are statistically correct. In order to test the conformational dynamics predicted by the stochastic finite element model we consider the flexing of a thin rod due to thermal fluctuations, and compare the vibrational modes this system sustains from those derived from Euler Beam Theory[94]. For a classical beam undergoing pure bending, the equilibrium deflection h due to an external torque τ is given by:

Figure 3.2: Simulated velocity distribution functions of the x, y and z components of a representative node within a cylindrical beam, plotted against the distributions expected theoretically. The velocities have been made dimensionless and scaled by a velocity $v_c = \left(\frac{k_B T}{m}\right)^{0.5}$ where m is the average mass on an element in the simulation.



$$\frac{d^2h(x)}{dx^2} = -\frac{\tau}{EI}, \tag{3.7}$$

where E is the Young's Modulus and I is the second moment of inertia of the cross-section. The product EI is the flexural rigidity. Equation (3.7) holds for thin beams when the deflection $h(x)$ is small relative to the length. For a uniform torque τ the solution to Equation (3.7) with boundary conditions $h(0) = 0$ and $h(L) = 0$ (where L is the beam length):

$$h(x) = \frac{\tau x(L-x)}{2EI}. \tag{3.8}$$

Since Equation (3.8) provides the solution of Equation (3.7) for beam undergoing bending due to an external torque, we can obtain the flexural rigidity EI by applying an external torque to a beam in the absence of thermal noise. The amount of work required to bend a beam is given by:

$$W = \frac{EI}{2}\int_0^L \left(\frac{\partial^2 h(x)}{\partial x^2}\right)^2 dx. \tag{3.9}$$

If $\tau = 0$ at both ends of the beam so that $\frac{\partial^2 h}{\partial x^2} = 0$, $h(x)$ can be expressed as a Fourier sine series:

$$h(x) = \sum_p h_p \sin\left(\frac{p\pi x}{L}\right). \tag{3.10}$$

Substituting Equation (3.10) into (3.9) gives the amount of work done in each of the Fourier modes that correspond to a degree of freedom of the system, so that:

$$W = \sum_p W_p, \tag{3.11}$$

where:

$$W_p = h_p^2 \left(\frac{EI}{4}\right) \left(\frac{p\pi}{L}\right)^4 L. \tag{3.12}$$

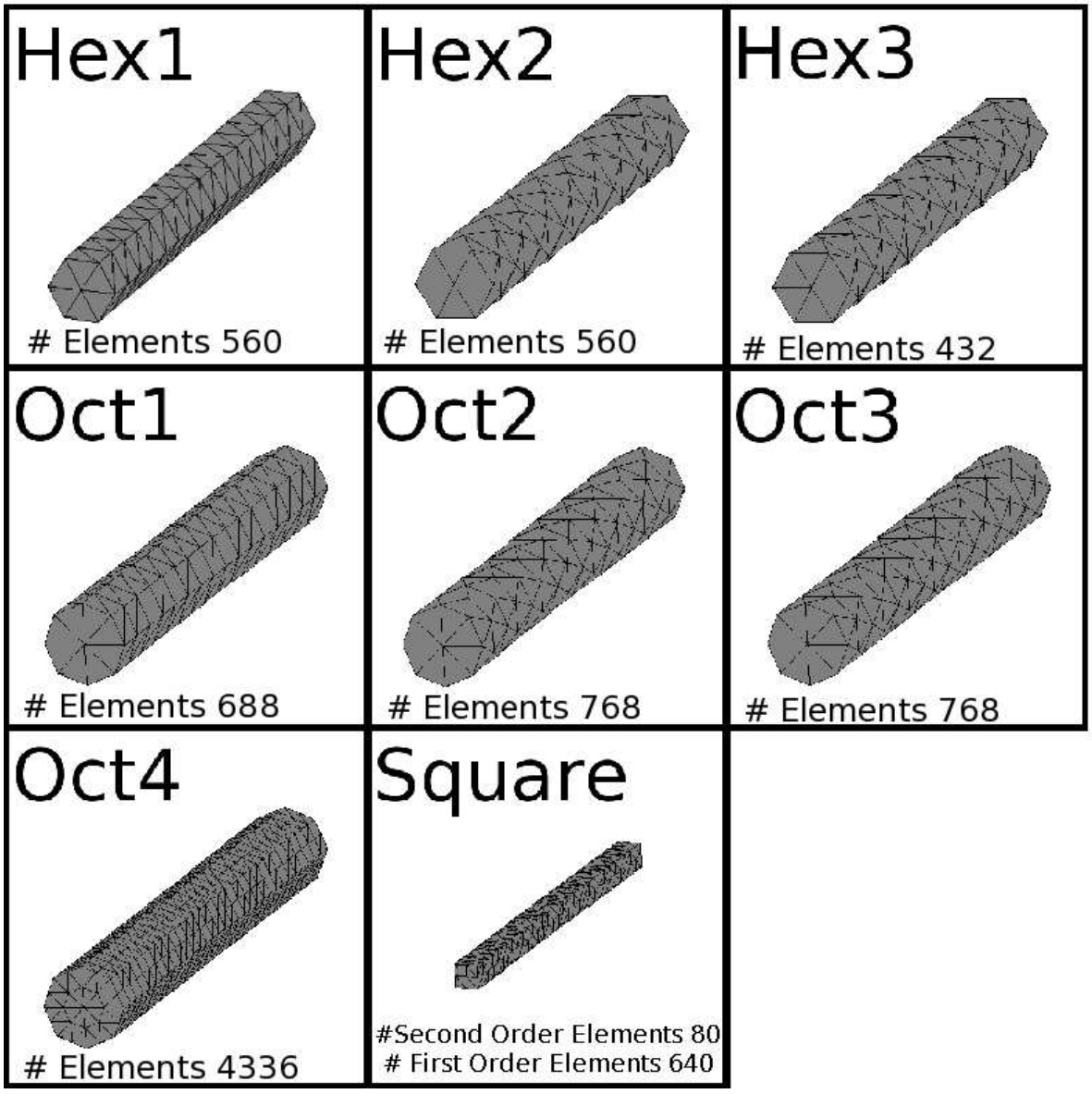
Therefore, from equipartition of energy it follows that if the beam is subject to thermal fluctuations then:

$$\langle W_p \rangle = \frac{k_B T}{2}. \tag{3.13}$$

3.2.4 Numerical Calculations for Beam Bending

I tested a total of eight finite element meshes; three have a hexagonal cross-section, four octagonal and one square (See Figure 3.3). The hexagonal and octagonal beams have a maximum radius of 10nm, the square beam has sides of length 10nm and all beams have a total length L of 160nm. In all simulations the viscosities were set to 3 mPas (3 times that of water), the elastic moduli G and B were set to $10MPa$, giving a Young's modulus of $20MPa$. The density was $1000\frac{Kg}{m^3}$. Thus, these simulations reproduce the thermal fluctuations of a hypothetical "nanogel" beam. The numerical tests are divided into two sections. First, we determine the flexural rigidity EI of the beams. This tests the influence of the mesh resolution, and in addition investigates the effect of different finite element meshes. Secondly, we obtain the average energies in the first and second Fourier modes to confirm that the deformations of the beams obey the correct statistics.

Figure 3.3: The eight beam meshes used to test configurational fluctuations in the stochastic finite element method. Only the surface meshes of Hex2 and Hex3 are the same, internally the element structure is different. Similarly, for Oct2 and Oct3 the internal nodes are placed slightly differently to ensure the the results obtained are independent of the arrangement of finite elements. Oct4 is the beam mesh used to perform the fine grained calculations in Section 3.2.3 and the square beam mesh is used in the second order element scheme described in Section 3.2.4. For the square beam mesh, all the linear elements in the system within the second order element structure are shown.



To determine the flexural rigidity an external torque τ was applied to the end of each beam with thermal noise. Prior to finite element discretisation, the governing continuum equation for this system is:

$$\rho\left(\frac{\partial u_i}{\partial t}+u_j\frac{\partial u_i}{\partial x_j}\right)=\frac{\partial \sigma_{ij}^v}{\partial x_j}+\frac{\partial \sigma_{ij}^e}{\partial x_j}+\tau_i. \tag{3.14}$$

To impose a torsional stress on each beam, a torque τ of magnitude $\tau = 5x \cdot 10^{-12} Nm$ (where x is the linear distance from the central axis of the beam) was applied to the end faces only. Stress-free boundary conditions were used elsewhere. The results of these eight calculations are presented in Table 1.

Table 3.1: Flexural Rigidity Results for Different Meshes

Beam	$\frac{(EI_x)_{Simulated}}{(EI_x)_{Theory}}$	$\frac{(EI_y)_{Simulated}}{(EI_y)_{Theory}}$
Hex1	1.70	1.70
Hex2	1.60	1.60
Hex3	1.82	1.75
Oct1	1.61	1.50
Oct2	1.48	1.48
Oct3	1.48	1.48
Oct4	1.30	1.26
Square	1.00	1.00

For linear finite elements the flexural rigidity of the long thin beams is larger than is predicted theoretically. This is a consequence of there being only a small number of linear elements across each cross-section, which artificially stiffens the rods. There are two alternate strategies to improve the accuracy of the flexural rigidity for linear finite elements. One solution (discussed in Section 3.2.6) is to use h-refinement wherein more linear finite elements are placed across the cross-section of the beam. The second is p-refinement (discussed in Section 3.2.7) using higher order elements to describe the displacements and elastic stresses.

Now that the flexural rigidity of each beam has been obtained (see Table 1), the thermal noise is reintroduced so that the Fourier modes can be extracted. The temperature of the system was set to be $300K$. To maintain small deformations, $\frac{k_B T L}{EI}$ is

set to be approximately 10^{-3}. With stress free boundary conditions everywhere, the governing equation for this simulation prior to finite element discretisation is now given by:

$$\rho\left(\frac{\partial u_i}{\partial t}+u_j\frac{\partial u_i}{\partial x_j}\right)=\frac{\partial \sigma^v_{ij}}{\partial x_j}+\frac{\partial \sigma^e_{ij}}{\partial x_j}+\frac{\partial \sigma^t_{ij}}{\partial x_j}. \tag{3.15}$$

For the Hex1-3 and Oct1-3 beams, a total of 21 independent repeat simulations were performed. This ensured sufficient sampling of the first and second Fourier modes in the x and y directions perpendicular to the beam axis. There are two important time scales in this system; the period of the first harmonic and the decay time of these oscillations. The period of oscillation can be found by considering the beams as vibrating strings[95] for which the period of the first harmonic is given in terms of the tension T on the string, the total length L and the mass per unit length μ:

$$\tau=2\left(\frac{\mu L^2}{T}\right)^{\frac{1}{2}}. \tag{3.16}$$

Equation (3.16) can be re-written in terms of the cross sectional area of the beam A such that $\mu=\rho A$. Similarly by performing dimensional analysis, the tension T can be expressed as $T=\frac{EI}{L^2}$. So that in terms of the important parameters the beam the period of the fundamental mode is of order:

$$\tau=2\left(\frac{\rho A L^4}{EI}\right)^{\frac{1}{2}}. \tag{3.17}$$

From this estimate the period of oscillation for the Hex1-3 and Oct1-3 beams are of order $10ns$; the longest decay time scale for these oscillations was measured by simulation and is also around $10ns$. Since the total simulation time was $1.5\mu s$, both of these important time scales were adequately sampled. Each Fourier amplitude was then averaged, the variance of the distribution obtained; substitution into Equation (3.12) then provides the average energy of that particular Fourier mode (see Table 2).

The results for the different meshes using the flexural rigidities from Table 1 all show

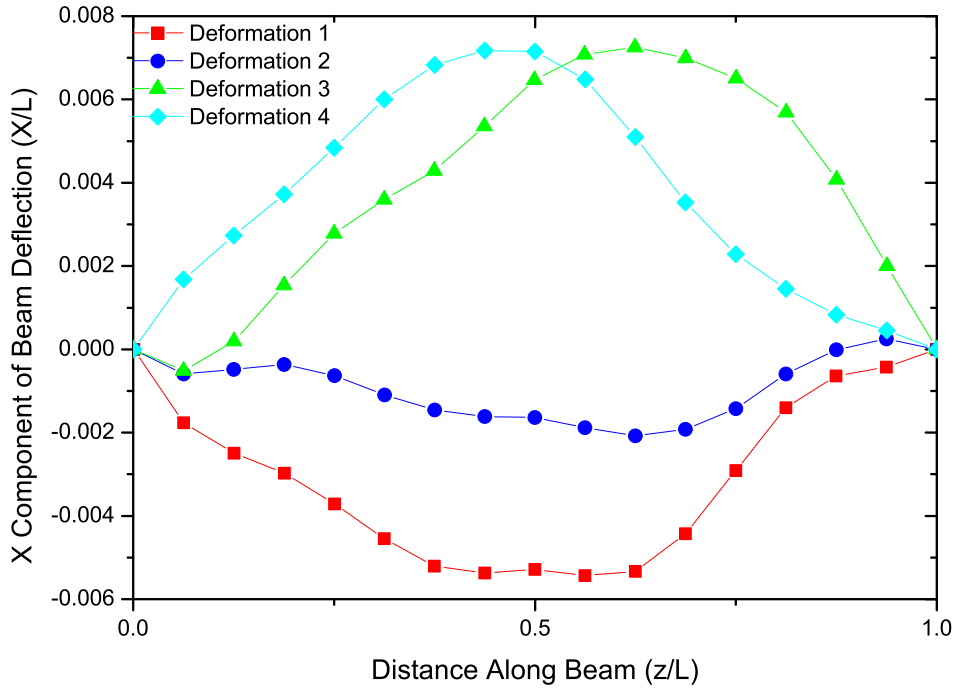
Table 3.2: Average energies in the first and second Fourier modes normalised by $\frac{k_BT}{2}$ (so that the correct theoretical answer is 1). FM1X and FM1Y denote the average energy in the first and second Fourier modes in the X and Y directions respectively, while FM2X and FM2Y refer to the average energy in the second Fourier modes. Where the error is given by the standard deviation of the resultant distribution of energies from each of the 21 different simulations.

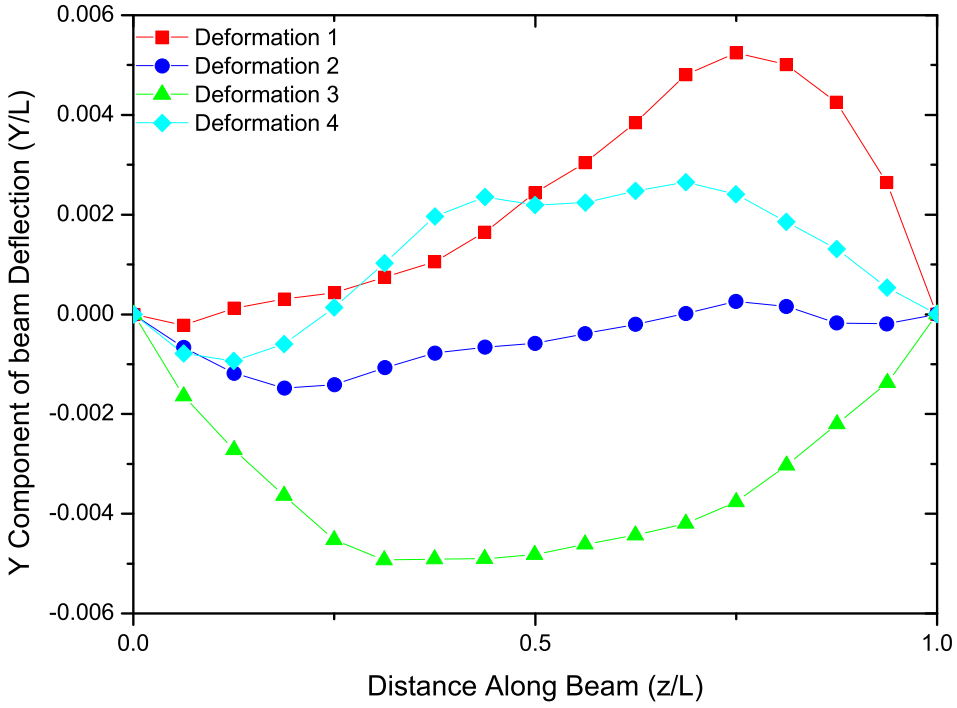
Beam	FM1X $\left(\frac{2\langle W_p\rangle}{k_BT}\right)$	FM1Y $\left(\frac{2\langle W_p\rangle}{k_BT}\right)$	FM2X $\left(\frac{2\langle W_p\rangle}{k_BT}\right)$	FM2Y $\left(\frac{2\langle W_p\rangle}{k_BT}\right)$
Hex1	1.014 ± 0.068	1.058 ± 0.038	1.000 ± 0.034	1.024 ± 0.032
Hex2	0.940 ± 0.064	0.982 ± 0.056	0.960 ± 0.034	0.966 ± 0.030
Hex3	0.976 ± 0.044	1.004 ± 0.062	0.910 ± 0.028	0.924 ± 0.024
Oct1	0.984 ± 0.048	0.928 ± 0.052	1.022 ± 0.030	1.052 ± 0.028
Oct2	1.026 ± 0.082	0.982 ± 0.062	1.046 ± 0.036	1.064 ± 0.042
Oct3	1.010 ± 0.076	0.906 ± 0.070	0.974 ± 0.044	0.980 ± 0.034

good agreement with the theoretical prediction for the average energy in the first and second Fourier bending modes. The results agree with the theoretical average energy predicted by the equipartition theorem within the calculated sampling error.

Figure 3.4 shows four representative conformations of the beams sampled from the FFEA simulations. These were obtained by plotting the centre of mass of different sub-sections of the beam along its length relative to the beam ends to represent the instantaneous configuration (consequently the ends of the beams always have a total displacement of zero). Furthermore, the deflections of each centre of mass of the beam follow a Gaussian distribution as expected for a beam subject to thermal noise.

Figure 3.4: Four different conformations adopted by the Hex1 beam due to thermal noise. The boundary condition of no external torque $\frac{\partial^2 h}{\partial x^2} = 0$ enables the deformations $h(x)$ to be measured relative to the positions of stationary beam ends.





From the calculation of the flexural rigidities from the finite element meshes of the eight beams (Table 1), and the calculation of the average energies in the first two Fourier modes, we conclude that the average energies obtained from the stochastic finite element model are in agreement with theoretical predictions. However, the linear approximation for the elements leads to an over-estimation of the flexural rigidity when the mesh contains too few elements. To demonstrate that this can be corrected by increased spatial refinement, two simulations were performed one employing a finer mesh of linear elements (See Section 3.2.6), and another using second order elements for the elastic deformation of the rods (See Section 3.2.7).

3.2.5 Distribution of the Fourier Amplitudes

The energy in a given Fourier mode is quadratic in the respective Fourier amplitude (as shown in Equation (3.12)). Therefore, the distribution of the Fourier amplitudes must be Gaussian[87]. The variance of the distribution of each of the Fourier amplitudes is given by:

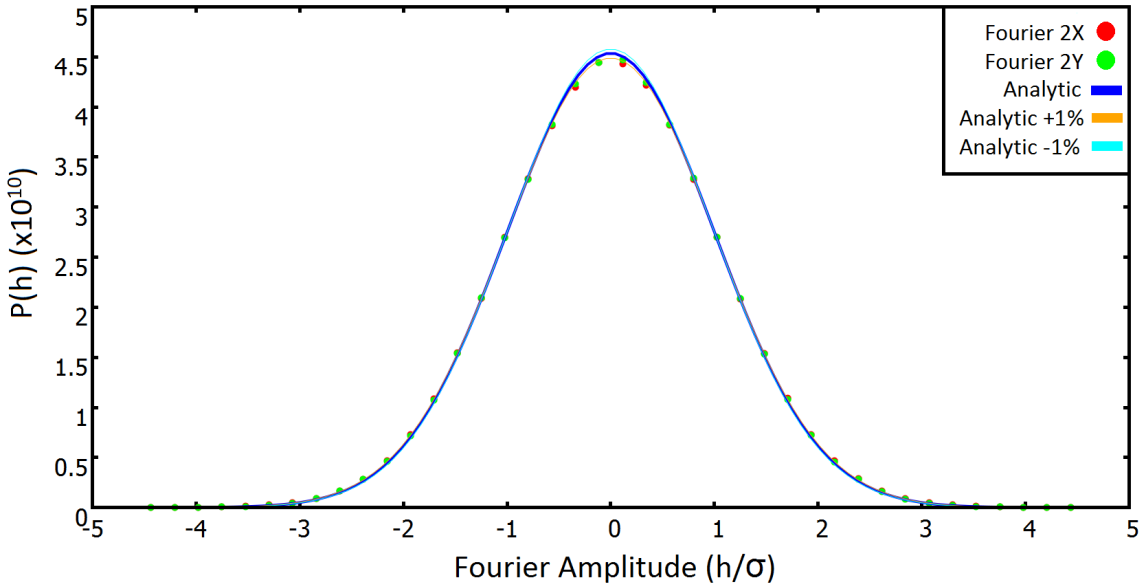
$$\langle h_p^2 \rangle = \left(\frac{k_B T}{2L} \right) \left(\frac{EI}{4} \right)^{-1} \left(\frac{p\pi}{L} \right)^{-4}. \tag{3.18}$$

For the Oct2 beam, I computed the distribution of the Fourier amplitudes by running long time scale stochastic finite element model simulations and comparing the results with Equation (3.18), as shown in Figure 3.5. The agreement between the theoretical and experimental curves confirms that the thermal statistics of the model are correct and consistent with the results derived in Chapter 2.

3.2.6 Fine Grained Mesh, h-refinement

To capture the bending of the beams more accurately I calculated the flexural rigidity using an octagonal mesh with four elements across the diameter of the beam (Oct 4) compared to the two used in Oct1, 2 and 3. The same viscosities, elastic moduli and

Figure 3.5: The simulated Second Fourier modes are shown in green and red and the analytic distribution is shown in blue. The orange and cyan lines represent ±1% error in the analytic standard deviation of the distributions. The Fourier amplitudes are normalised against the the second moment average of the analytic distribution, as given in Equation (3.18).



density were used as previously. As shown in Table 2, improving the mesh resolution halves the error in the flexural rigidity measured. However, this solution is more numerically costly as there are approximately 8 times as many elements to be considered in this finer grained mesh.

3.2.7 Second Order Element Solution, p-refinement

An alternative method for improving the spatial resolution is to use quadratic functions in the interpolation of displacements. This requires a solution of Equation (2.9) in which the elastic terms and the mass matrix are solved using second order elements[80]. The elastic stress is calculated using 10 node isoparametric tetrahedral elements from which Equations (2.10 and 2.12) can be solved using second order shape functions. In general, this integral cannot be performed analytically since in the second order regime the local compression within an element is not homogeneous. Thus, the integrals need to be

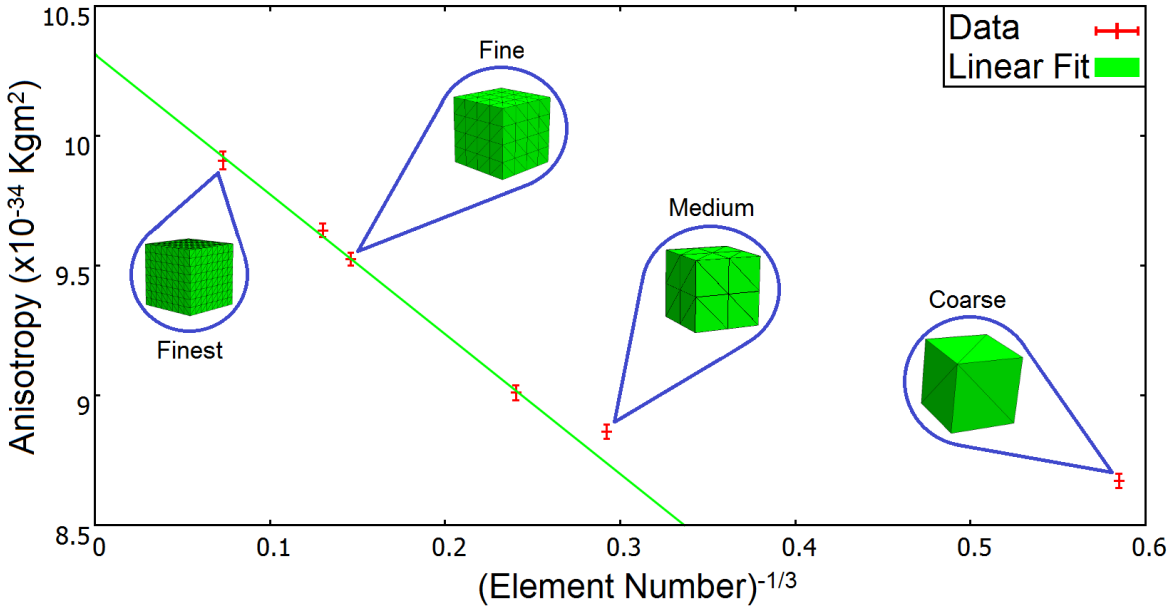
performed numerically using Gaussian quadrature. As discussed in Section 2.3.2, it is much more difficult to include viscous and thermal noise terms for a second order finite element mesh. We therefore retain a linear solution for the thermal and viscosity terms by subdividing each quadratic element into 8 linear elements, and using these sub-elements to calculate the viscous and thermal noise terms. The viscosity matrix and thermal force vector are calculated by subdividing each isoparametric tetrahedron into linear elements and then performing the integrals in Equations (2.11) and (2.13) for each of the linear sub-elements.

To test the quadratic element solution, I repeated the beam bending calculations and obtained the flexural rigidity for a simple square cross-section beam with sides of unit length . As shown in Table 3.1, the second order elements give the correct flexural rigidity for a square cross-section. In this case, the use of second order elements provides a more efficient method to improve the of accuracy of the solution than increasing the number of linear elements. Since the main increase in the computational expense for the quadratic elements arises from calculating the thermal and viscosity terms, which involve the contributions from the eight linear sub-elements that make up each quadratic element, the second order solution gives better efficiency in the trade-off between accuracy and computational expense. However, this method does suffer from stability issues.

3.2.8 Spatial Convergence

To demonstrate spatial convergence of the stochastic finite element model method, I simulated a series of 6 cubes with identical side length (1μm) but an increasing number of finite elements, as shown in Figure 3.6. The other material parameters such as the viscosity, temperature and density were held constant in all 6 simulations.

Figure 3.6: Convergence of the anisotropy of the inertia tensor as a function of the element number. The error bars are the standard deviation of the anisotropy. Meshes corresponding to the data points are also shown.



To characterise the thermal fluctuations of each cube I calculated the average fluctuations in the anisotropy of the inertia tensor. The inertia tensor I_{ij} is defined as:

$$I_{ij} = \int_V \rho(r^2\delta_{ij} - r_i r_j)dV, \tag{3.19}$$

where $\mathbf{r}$ is the position vector relative to the centre of mass and $r^2 = \mathbf{r}\cdot\mathbf{r}$. The inertia tensor is a positive definite symmetric tensor and therefore has three real eigenvalues and corresponding linearly independent eigenvectors. We define anisotropy of the inertia tensor as the difference between the largest and smallest eigenvalues. The anisotropy can then be time averaged as the cubes deform due to thermal fluctuations for long times. The results of these simulations are shown in Figure 3.6 alongside the meshes used in each simulation. It can be seen that with the exception of the coarsest mesh that the anisotropy converges linearly with the inverse cube root of the number of elements. This indicates that the error is first order with respect to the distance between nodes, as expected for linear finite elements.

3.3 Validation of the External Hydrodynamics

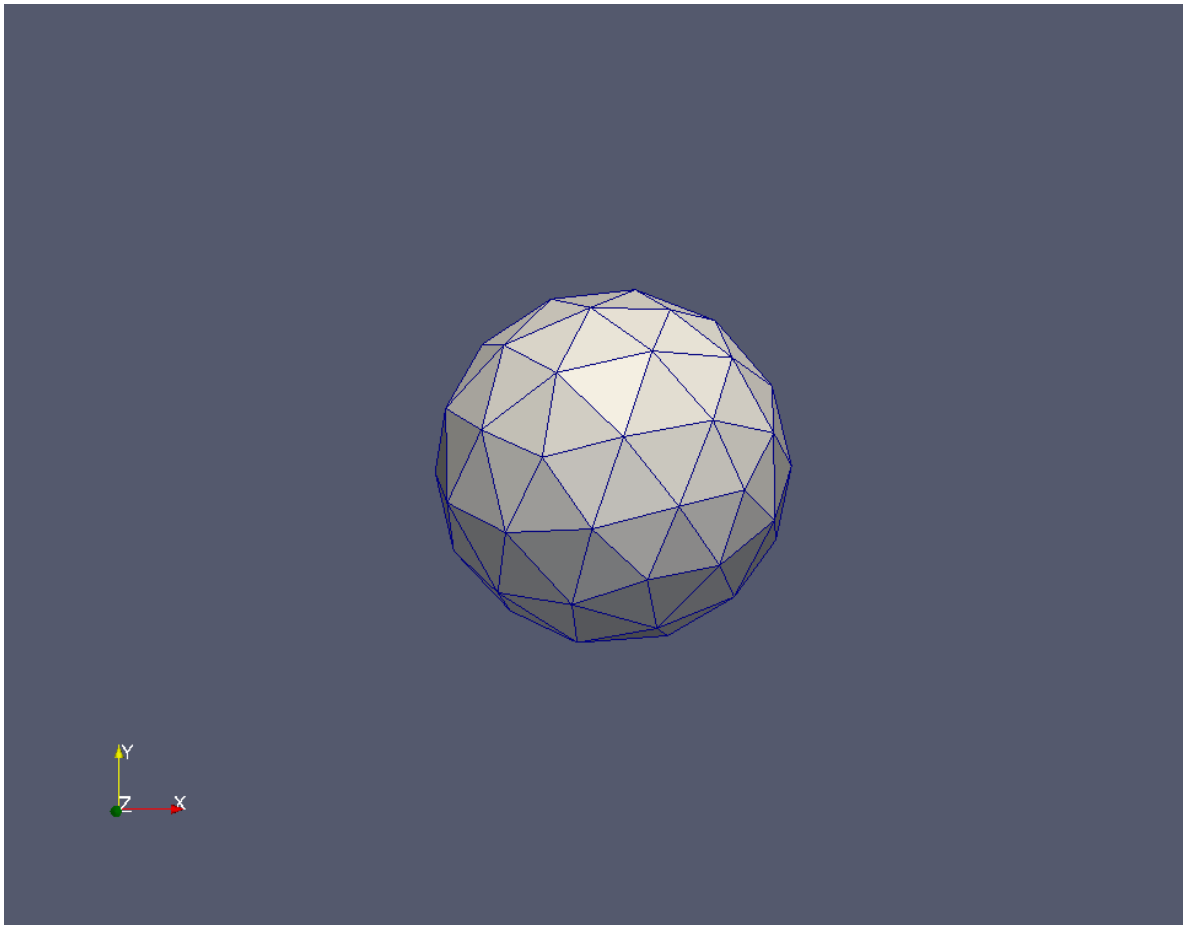
Having validated the simulations for the case of no external solvent, we now test the external hydrodynamics to ensure they too conform with expected behaviour. The external hydrodynamics used in this model are crude and correspond to only a background solvent that the nodes of the finite element mesh can drag against. The largest limitation is that the input fluid mechanics does not conserve momentum as the implementation mirrors that of Brownian dynamics and thus cannot replicate the physics of the Navier-Stokes equation. Furthermore, the interior nodes of the finite element mesh feel the influence of the background fluid as much as the surface nodes. In reality, this is unphysical as only the surface nodes should feel the external fluid and so the material model will not respond correctly to external flow. A more sophisticated fluid model could be included by instead resolving a surface force from the fluid onto the finite element mesh through the boundary element method. For now though, we will only include this crude model as it is sufficient to impose an external viscosity that we shall then use in chapter 6.

3.3.1 Validation of the External Fluid Dynamics

In order to validate the external fluid dynamics a number of numerical tests have been performed on a sphere (See Figure 3.7) of radius $100nm$ with elastic moduli similar to that of steel but with very low internal viscosity and an external viscosity similar to that of water. This ensures that deformation of the ball is small so that the ball moves as a rigid body on the time scale of the external fluid dynamics.

I have performed four different numerical simulations that test different aspects of the implementation of the external fluid. Specifically, I have tested that the trajectory of the steel ball agrees with theoretical predictions when the thermal noise is turned off; that the total average kinetic and average total potential energies agree with theoretical predictions; and that the sphere diffuses at a rate consistent with the Stokes Einstein relation.

Figure 3.7: Finite Element Nanosphere of radius of 100 nm



3.3.2 Trajectory Analysis

The equation of motion for a rigid particle moving through a viscous background medium is given by:

$$m\frac{d\mathbf{v}}{dt} = -\eta\mathbf{v}. \tag{3.20}$$

Where m is the mass of the particle, $\mathbf{v}$ the velocity in space and, η the friction constant.

So that if the sphere as initial velocity $\mathbf{v}(0) = \mathbf{v_0}$:

$$\mathbf{v} = \mathbf{v_0}\exp(-\frac{\eta t}{m}) \tag{3.21}$$

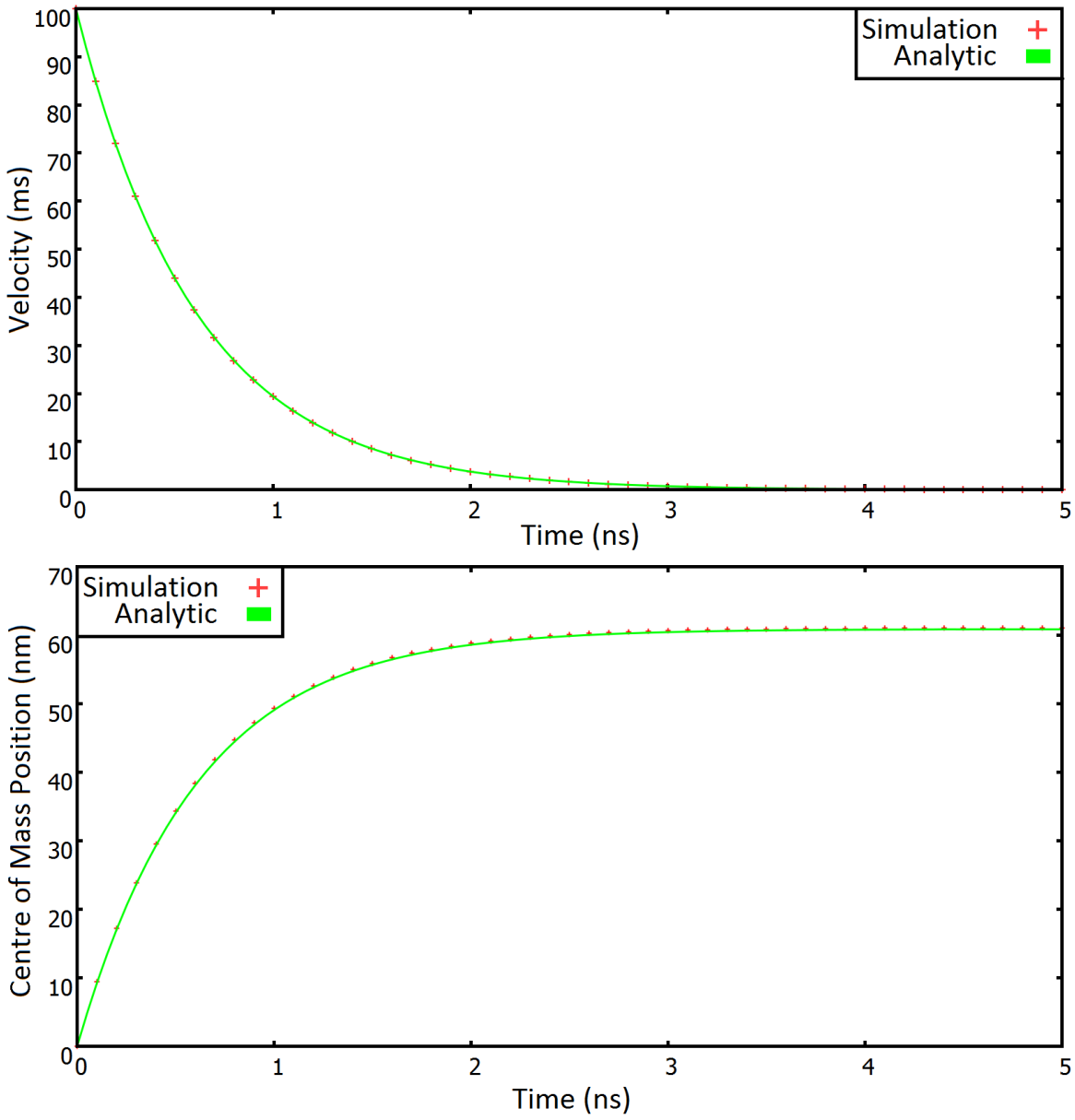
and the displacement from the initial position is given by:

$$\mathbf{x} = \frac{m\mathbf{v_0}}{\eta}(1 - \exp(-\frac{\eta t}{m})). \tag{3.22}$$

Equations (3.21) and (3.22) describe the velocity and position of a point like particle undergoing deceleration due to the viscous drag. To test that the simulation reproduces this behaviour, we use the sphere mesh shown in Figure 3.7 and place the same initial velocity on each node of the system. To parametrise the friction on each node of the system, we assume that the ball experiences a drag force of $6\pi\mu R\mathbf{v}$ from Stokes' drag in the surrounding fluid and divide this equally between the nodes of the system such that: $\eta = \frac{6\pi\mu R}{N}$ where N is the number of nodes and R is the radius of the sphere. The results of these simulations are compared against the analytical results in Figure 3.8.

The results from these simulations show accurate agreement between the theory and the simulations. Thus, the background fluid correctly retards the velocity of the particle.

Figure 3.8: Velocity (above) and position (below) as a function of time of a sphere of radius $100nm$ being retarded by an external solvent. The analytical result is shown in green with the simulated results in red.



3.3.3 Average Kinetic and Average Potential Energy Convergence

The inclusion of external friction means that we have added extra dissipation to the system and so a corresponding additional noise term is required by the fluctuation dissipation relation (2.53). We shall therefore retest the total average kinetic and potential energy of the system to ensure agreement with the equipartition theorem.

In order to calculate the average kinetic and average potential energy theoretically, we must calculate the number of degrees of freedom of the system. Inclusion of the force from the external fluid means that the system can now rotate and translate. This increases the number of kinetic degrees of freedom by 6 (3 translations degrees of freedom and 3 rotational) so that there are now a total of $3N$ kinetic degrees of freedom in the system where N is the number of nodes. Thus, the average kinetic energy is given by:

$$\langle \mathcal{E} \rangle = \frac{3Nk_BT}{2}. \tag{3.23}$$

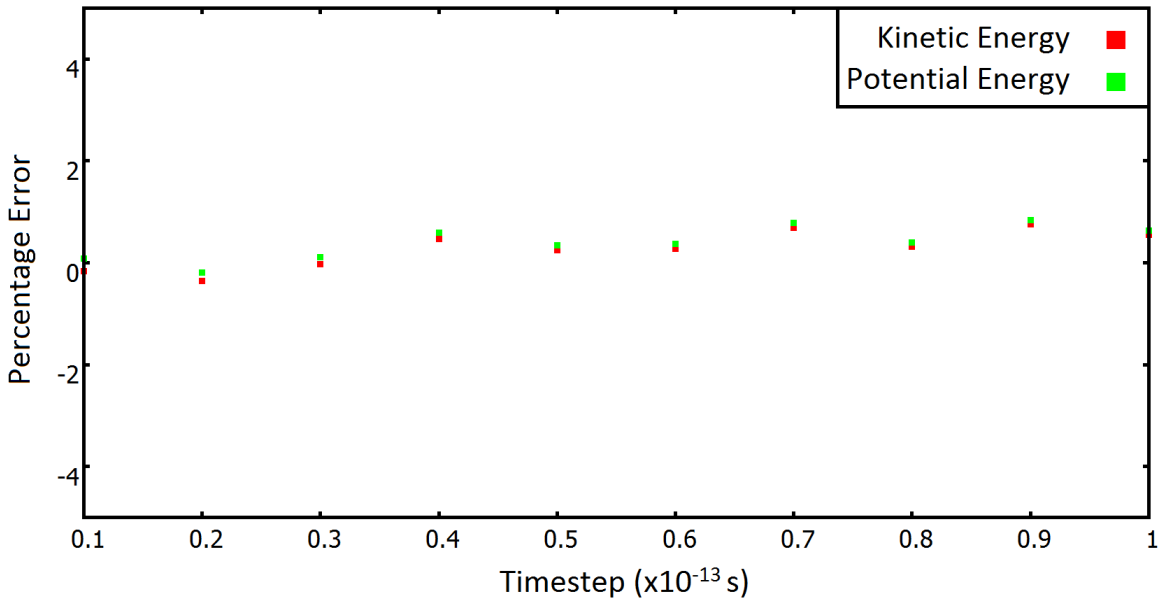
However, the potential energy is invariant under translation and rotation and thus the addition of translation motion and rotational motion does not effect the statistics of the potential energy. Thus the average potential energy remains given by:

$$\langle U \rangle = \frac{3(N-6)k_BT}{2}. \tag{3.24}$$

In order to test the average kinetic and potential energies numerically, long time simulations were run at a temperature of $300K$ for a variety of different time steps until the kinetic and potential energies converged to within half a percent accuracy. The results of these simulations are presented in Figure 3.9.

The average kinetic and potential energies are accurate within a fraction of a percent error for a time step spanning an order of magnitude. This confirms that the solution of the external noise is correct. The accuracy of these results reflects that the particle is

Figure 3.9: Convergence of the Kinetic energy (Red) and Potential Energy (Green) as a function of time of the sphere shown in Figure 3.7. Note the very high accuracy of the test regardless of time step.



essentially rigid and the shortest time scales in this simulation come from resolving the high frequency oscillations of the nodal positions not the external fluid dynamics. This has the effect of keeping the deformation of each of the elements in the system very small and thus the errors due to numerical time stepping in this simulation are also very small. Thus, the energies of the simulation are accurate regardless of the time step until the fast oscillations of the nodes are no longer resolved and the system becomes unstable.

3.3.4 Einstein Relation

The Einstein relation[71] relates the viscosity and average displacement of particles in a fluid as follows:

$$\langle \Delta x^2 \rangle = \frac{2k_B T}{\eta} t. \tag{3.25}$$

Here Δx is the displacement of the particle relative to its starting point in one dimen-

sion, η is the friction coefficient and t is time. This relation was revealed in Einstein's 1905 paper on Brownian Motion[71] and derived independently by Smoluchowski[72] and Sutherland[73] in the same year. For a spherical particle $\eta = 6\pi\mu R$ where R is the radius of the particle and μ the solvent viscosity.

In order to provide a further validation of the thermal noise and Stokes' drag, we have measured the diffusion of the the rigid sphere shown in Figure 3.7 at a temperature of $300K$ in 3 dimensions and recorded the position of the center of mass of the sphere as a function of time for $100ns$ averaged over 100 realizations.

As the sphere is a 3 dimensional particle and each axis is indistinguishable we can simply add the result of equation (3.25) 3 times to get the relation full 3 dimensional diffusion of the particle such that:

$$\langle \Delta x^2 + \Delta y^2 + \Delta z^2 \rangle = \frac{6k_B T}{\eta} t. \tag{3.26}$$

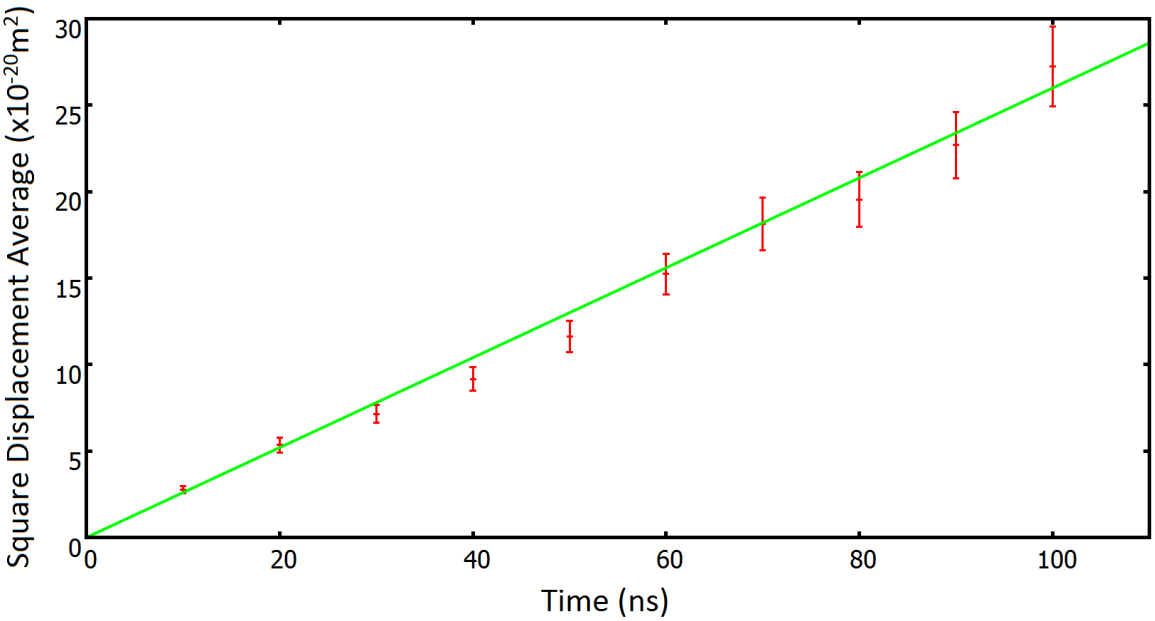
The theoretical results from equation (3.26) and simulation results are compared below in Figure 3.10.

The theoretical and analytical results are in agreement within the error bars. From this, we conclude that the external viscosity and corresponding thermal noises have been implemented correctly and yield the appropriate thermal physics.

3.4 Summary

In this chapter, I have demonstrated that the numerical algorithm with and without and external solvent conserves the correct thermal physics as well as the correct mechanical properties. I have also demonstrated both spatial and temporal convergence of the algorithm. From this, we conclude that the algorithm is functioning correctly and that the algorithm can be applied to biological problems. We can now apply this method to biological problems.

Figure 3.10: Diffusion of the sphere shown in Figure 3.7 against time. The theoretical result from 3.26 is shown in green against the observed average diffusion in red. The analytic and simulated data agree within error bars and thus confirm the external solvent is implemented correctly.



Chapter 4

Introduction to Biological Simulations

In this chapter, I describe the method used to take the mathematical and theoretical results from Chapters 2 and 3 and apply them to biological systems. In order to run a biological simulation utilising this method (Section 4.3), a finite element mesh of a biological molecule must be constructed (Section 4.1) and parametrised with appropriate values for the density, elasticity and viscosity (Section 4.2). This chapter discusses in detail how to construct a finite element mesh of a biological molecule from low resolution imaging data as well as appropriate values for the elasticity, viscosity and density of biological matter. For reference, a Fortran version of the stochastic finite element algorithm and the improved coarse graining algorithm are included in appendix A.4 and appendix A.5 respectively.

4.1 Finite Element Mesh

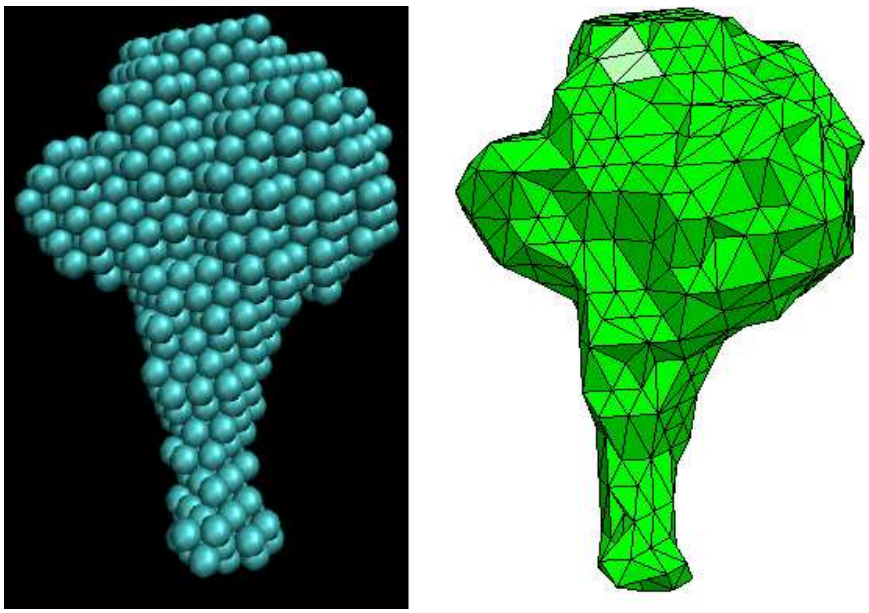
In order to construct a finite element mesh of a protein or other biological material, we are required to know the 3 dimensional shape of a biological molecule ab initio. There are many techniques for obtaining structures of biological molecules, a few examples being Nuclear Magnetic Resonance (NMR)[96], X-Ray crystallography[97], Cryo Electron

Microscopy (Cryo-EM)[98] and Small Angle X-Ray Scattering (SAXS)[99]. The structures obtained by these methods vary in resolution, for example X-Ray crystallography can yield structures with sub-angstrom resolution[100] that reveal protein secondary structure and atomic positions, whereas (Cryo-EM) yields structures to within a resolution of a few angstroms[101] and so can reveal the overall shape.

Unlike an atomistic simulation, a continuum approach can use low resolution imaging data as only the overall shape of the biomolecule is important from the continuum mechanical point of view, not the positioning of individual atoms. This is advantageous because even low resolution imaging data with no atomic information can be used to build a finite element mesh. However, there is a minimum length scale below which the continuum approach cannot probe. This length scale is on the order of several inter atomic distances and for simplicity can be thought of as approximately 5 Å. Below this length scale, the thermal fluctuations result in large strains that cause the numerical algorithm to become unstable unless a time step smaller than that of molecular dynamics is used. This length scale sets the absolute minimum length of any edge in a finite element mesh that represents a biomolecule. Furthermore, below this length the material will not behave like a continuum as it is intrinsically atomic. There are better simulation methods such as molecular dynamics capable of treating these small length scales with greater accuracy.

In this Section, I will describe how a finite element mesh for the 3 dimensional structure of a CoA ligase enzyme from the organism *Fusobacterium nucleatum* is obtained through Small Angle X-Ray Scattering (SAXS)[102]. The SAXS technique will be described in more detail in Chapter 5. For now only the fact that a 3 dimensional structure can be obtained from it is important. From this a mesh is generated with an edge length of 2.4Å (Section 4.1.1). As this is below the minimum length scale appropriate for this method, I have developed two different coarse-graining methods to increase the element size (Section 4.1.2 and Section 4.1.4).

Figure 4.1: Low resolution SAXS[102] envelope of a CoA Ligase enzyme[103] from *Fusobacterium nucleatum* with the initial finite element mesh on the right.

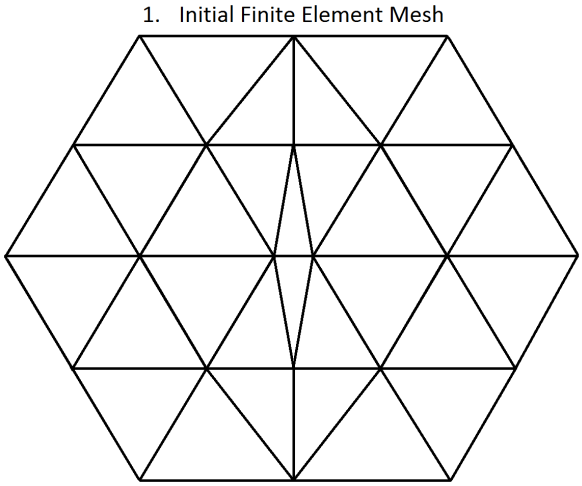


4.1.1 Small Angle Scattering Envelope to a Finite Element Mesh

A SAXS[102] envelope of the CoA ligase enzyme[103] is shown in Figure 4.1 with a homologous protein from Archaeoglobus fulgidus (PDB ID:3G7S[104][38]) together with the corresponding finite element mesh on the right. The SAXS data is used to define the envelope of the molecule as a cluster of spheres of radius 2.4 Å. This length corresponds to the shortest length scale probable in SAXS before the SAXS data becomes too noisy. By taking points on the outer surface of the sphere a finite element mesh can be constructed using generation software such as TETGEN[105] to give an initial finite element mesh of the system shown on the right in Figure 4.1.

While a simulation based on the initial finite element mesh is in principle possible, the length scale of this mesh (2.4 Å) is below the level for which a continuum approximation is valid. Resolving the thermal fluctuations of such small elements in practise requires a time step of order $0.1fs$, smaller than that required in MD simulations and is impractical. Thus, we are required to coarse grain this initial finite element mesh to

Figure 4.2: A general two dimensional surface mesh with one short edge.



a more suitable length scale to ensure the validity of the continuum approximation. I have devised two different coarse graining algorithms, one basic and one advanced, to tackle this problem.

4.1.2 Basic Coarse Graining Method

Coarse graining the finite element mesh involves reducing the total number of nodes and elements in the system whilst preserving the shape of the biomolecule as accurately as possible. To reduce the geometric complexity, we will coarse grain just the surface mesh and regenerate the volume mesh using standard mesh generation software.

In order to construct a coarse graining algorithm, we first consider how to coarse grain a simple test surface shown in Figure 4.2. Figure 4.2 shows a hypothetical section of a surface mesh with one short edge located at the centre. In order to coarse grain this mesh we must remove this short edge by removing one of the nodes. Given we can solve this problem we can then iterate the solution many times over a full surface mesh to arrive at a coarse grained surface mesh.

A solution to this problem is presented in Figure 4.3. The solution employs four steps as follows:

1. Construct the initial fine grained surface mesh.

2. Identify the shortest edge, in this case the two nodes **A** and **B** forming the edge **AB**.

3. Construct a new node **C** at the midpoint between nodes **A** and **B** and delete the two surface elements (Orange) that contain the edge **AB**.

4. Reconnect the surface mesh by merging nodes **A** and **B** into node **C**.

As shown in the final pane of Figure 4.3 this does indeed coarse grain the mesh and returns us to an initial mesh that, in principle, we could parse again with this algorithm.

This procedure can be iterated until the minimum edge length is above the desired threshold.

4.1.3 Problems of the Basic Coarse Graining Method

The basic method for coarse graining described in Section 4.1.2 preserves the shape of a planar surface. However, if the surface is curved there will be a small change to surface shape and hence the volume occupied by the biomolecule.

To illustrate this point, in Figure 4.4 we consider the change in the volume below the surface as a consequence of implementing one iteration of the coarse graining algorithm. There is a small reduction in the volume below the surface caused by the curvature of the mesh, as shown in Figure 4.5.

The basic coarse graining algorithm causes volume loss when the surface is locally convex and volume gain when it is locally concave. While each step only produces a small change in the volume the cumulative effect can produce a significant change in the shape of the biomolecule. Therefore it would be desirable to have a coarse-grainer that preserves the volume of the biomolecule at each iteration.

Figure 4.3: Simple Algorithm for coarse graining a surface mesh.

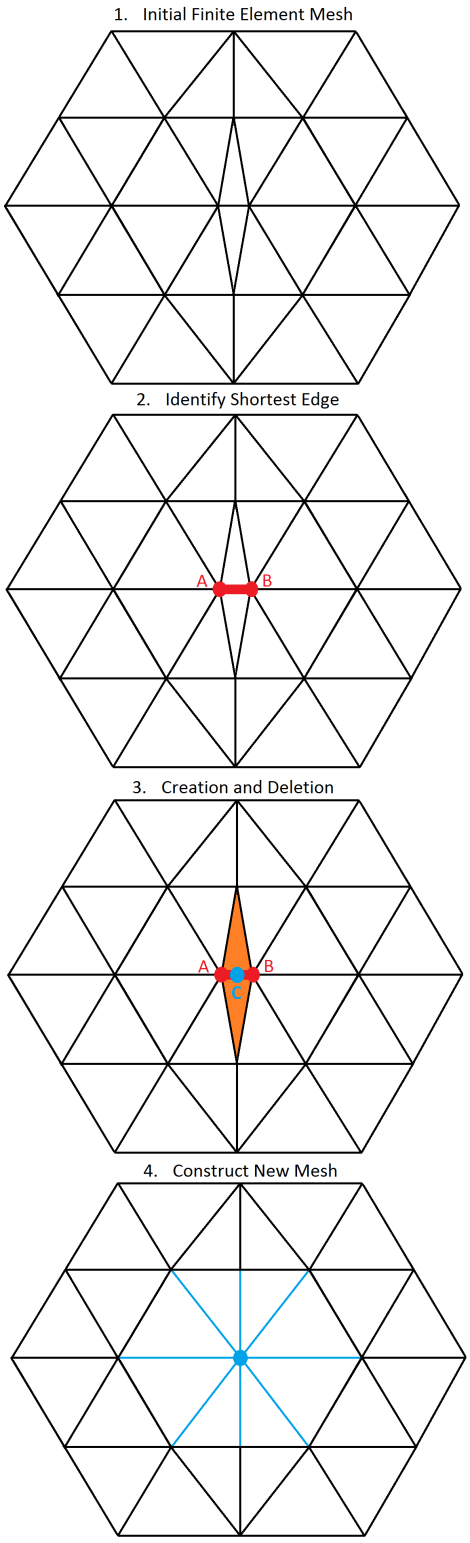


Figure 4.4: 3D view of the basic coarse graining algorithm.

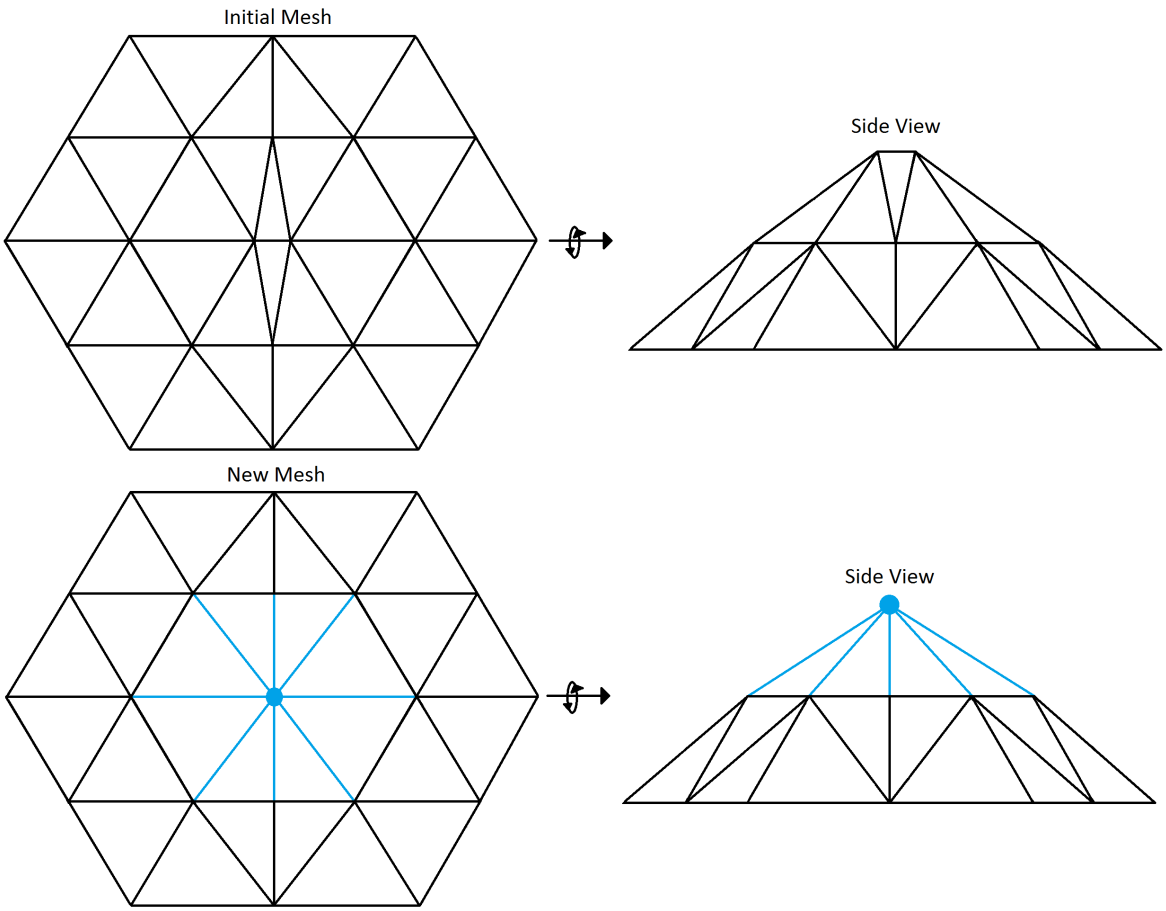
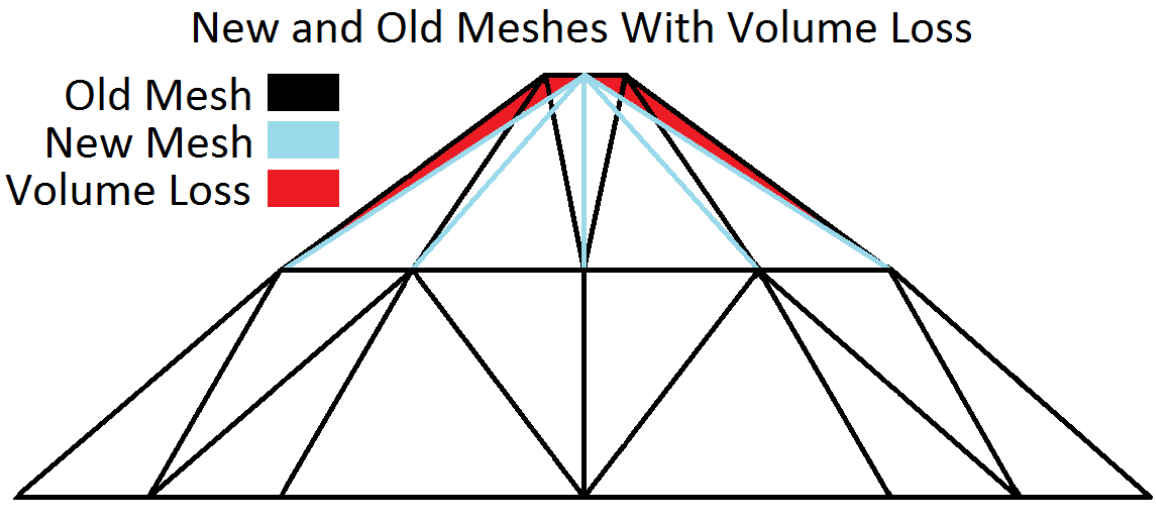


Figure 4.5: Volume Loss.



4.1.4 Improved Coarse Grainer

In order to design a coarse grainer that conserves volume we shall modify the basic algorithm presented in Section (4.1.2) so that no net volume is lost or gained as a consequence of removing the node. An algorithm for this is presented in Figure 4.6 and 4.7. The difference from the previous algorithm is that the position of the new node, termed **C**, is determined so that there is no volume change. The algorithm is as follows and a complete Fortran 90 version of the code is included in Appendix A.5:

1. Construct an initial surface mesh.
2. Identify the shortest edge, in this case the two nodes **A** and **B** forming the edge **AB**.
3. Find all the elements that contain either node **A** or **B**.
4. Calculate the outward pointing unit normals **N1** and **N2** that belong to the two elements containing nodes **A** and **B**.
5. Construct two new nodes **C** and **D** on the line through the midpoint of **A** and **B** directed parallel to the vector **N** defined as the average of the unit normals **N1** and **N2**.
6. Construct a local volume mesh by connecting the surface elements containing nodes **A** or **B** to node **D** and the equivalent volume mesh using new node **C** instead of **A** and **B**.
7. Calculate the total volumes of the two volume meshes constructed in part 6 and move new node **C** along the vector **N** to ensure that the volume of both volume meshes is equal.
8. Construct the new surface mesh using new node **C** to ensure no volume loss.

In the algorithm described above, we have ignored the specific placement of nodes **C** and **D**. There are two useful scaling lengths that are found in the initial surface mesh

Figure 4.6: Improved Coarse Grained Algorithm Part 1.

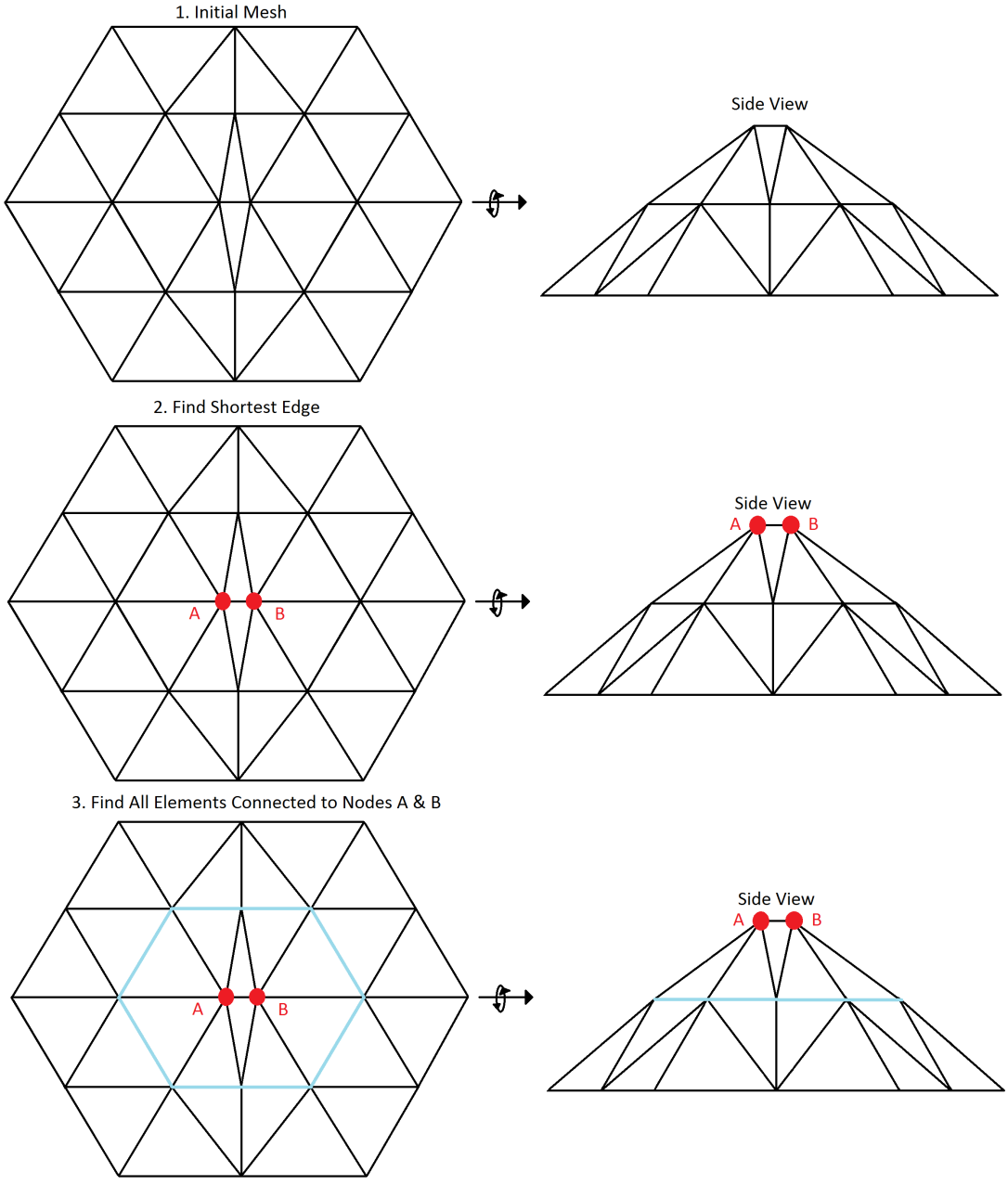


Figure 4.7: Improved Coarse Grained Algorithm Part 2.

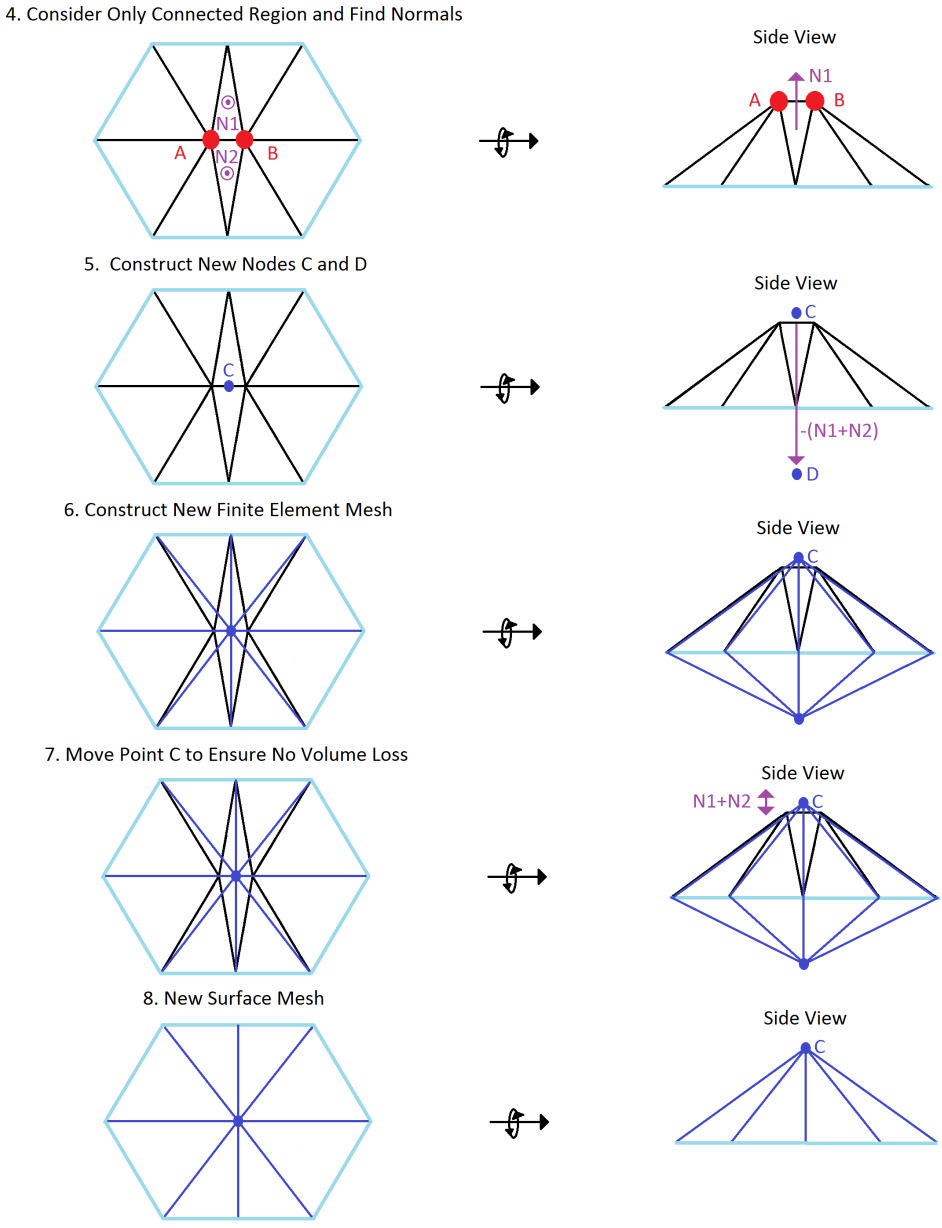
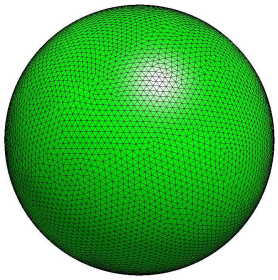


Figure 4.8: Example sphere that will be coarse grained using the basic and advanced method.



the longest edge length and the length of the edge between nodes **A** and **B**. The node **D** is placed 10 times the longest edge length in the initial surface mesh away from the midpoint of nodes **A** and **B**. This avoids any clashes between volume elements such as self intersection. Node **C** is placed utilising the length between nodes **A** and **B** to scale the average normal vector **N**. This provides a sensible initial location for node **C** that will be close to the point where no volume change takes place. The point **C** can then be moved along the vector **N** using a simple convergence routine based on the changes in volume.

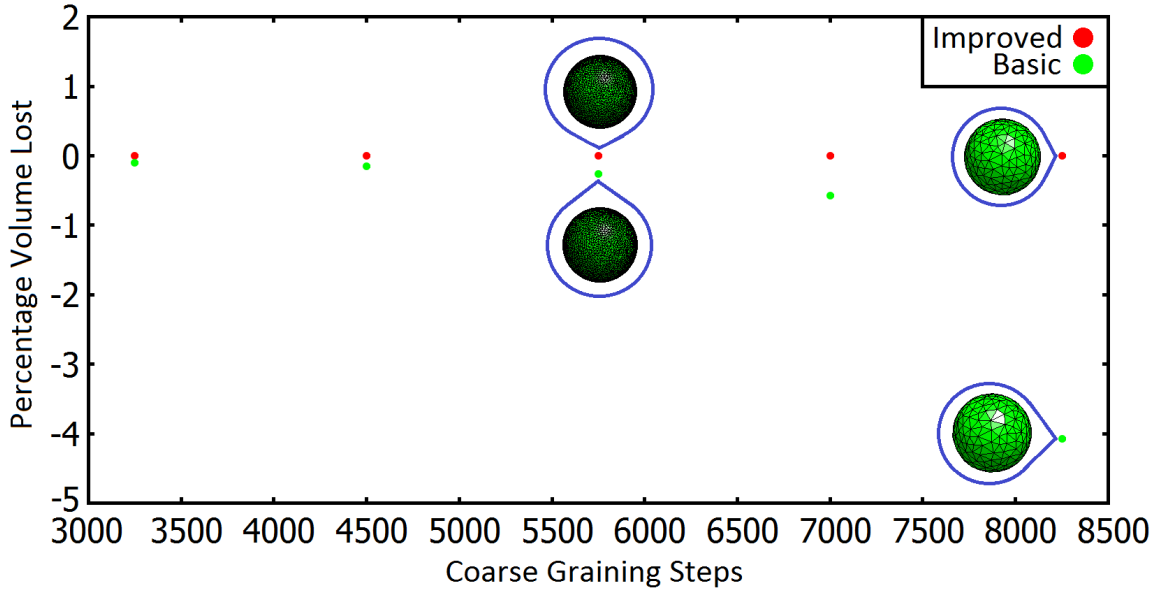
The principle behind this method is to place the new node **C**, not at the average position of **A** and **B**, but at a nearby point such that the reconnection produces no change in volume. This is achieved by constructing a local volume mesh constructed from the tetrahedra formed from the surface elements.

4.1.5 Performance of the Improved Coarse Grainer and the Basic Coarse Grainer

As a simple test to show the relative performances of the two coarse graining algorithms, a sphere shown in Figure 4.8 was coarse grained using both coarse grainers for a varying number of steps.

The results of these simulations is shown in Figure 4.9 where we see that the basic algorithm loses up to 5 percent of the volume of the sphere while the improved algorithm

Figure 4.9: Volume loss as a function of coarse graining step.



is accurate to machine precision for the volume of a sphere.

4.2 Material Parameters for Proteins / Biological Matter

In order to simulate the motion of biomolecules we require estimates for the material properties of biomolecules. While some continuum properties such as density are easily determined, others are much harder. In this section, we discuss the existing literature and justify the range of values we use.

4.2.1 Experimental Techniques to Probe the Material Properties of Biomolecules

Here I will elaborate on the precise nature of the material properties used in the continuum model. There are a few experimental techniques that have been used to probe the material properties of biomolecules such as the internal viscosity (Section 4.2.3), the

Young's modulus (Section 4.2.4) and density (Section 4.2.2). These include solvent based methods[106][107] and Atomic Force Microscopy (AFM)[108][109] techniques to measure the internal viscosity of proteins as well as surface force apparatus[110], nanoindentation[111] AFM techniques and sound wave experiments[112] to measure the Young's Modulus. Of these methods the most reliable measurement of the internal viscosity and Young's modulus comes from solvent based methods and nanoindentation AFM techniques respectively. The reasons for this will be discussed in Sections (4.2.3) and (4.2.4).

4.2.2 Biomolecule Density

The density of biomolecules is easy to calculate as the masses of atoms are known and there are many complete crystal structures in the protein database (PDB)[38]. On average the density of biomolecules is around $1500 kgm^{-3}$ or 1.5 times that of water[113][114].

4.2.3 Biomolecule Internal Viscosity

There are two experimental techniques used to probe the internal viscosity of proteins. These are solvent based techniques[106][107] and AFM[108][109]. The solvent based techniques work by measuring the effect of the solvent viscosity on the unfolding / refolding rates and extrapolating the effect of the internal viscosity. AFM techniques work by measuring the relaxation time of a protein by either oscillating the AFM tip across a frequency range[108] or by deforming a protein and then retracting the AFM cantilever while monitoring the deflection of the tip[109].

The solvent based experiments[106] model protein folding as a diffusive process over energy barriers. The method works by assuming that there are two contributions to the diffusion coefficient over the folding energy barrier, namely the solvent viscosity and the internal viscosity of the protein[106]. Thus, if one can alter the solvent viscosity, while maintaining the shape of the free energy surface, and measure the protein relaxation rate between folding and unfolding. Then the internal viscosity can be extrapolated

using a result from Ansari et al[115].

In the oscillating tip AFM experiments[108] the protein is attached to a gold film and a gold nano particle by tethers. The tip can then be oscillated back and forth at different amplitudes and frequencies to probe various regimes of viscoelastic response. An alternative to tethering the protein and thus any chemical sequence modification is to dry proteins onto a film. Once the tip finds a protein on the film, the force indentation curve can be measured. This allows one to extract a Young's modulus (See Section 4.2.4) by compressing the protein. The AFM tip can then be retracted. However, as the tip is effectively attached to the protein at this point, a viscous response[109] is observed that allows the internal viscosity of the protein to be calculated using a Maxwell element[109] to model the tip deflection.

These two methods yield radically different values for the internal viscosity with solvent based methods yielding values on the order of $10^{-3}Pas$ and the AFM techniques $10^{4}Pas$. The reason for these differences is that the solvent and AFM techniques probe different relaxation time scales. The AFM techniques use either an oscillating tip with a frequency on the order of a few kHz[108] and thus measure millisecond relaxation times or measure the time taken for the AFM tip to relax (hundreds of microseconds)[109]. By contrast, the solvent based techniques intrinsically depend on the relaxation time of the solvent which is a much faster time scale of either of the AFM techniques[106]. Hence the higher viscosities found in AFM based measurements result from all the modes with relaxation times less than a few milliseconds, while the solvent based methods probe much shorter time scales on which the longer relaxation modes behave elastically. Given that my simulations have run times on the order of microseconds, much shorter than those of the AFM measurements, the solvent based measurements provide a more appropriate value for the internal viscosity of biomolecules in the continuum model.

Table 4.1: Young's modulus and Poisson ratio for a variety of different materials.

Material	Young's Modulus	Poisson ratio
Rubber[116]	1-10 MPa	0.48-0.50
Low Density Polyethylene[117]	200 MPa	-
High Density Polyethylene[118]	800 MPa	0.46
Steel[116]	190-210 GPa	0.27-0.30

4.2.4 Biomolecule Young's Modulus and Poisson Ratio

The Young's modulus of a material describes how much stress builds up in a material according to a given strain[83]. Therefore, the Young's modulus is defined as:

$$E = \frac{\sigma}{\epsilon}. \tag{4.1}$$

The Young's modulus is a useful quantity because it is normalised for the dimension of a material and the applied deformation. Thus, it is straightforward to compare results of materials with different dimensions undergoing different deformations.

The Poisson ratio[83] is best explained by the following thought experiment, consider a cube that is compressed in the x-axis. This cube will shorten in the x-axis and typically expand in the y and z axis. The Poisson ratio ν describes the ratio of compression in the transverse axes (y and z) to the axial compression in the x-axis. As such the Poisson ratio is defined by:

$$\nu = -\frac{d\epsilon_{trans}}{d\epsilon_{axial}}. \tag{4.2}$$

Where ϵ_{trans} is the strain in the y and z axes and ϵ_{axial} is the strain in the x-axis. The Poisson ratio varies from -1 to 0.5, where materials with a Poisson ratio of 0.5 are incompressible.

Before I discuss how these two elastic moduli relate to biomaterials, a list of typical values of the Poisson ratio and Young's modulus for common materials is provided in table 4.1 to provide context for the values derived for biomolecules.

Table 4.2: Young's Modulus for different globular proteins.

Globular Protein	Young's Modulus
Lysozyme[109]	300-700 MPa
Bovine carbonic anhydrase II[120]	75 MPa
Lactate Oxidase[121]	500-800 MPa

Of the methods listed in Section 4.2.1 that measure the Young's modulus of biomolecules, only the nanoindentation[111] AFM method yields values that are appropriate for the continuum model. The reason for this is that the nanoindentation AFM method measures the Young's modulus of a single[111] globular protein by using compressive indentation to measure the resultant force as a function of the depth of the indentation. This indentation can be modelled purely elastically using the Hertz contact model and the Young's modulus extracted[111]. It is required that the indentation be fully reversible so that the protein is not damaged by the experimental process. The other methods listed in Section 4.2.1 measure the Young's modulus of either multiple proteins in a crystalline phase[112] or long fibrous (non globular) proteins such as actin[119] tracks making the extracted Young's moduli incompatible with the continuum model.

It may seem contradictory that we use the nanoindentation AFM technique to measure the Young's modulus while rejecting the other AFM methods to measure the internal viscosity. The reason for this is that the viscoelastic response of globular proteins to deformation is complex and that the AFM techniques probe different physics. The internal viscosities derived from AFM explore relaxation time scales much longer than any of my simulation run times and are thus inappropriate[108][109]. However, the nanoindentation technique measures only the elastic response of a protein due to deformation[111]. Thus, the Young's modulus obtained from nanoindentation is directly applicable to the continuum model because the continuum model should yield the same behaviour under the same conditions.

The observed values for the Young's modulus using the nano indentation AFM technique for globular proteins are given in table 4.2.

Based on these experimental values of the Young's Modulus, we conclude that a reasonable range for the Young's modulus of globular proteins is probably on the order of 100-1000 MPa. This range correlates well with what we would expect from table 4.1 in that proteins are at the most basic level polymer chains folded in a specific way due to hydrogen bonding, hydrophobicity, electrostatics and Van der Waals. So it should come as no surprise that the Young's moduli compare favourably with those of low and high density polyethylene.

The Poisson ratio is more difficult to measure experimentally in part due to the difficulty of measuring the exact volume of proteins undergoing deformations. However it has been estimated to be between 0.3 and 0.4[122].

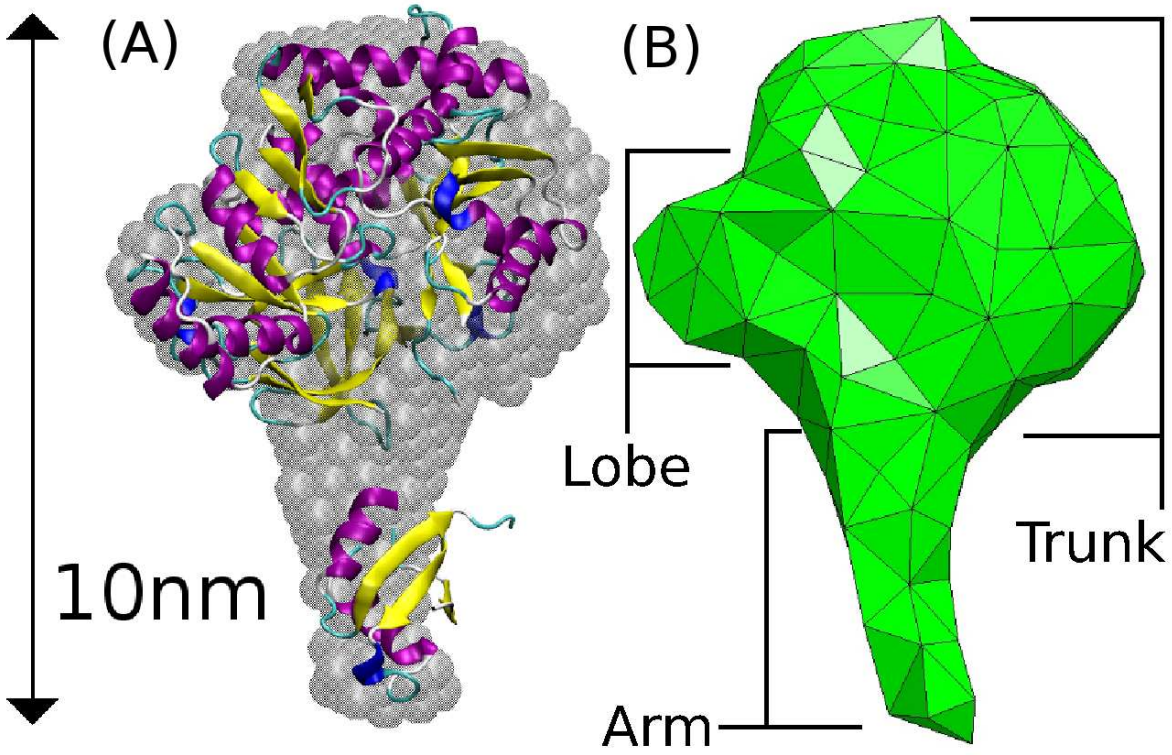
4.3 Application to Biology

I now demonstrate the use of the continuum model to probe the conformational flexibility of a globular protein using the first order element approximation to improve computational efficiency. As a representative system, we have used the long fatty acid chain CoA ligase[103] enzyme from the organism *Fusobacterium nucleatum*. To date, it has not been possible to obtain atomically detailed structural information for this protein. However, the overall 3-dimensional shape of the biomolecule has been determined using Small Angle X-ray Scattering (SAXS)[102]. Figure 4.10(a) shows the atomistic structure of the homologous protein long-chain-fatty-acid-CoA ligase from Archaeoglobus Fulgidus (PDB ID: 3G7S[104]), with the SAXS structural envelope of the CoA ligase superimposed. The experimentally determined structural envelope was converted into a finite element mesh using TETGEN[105], which was then further refined using NETGEN[123] and the basic coarse graining method described in Section 4.1.2.

The resulting mesh is compared with the original SAXS structural envelope in Figure 4.10(b).

For these calculations, I tested the model using three values of the Young's modulus

Figure 4.10: Comparison of the SAXS[102] structural envelope and atomistic structure of the homologue CoA ligase[103][104] from Archaeoglobus Fulgidus with the equivalent finite element mesh viewed in NETGEN[123].



corresponding to low ($450MPa$), medium ($560MPa$) and high ($800MPa$) biomolecular flexibility. Additional simulations were run with Young's moduli at $340MPa$ and $100MPa$ however these proved to be unstable unless an exceptionally small time step, smaller than that of atomistic molecular dynamics was used. This made producing long time simulations impractical for these values of the Young's modulus. The reason for this is that the finite element mesh of CoA ligase is on the lower bound of the continuum approximation. At this length scale thermal fluctuations dominate and make the numerical algorithm unstable. Higher Young's moduli help to compensate for numerical instability making it more difficult for elements to deform significantly. The remaining material properties of the biomacromolecule were assigned based on the existing literature values quoted for proteins, where available see Section (4.2). The density was set to be $1500kgm^{-3}$. I have assumed a value that corresponds to the shear viscosity of water, $10^{-3}Pas$. The same value was used for the bulk viscosity[124]. The temperature was set to $300K$ and the Poisson ratio set to 0.4. The bulk and shear moduli can then be calculated from the following, where E is the Young's Modulus and ν the Poisson ratio:

$$G = \frac{E}{2(1+\nu)} \tag{4.3}$$

$$K = \frac{E}{3(1-2\nu)} \tag{4.4}$$

Prior to finite element discretisation, the governing equation for the protein model is given below in Equation (4.5):

$$\rho\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right) = \frac{\partial \sigma^v_{ij}}{\partial x_j} + \frac{\partial \sigma^e_{ij}}{\partial x_j} + \frac{\partial \sigma^t_{ij}}{\partial x_j}. \tag{4.5}$$

Using the governing Equation (4.5), with stress free boundary conditions and the mesh shown in Figure 4.10(b), each simulation was run for $500ns$ for the three different choices of material parameters. Each calculation took approximately 2 weeks

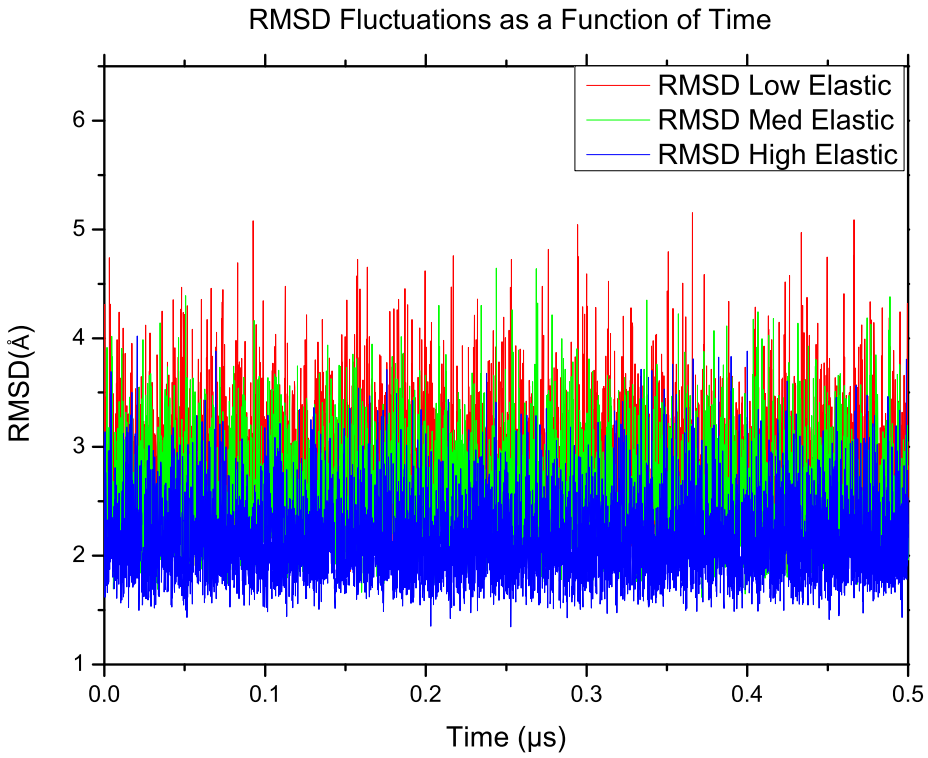
of CPU time on a single processor. The simulation trajectories were visualised using paraFEM[125]. On visualising the trajectories, it was apparent that the molecule changes its orientation relative to the starting structure, whilst conserving angular momentum, as the fluctuations in the shape of the biomolecule cause the inertia tensor to change[126]. Therefore, the trajectories were post-processed to reorientate the molecule prior to analysis. The biomolecular flexibility was quantified by calculating the Root Mean Squared Deviation (RMSD) of the co-ordinates of the mesh nodes from their initial values during the simulations, as shown in Figure 4.11.

As expected, increasing the Young's modulus from $450MPa$ (red line) to $800MPa$ (blue line) results in a smaller RMSD from the initial structure, indicating a less flexible protein. The value of (2Å) obtained lies within the range 1 to 6Å found from $10ns$ atomistic MD simulations of small proteins selected from the protein data bank[127]. In these MD simulations the magnitude of the thermal fluctuations about the native state of the protein was measured by calculating the RMSD. As the proteins in the MD simulations did not refold into a different state, both the finite element simulations and MD simulations should yield comparable results for the RMSD. Other areas of the simulation literature also support this conclusion as well as the protein literature in general[128] [129] [130].

The stochastic finite element model simulations provide a series of conformers of the protein as it undergoes thermal fluctuations that are analogous to conventional particle-based molecular dynamics, but now at the continuum level. Figure 4.12 shows 9 representative conformations of the protein extracted from the stochastic finite element model simulation trajectories performed with the lowest Young's Modulus ($450MPa$).

Each of the 9 snapshots are coloured by their overall deviation from the equilibrium configuration with the scale bar showing the displacement in nanometers. Figure 9 shows that the "trunk" of the protein is relatively immobile, and undergoes minor structural disruptions while retaining its overall shape. However, the lobe located to the left of the biomolecule moves more significantly than other regions of the trunk

Figure 4.11: RMSD obtained for three different sets of elastic parameters (with rotations removed prior to analysis).



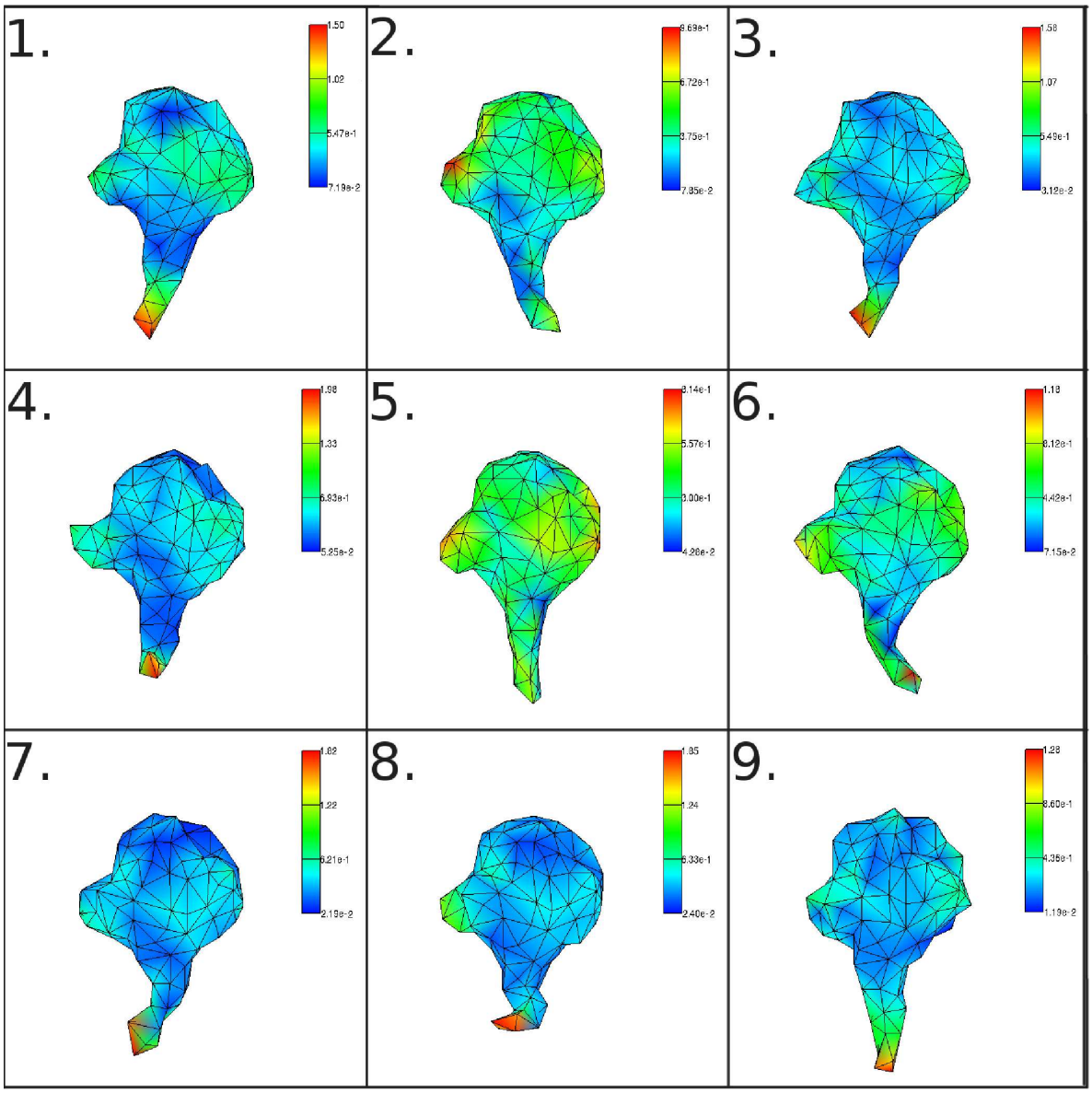


Figure 4.12: Nine representative conformers CoA ligase sampled from stochastic finite element model simulations with $E = 450MPa$: 1. Arm swings left. 2. Small thermal disruptions to the entire trunk, including lobe. 3. Large arm swing to the left and lobe disruption. 4. Arm sticks out of the page and disruption to the leftward lobe. 5. Entire protein elongated by the thermal noise. 6. Arm swings to the right. 7. Arm swings to the left and major disruption to the shape of the arm. 8. Arm swings out of the page. 9. Elongation with change in shape of the entire trunk.

as it is considerably thinner than the main body of the protein. The most striking deformations occur in the "arm" at the base of the enzyme, which is highly flexible and swings back and forth around the bottom of the molecule during the course of the simulation. This indicates that the intermediate region between the arm and the trunk acts as a flexible hinge region in the biomolecule. It is interesting to note from Figure 4.10(a) that it is precisely in the region that stochastic finite element model predicts should be of greatest flexibility that the homologous protein Archaeoglobus Fulgidus has missing electron density, indicating that this region was too mobile for its structure to be determined crystallographically.

Chapter 5

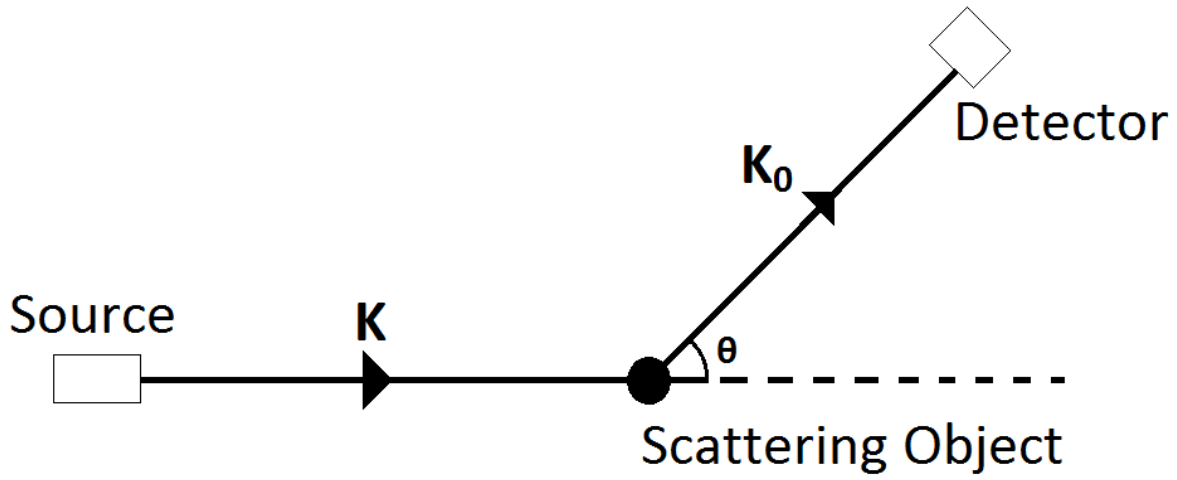
Finite Element Simulations of Small Angle X-ray Scattering Experiments

In this chapter, I describe the application of the finite element model to Small Angle X-ray Scattering (SAXS) experiments, extending the work done in Chapter 4. Here I use the static SAXS envelope to construct the finite element mesh and then use the finite element model to generate dynamics for the structure (Section 5.3). The dynamics of the X-ray scattering signal can then be generated and compared against the experimentally determined X-ray scattering curve (Section 5.4 and Section 5.5).

5.1 Scattering Theory

The X-ray scattering signal can be determined from the shape of a protein. Scattering theory describes how a wave interacts with matter via processes such as collision, interference or diffraction[131]. These effects produce scattering patterns that contain structural information about the object which scatters the wave. Throughout this chapter, we are exclusively concerned with elastic scattering, the mechanism relevant to SAXS[132]. The theory of elastic scattering[131][133] is developed in Section 5.1.1

Figure 5.1: Construction of a scattering experiment.



for point like particles and extended to continuum materials[133] in Section 5.1.3 before introducing the idea of form and structure factors[133] in Section 5.1.4.

5.1.1 Elastic Scattering Theory

In elastic scattering, the kinetic energy of the incident wave is conserved during scattering[131]. A typical scattering experiment is illustrated in Figure 5.1. The source emits a wave, such as X-rays, that interact with the scattering object. This produces a scattering signal that we consider to be a radial wave emitted by the scattering object[133]. The amplitude of this scattered wave is a function of the scattering angle θ. The variations in this amplitude as a function of θ lead to patterns in the scattering signal that reveal information about the structure of the scattering object.

The incoming beam is given by[133]:

$$A = A_0 \exp(i\mathbf{k} \cdot \mathbf{x}), \tag{5.1}$$

where $|\mathbf{k}| = \frac{2\pi}{\lambda}$ is the wavenumber for waves of length λ.

The scattered radiation pattern is formed from the superposition of the waves from each object[133]. We consider n identical particles whose dimension is much smaller

than the wavelength of the incoming wave. The interaction between the incoming beam and particle m at position $\mathbf{r}_m$ will then produce a radially symmetric scattered wave:

$$A_m = \frac{A_0 b_m}{R} \exp(i\mathbf{k} \cdot \mathbf{r}_m) \exp(i\mathbf{k}_0 \cdot (\mathbf{x} - \mathbf{r}_m)). \tag{5.2}$$

The first exponential in equation (5.2) comes from evaluating equation (5.1) at the position of the scattering particle while the second exponential gives the phase factor of the radially emitted wave. The pre-factor $\frac{b_m}{R}$ arises from the reduction in amplitude with distance for a spherical wave, where R is the distance between the scattering source and the detector, and b_m is called the scattering length[133]. Equation (5.2) can be re-arranged by introducing a new variable $\mathbf{q} = \mathbf{k} - \mathbf{k_0}$ such that:

$$A_m = \frac{A_0 b_m}{R} \exp(i\mathbf{r}_m \cdot \mathbf{q}) \exp(i\mathbf{k}_0 \cdot \mathbf{x}). \tag{5.3}$$

We note that:

$$|\mathbf{q}| = \frac{4\pi}{\lambda} \sin\left(\frac{\theta}{2}\right). \tag{5.4}$$

Where θ is the angle between the incident and scattered beams.

From equation (5.3), we can now calculate the total amplitude of the resultant scattered wave by summing over all of the scattering particles:

$$A = \frac{A_0}{R} \exp(i\mathbf{k}_0 \cdot \mathbf{x}) \sum_{m=1}^{n} b_m \exp(i\mathbf{q} \cdot \mathbf{r_m}). \tag{5.5}$$

In experiments one does not measure the amplitude of the scattered beam, but the intensity[133] $I = AA^*$ where A^* denotes complex conjugate. Thus, the intensity is given by:

$$
\begin{aligned}
I &= \frac{A_0^2}{R^2} \sum_{m=1}^{n} b_m \exp(i\mathbf{q} \cdot \mathbf{r}_m) \sum_{l=1}^{n} b_l \exp(-i\mathbf{q} \cdot \mathbf{r}_l), \\
&= \frac{A_0^2}{R^2} \sum_{l,m} b_m b_l \exp(i\mathbf{q} \cdot (\mathbf{r}_m - \mathbf{r}_l)). \qquad (5.6)
\end{aligned}
$$

Equation (5.6) relates the intrinsic position of the particles in the scattering experiment shown in Figures 5.1 and 5.2 to the observed intensity.

5.1.2 Absolute Intensity

The intensity I detected at radius R from the scattering material depends upon both the amount of material and the distance between the material and the detector. It is therefore useful to normalise the intensity so that the effects of distance and the amount of material are scaled out[133]. This gives the absolute intensity I_{abs}, defined as follows:

$$
I_{abs} = \frac{IR^2}{I_0 V}, \qquad (5.7)
$$

where $I_0 = A_0 A_0^*$ is the intensity of the incident radiation and V is the volume of the scattering source.

By substitution of equation (5.6) into equation (5.7) this is the quantity generally reported by experimentalists:

$$
I_{abs} = \frac{1}{V} \sum_{l,m} b_m b_l \exp(i\mathbf{q} \cdot (\mathbf{r}_m - \mathbf{r}_l)). \qquad (5.8)
$$

5.1.3 Continuum materials

We now consider the scattering from a continuum material. We modify equation (5.2) to incorporate the number density of particles $\rho(\mathbf{r})$ in a specific small scattering volume ΔV[133] such that:

$$A_{\Delta V} = \frac{A_0 b}{R}\rho(\mathbf{r})\Delta V \exp(i\mathbf{k}\cdot\mathbf{r}_m)\exp(i\mathbf{k}_0\cdot(\mathbf{x}-\mathbf{r}_m)). \tag{5.9}$$

The derivation now proceeds in a similar manner to that presented in Section 5.1, but instead of summing over all particles we sum over all volumes to calculate the total scattering amplitude[133] $A = \sum_{\Delta V\, in\, V} A_{\Delta V}$ so that the total scattering amplitude:

$$A = \sum_{\Delta V\, in\, V} A_{\Delta V}\frac{A_0 b}{R}\rho(\mathbf{r})\Delta V \exp(i\mathbf{k}\cdot\mathbf{r}_m)\exp(i\mathbf{k}_0\cdot(\mathbf{x}-\mathbf{r}_m)), \tag{5.10}$$

and in the limit that ΔV tends to 0 the summation can be replaced by an integral:

$$A = \frac{A_0 b}{R}\exp(i\mathbf{k}\cdot\mathbf{x})\int_V dV\rho(\mathbf{r})\exp(i\mathbf{q}\cdot\mathbf{r}). \tag{5.11}$$

Once again we now convert to measuring the intensity of the scattered beam by computing $I = AA^*$ so that:

$$\begin{aligned} I &= \frac{A_0^2 b^2}{R^2}\int_V dV\rho(\mathbf{r})\exp(i\mathbf{q}\cdot\mathbf{r})\int_{V'} dV'\rho(\mathbf{r}')\exp(-i\mathbf{q}\cdot\mathbf{r}') \\ &= \frac{A_0^2 b^2}{R^2}\int_V dV\int_{V'} dV'\rho(\mathbf{r})\rho(\mathbf{r}')\exp(i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}')). \end{aligned} \tag{5.12}$$

Equation (5.12) gives the intensity in terms of the pair distance between any two infinitesimal volumes of a continuum solid. Once again we must normalise equation (5.12) to account for the experimental setup as in Section 5.1.2 using equation (5.7) to yield:

$$I_{abs} = \frac{b^2}{V}\int_V dV\int_{V'} dV'\rho(\mathbf{r})\rho(\mathbf{r}')\exp(i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}')). \tag{5.13}$$

Equation (5.13)[133] defines the absolute intensity and completes the derivation of the form of the intensity in terms of the vector $\mathbf{q}$. We will now examine what these formulas can tell us about the structure of the scattering object. This will lead to two important terms that describe different aspects of the object structure: The form factor

and structure factor.

5.1.4 Form Factors and Structure Factors

We now consider scattering from a set of identical objects with some large scale spatial arrangement. In this case, there are two major contributions to the overall scattering pattern. The two contributions are from the global arrangement of the objects and second from the internal structure of the objects themselves. The scattering from the global arrangement is called the structure factor[134] $S(\mathbf{q})$ while the scattering from the internal structure of the objects is called the form factor[131] $P(\mathbf{q})$. In order to see the origin of these two terms we consider the Fourier transform of the scattering density:

$$\widetilde{\rho}(\mathbf{q}) = \int d^3\mathbf{r}\rho(\mathbf{r}) \exp(i\mathbf{q}\cdot\mathbf{r}). \tag{5.14}$$

We consider a density $\rho(\mathbf{r})$ where a constant background density ρ_0 is perturbed locally by some additional objects, with centres of mass given by $\mathbf{r}_\alpha$ (see Figure 5.2). The total density of the system[134] is given by:

$$\rho(\mathbf{r}) = \rho_0 + \sum_\alpha \rho_p(\mathbf{r} - \mathbf{r}_\alpha). \tag{5.15}$$

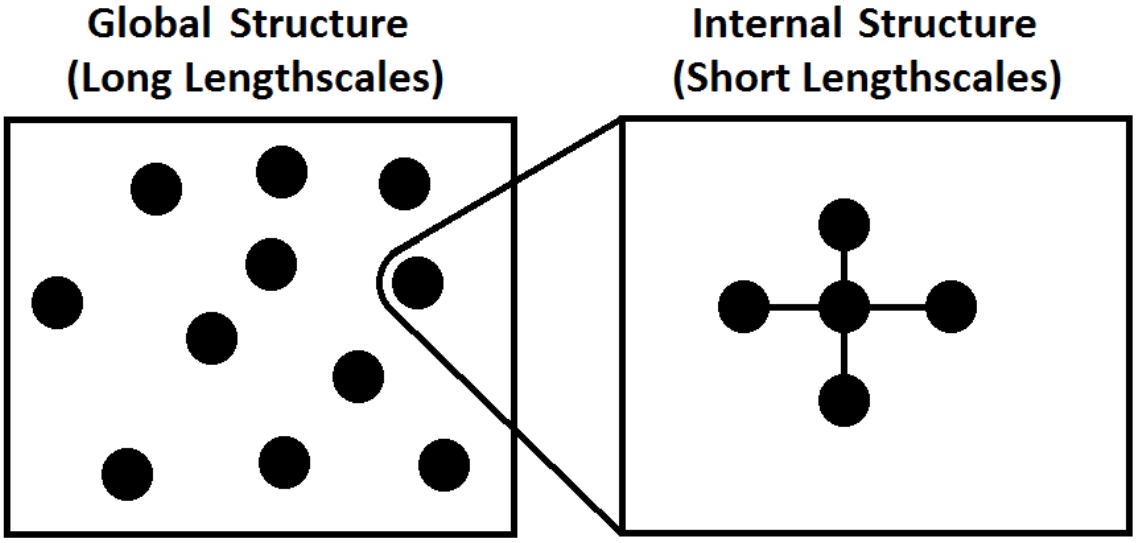
where $\rho_p(\mathbf{r} - \mathbf{r}_\alpha)$ is the density from the object centred at $\mathbf{r}_\alpha$.

Substitution into equation (5.14) to yields:

$$\begin{aligned} \widetilde{\rho}(\mathbf{q}) &= \int d^3\mathbf{r} \left(\rho_0 + \sum_\alpha \rho_p(\mathbf{r} - \mathbf{r}_\alpha) \right) \exp(i\mathbf{q}\cdot\mathbf{r}) \\ &= \sum_\alpha \int d^3\mathbf{r}\rho_p(\mathbf{r} - \mathbf{r}_\alpha) \exp(i\mathbf{q}\cdot\mathbf{r}) \end{aligned} \tag{5.16}$$

and, making the substitution $\mathbf{r}' = \mathbf{r} - \mathbf{r}_\alpha$:

Figure 5.2: Diagram of number density variation at different length scales. Clearly, at long length scales there is an overall density of the system given by ρ_0 that is perturbed locally by the internal structure.



$$\widetilde{\rho}(\mathbf{q}) = \sum_{\alpha} \int d^3\mathbf{r}' \rho_p(\mathbf{r}') \exp(i\mathbf{q}\cdot\mathbf{r}') \exp(i\mathbf{q}\cdot\mathbf{r}_\alpha) \tag{5.17}$$

$$= \widetilde{\rho}_p(\mathbf{q}) \sum_{\alpha} \exp(i\mathbf{q}\cdot\mathbf{r}_\alpha). \tag{5.18}$$

The scattering intensity is then proportional to $\widetilde{\rho}(\mathbf{q})\widetilde{\rho}^*(\mathbf{q})$ such that:

$$I = \widetilde{\rho}_p(\mathbf{q})\widetilde{\rho}_p^*(\mathbf{q}) \sum_{\alpha\beta} \exp(i\mathbf{q}\cdot(\mathbf{r}_\alpha - \mathbf{r}_\beta)) \tag{5.19}$$

From equation (5.19), we identify the structure factor $S(\mathbf{q})$ and the form factor[134] $P(\mathbf{q})$ as follows:

$$S(\mathbf{q}) = \sum_{\alpha\,\beta} \exp(i\mathbf{q}\cdot(\mathbf{r}_\alpha - \mathbf{r}_\beta)), \tag{5.20}$$

$$P(\mathbf{q}) = \widetilde{\rho}_p(\mathbf{q})\widetilde{\rho}_p^*(\mathbf{q}). \tag{5.21}$$

The structure factor $S(\mathbf{q})$ is defined purely by the relative distances between the centre of masses of the objects and contains information about the overall placement of objects in the bulk volume of the material. In contrast the form factor[131][134] $P(\mathbf{q})$ depends only on the local perturbations in the scattering density due to the structure of an individual object. This is an example of convolution in real space becoming a multiplication of two functions in Fourier space. The difference between the form factor and structure factor[134] will be important when we discuss the nature of SAXS experiments to identify the shape of biomolecules in Section 5.2.

5.2 Small Angle Scattering Experimental Procedure and Envelope Construction

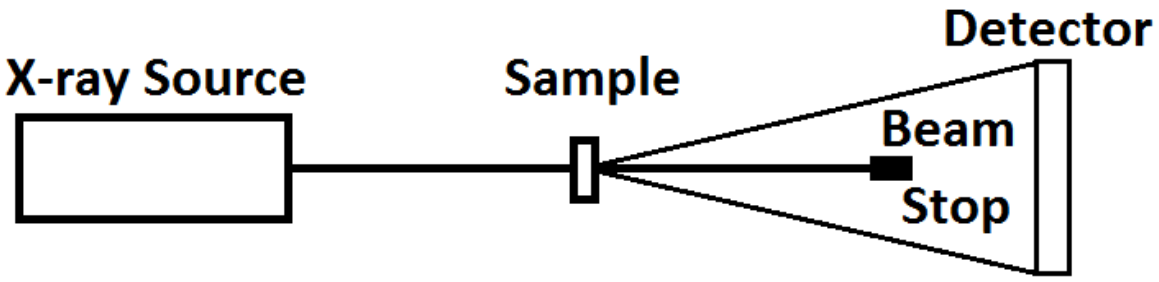
SAXS experiments determine the small angle X-ray scattering curve from a particular biomolecule. The small angle scattering curve contains information on the overall shape of biomolecules, not the location of individual atoms or protein secondary structure. From this a low resolution model 3D structure of the biomolecule can be generated (Section 5.2.3) called a SAXS envelope. The primary advantage of the SAXS technique is that all experiments can be run in solution with small sample volumes[102]. This avoids the need to grow crystals of biological molecules as is required by higher resolution techniques such as X-ray crystallography. In the following sections I introduce how a SAXS experiment is run (Section 5.2.1), spherical averaging of the X-ray scattering signal (Section 5.2.2) and construction of the SAXS envelope (Section 5.2.3).

5.2.1 Experimental Procedure

An example of the basic construction of a SAXS experiment is shown in Figure 5.3.

The experiments are run on dilute protein solutions[102]. The consequence of this is that the proteins are randomly distributed in space and do not aggregate. This means that there is no global structure to the proteins as they are randomly arranged

Figure 5.3: Experimental setup of an X-ray beam line.



in space and therefore the structure factor is flat. Secondly, as the proteins are free to translate and rotate in solution, the scattering signal is a population average of all protein conformations in all orientations[135]. Therefore, there is a need to average our X-ray scattering equations over all orientations detailed in Section 5.2.2.

5.2.2 Spherical Averaging of the X-Ray Scattering Signal

In a SAXS experiment there are two sources of scattering from the buffer and from the dissolved biomolecules. The effect of the buffer can be subtracted out by simply sampling the scattering pattern of the buffer without any biomolecules present[102]. Thus, what remains is simply the signal from the remaining biomolecules in solution. The solutions are assumed to be sufficiently dilute such that cross scattering between different biomolecules can be neglected[102]. Consequently, the scattering is due to the form factor not the structure factor.

The biomolecules are undergoing Brownian motion in the fluid and are thus free to rotate and translate in solution as well as explore their free energy landscape due to thermal fluctuations. Consequently, the scattering signal from the solution is a population average over all conformations and orientations of the biomolecule. This requires us to average our scattering equations over all orientations[131], so starting from equation (5.13) the scattering from a continuum material is given by:

$$I_{abs}(\mathbf{q}) = \frac{b^2}{V}\int_V dV \int_{V'} dV' \rho(\mathbf{r})\rho(\mathbf{r}')\exp(i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}')). \tag{5.22}$$

We start by averaging the absolute intensity about all orientations, so functions of $\mathbf{r}$ are unaffected, to yield:

$$\langle I_{abs}(|\mathbf{q}|)\rangle = \frac{b^2}{V}\int_V dV \int_{V'} dV' \rho(\mathbf{r})\rho(\mathbf{r}')\langle\exp(i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}'))\rangle. \tag{5.23}$$

The important quantity in equation (5.23) is $\langle\exp(i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}'))\rangle$. We can interpret this factor as the contribution to the absolute scattering from each pair of respective infinitesimal volumes dV and dV'. The average of $\langle\exp(i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}'))\rangle$ can be expressed as:

$$\langle\exp(i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}'))\rangle = \frac{1}{4\pi}\int_0^{2\pi} d\phi \int_0^{\pi} d\theta \sin\theta \exp(i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}')) \tag{5.24}$$

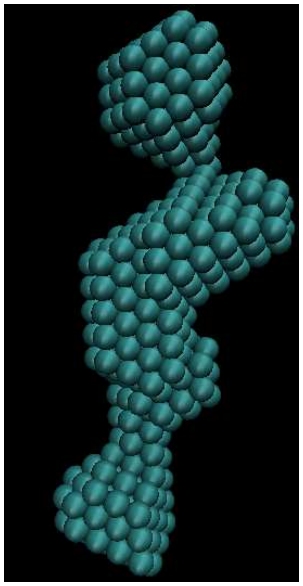
Equation (5.24) can be integrated analytically by choosing spherical coordinates about the direction of the vector $\mathbf{q}$. Thus, we integrate equation (5.24) as follows:

$$\begin{aligned}
\langle\exp(i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}'))\rangle &= \frac{1}{4\pi}\int_0^{2\pi} d\phi \int_0^{\pi} d\theta \sin\theta \exp(i\cos\theta|\mathbf{q}||\mathbf{r}-\mathbf{r}'|) \\
&= \frac{i}{4\pi}\int_0^{2\pi} d\phi \frac{\exp(-i|\mathbf{q}||\mathbf{r}-\mathbf{r}'|)}{|\mathbf{q}||\mathbf{r}-\mathbf{r}'|} - \frac{\exp(i|\mathbf{q}||\mathbf{r}-\mathbf{r}'|)}{|\mathbf{q}||\mathbf{r}-\mathbf{r}'|} \\
&= \frac{\sin(|\mathbf{q}||\mathbf{r}-\mathbf{r}'|)}{|\mathbf{q}||\mathbf{r}-\mathbf{r}'|}.
\end{aligned} \tag{5.25}$$

Substituting this expression in equation (5.25), in SAXS the average absolute scattering[131] is given by:

$$\langle I_{abs}(|\mathbf{q}|)\rangle = \frac{b^2}{V}\int_V dV \int_{V'} dV' \rho(\mathbf{r})\rho(\mathbf{r}')\frac{\sin(|\mathbf{q}||\mathbf{r}-\mathbf{r}'|)}{|\mathbf{q}||\mathbf{r}-\mathbf{r}'|}. \tag{5.26}$$

Figure 5.4: SAXS envelope of protein 'Tom1l1'[138] with sphere radius 2.4Å.



5.3 Small Angle Scattering Envelope Construction

Equation (5.26) relates the SAXS scattering to a known density distribution $\rho(\mathbf{r})$. However, the objective of the experiment is to infer the shape of the molecule from the scattering function. We define an envelope[136] for a biomolecule E as being the set of points $\mathbf{r}$ that lie inside it so that:

$$\rho(\mathbf{r}) = \begin{cases} \rho_0 & \mathbf{r} \in E \\ 0 & \mathbf{r} \notin E. \end{cases} \tag{5.27}$$

where ρ_0 is the average density of the biomolecule.

The objective is to find the form of the envelope E that reproduces the observed value of $I_{abs}(|\mathbf{q}|)$; this is referred to as the SAXS envelope. This type of problem is referred to as an inverse geometric problem and is ill-posed. However, a software package called 'ATSAS'[137] has been developed to solve this problem. ATSAS attempts to construct the SAXS envelope as a set of spheres on a lattice. An example is shown in Figure 5.4.

5.3.1 ATSAS Package

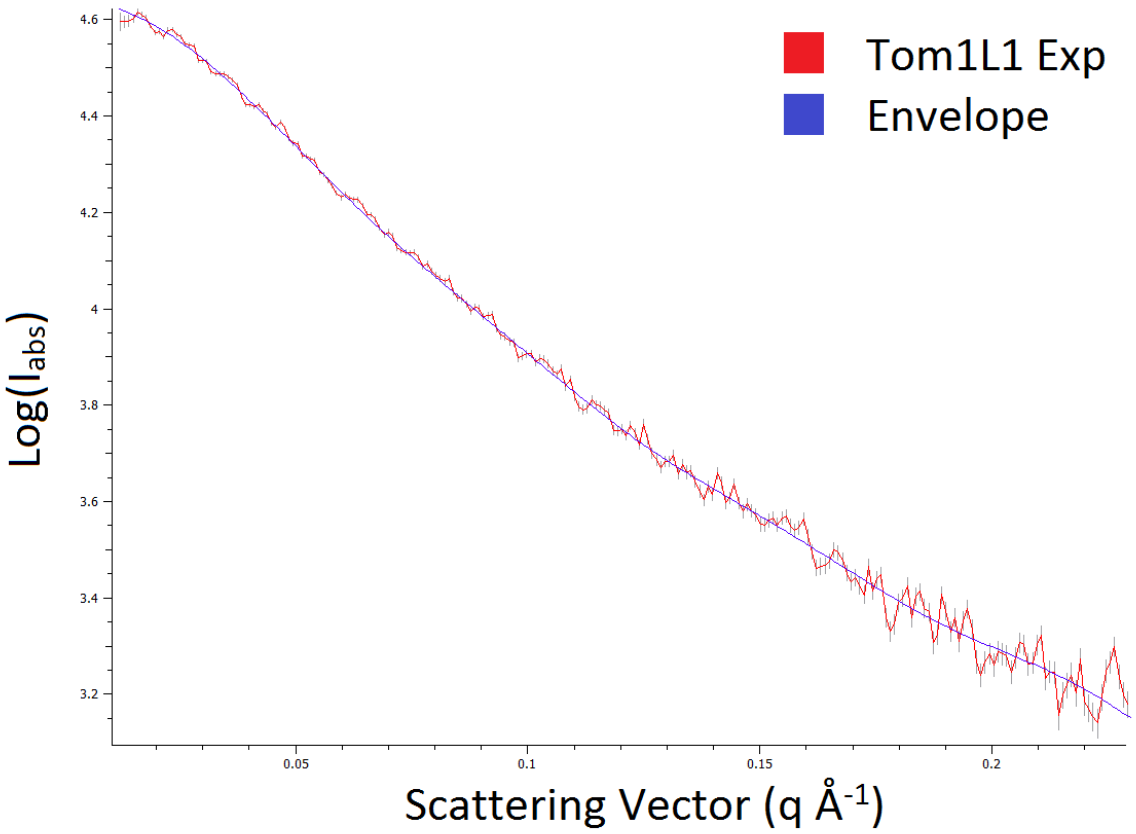
The ATSAS[137] package contains two important routines DAMMIF[139] and DAMAVER[140]. DAMMIF[139] takes an X-ray scattering curve and utilises simulated annealing, with the dummy atom model shown in Figure 5.4, to build a 3D structure. At each step of the DAMMIF[139] algorithm, the X-ray scattering curve from the current set of spheres is calculated. The calculated signal is then compared with the experimental curve. The structure is then mutated by adding or removing spheres to improve the fit, via simulated annealing to minimise the error in scattering.

As this simulated annealing only finds the local minima, in general there are many different solutions of packed spheres that will yield an X-ray scattering curve that fits with the experimental data within the tolerance required. Thus, the recommended practise is to generate many different models for the same X-ray scattering curve by changing the random number seed in the DAMMIF[139] input. These models are then averaged together to produce a consensus model using the DAMAVER[140] routine. The X-ray scattering curves from the consensus model shown in Figure 5.4 of Tom1L1 are shown in Figure 5.5 to demonstrate the use of this technique. The SAXS model shown in Figure 5.4 was produced at the Hauptmann Woodward Medical institute by Thomas Grant[138].

5.4 Simulations of Tom1L1 Using the Finite Element Model

In the previous sections, we have built an understanding of elastic scattering theory, the fundamentals of SAXS experiments and how one can generate a low resolution 3D structural model of the biomolecule Tom1L1[138]. There are potential problems with this structural model due to the fact that the X-ray scattering signal is a population average over all conformers and that the initial data is inherently noisy. The SAXS model of Tom1L1 is a static model of what is in reality a dynamic structure[135].

Figure 5.5: Experimental X-ray scattering curve of Tom1L1[138] (Red) compared against the model envelope generated by use of DAMMIF[139] and DAMAVER[140] (Blue). As shown the fit from the consensus model (Figure 5.4) tracks the experimental signal well visualised in PRIMUS[141].



In this Section, we will use the continuum finite element model to put the dynamics back into the model structure (Section 5.4.3) and then compare the experimentally determined data against the modelled / dynamical data (Section 5.4.4). In the next subsection we discuss the calculation of the X-ray scattering curve from a finite element mesh (Section 5.4.1) before the conversion of the spheres model of Tom1L1 to a finite element model along with quality control (Section 5.4.2).

5.4.1 Calculation of the X-ray Scattering from a Finite Element Mesh

From equation (5.26) the scattering integral is given by:

$$\langle I_{abs} \rangle = \frac{b^2}{V} \int_V dV \int_{V'} dV' \rho(\mathbf{r}) \rho(\mathbf{r}') \frac{\sin(|\mathbf{q}||\mathbf{r} - \mathbf{r}'|)}{|\mathbf{q}||\mathbf{r} - \mathbf{r}'|}. \tag{5.28}$$

For a volume that is divided up into n finite elements equation (5.28) becomes a double sum over all elements as follows:

$$\langle I_{abs} \rangle = \frac{b^2}{V} \sum_{i=1}^{n} \sum_{j=1}^{n} \int_{V_i} dV_i \int_{V'_j} dV'_j \rho_i(\mathbf{r}) \rho_j(\mathbf{r}') \frac{\sin(|\mathbf{q}||\mathbf{r}_i - \mathbf{r'}_j|)}{|\mathbf{q}||\mathbf{r}_i - \mathbf{r'}_j|}. \tag{5.29}$$

Where V_i is the volume of the i^{th} finite element.

To evaluate these integrals it is natural to transform the volume integrals to those of a right angled tetrahedron. We assume initially that the density in the system is uniform, however the local density will change as finite elements deform. From conservation of mass the local density satisfies $\rho \det(J) = \rho_0 \det(J_0)$, Where the subscript 0 represents the initial configuration of an element and J is the Jacobian that describes the transform of an element to the right-angled tetrahedron reference element. Thus, equation (5.29) can be re-written as:

$$\langle I_{abs} \rangle = \frac{\rho_0^2 b^2}{V} \sum_{i=1}^{n} \sum_{j=1}^{n} \int_{V_i} dV_i \int_{V'_j} dV'_j \frac{\det(J_{0i})}{\det(J_i)} \frac{\det(J'_{0j})}{\det(J'_j)} \frac{\sin(|\mathbf{q}||\mathbf{r}_i - \mathbf{r'}_j|)}{|\mathbf{q}||\mathbf{r}_i - \mathbf{r'}_j|}. \tag{5.30}$$

The two volume integrals can then be evaluated numerically using quadrature. To do this we utilise the fact that the vector $\mathbf{r}$ can be expressed in terms of the finite element shape function of linear elements[77] in this case:

$$\mathbf{r} = (1 - s - t - u)\mathbf{x}_1 + (s)\mathbf{x}_2 + (t)\mathbf{x}_3 + (u)\mathbf{x}_4. \tag{5.31}$$

Where $\mathbf{x}_1$ to $\mathbf{x}_4$ represent the nodal positions of a particular finite element.

This completes our discussion of the calculation of the X-ray scattering curve from a finite element model. We now move onto the construction of the finite element mesh and quality control.

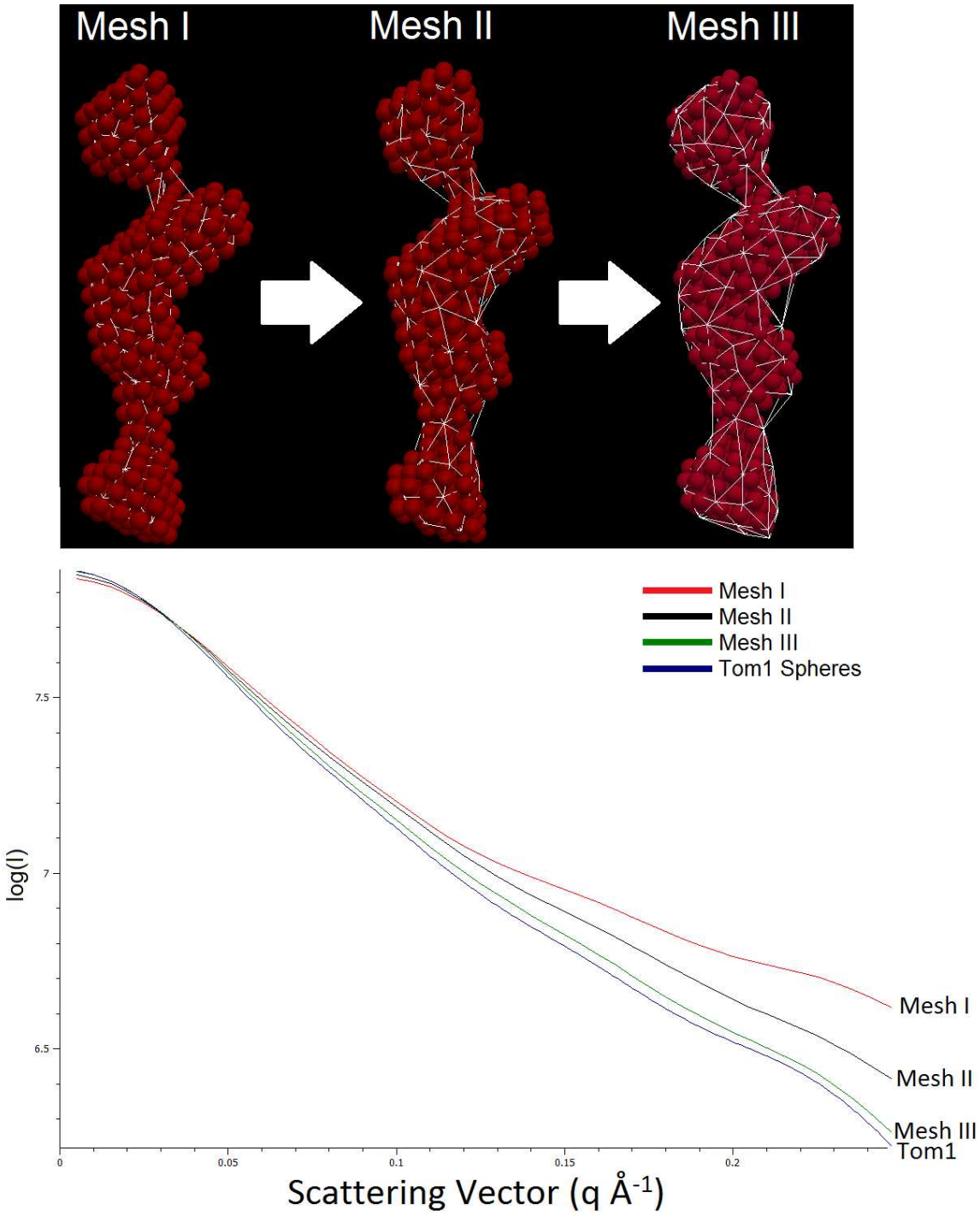
5.4.2 Construction of the Finite Element Mesh for Tom1L1

As briefly discussed in Chapter 4, the finite element mesh is constructed by utilising the initial SAXS envelope (See Figure 5.4). In order to capture the full volume of the envelope extra points are placed on the edges of the spheres and then an initial mesh is generated using TETGEN. This initial mesh is then coarse-grained by using the advanced coarse-graining algorithm described in Chapter 4. As a check, we calculate the X-ray scattering curve from the finite element mesh, to ensure that we have not distorted the shape in the process of coarse-graining the finite element mesh.

In order to demonstrate the effects that coarse-graining can have on the SAXS scattering, I have produced 3 finite element meshes from the scattering envelope for Tom1L1[138]. The meshes are termed Mesh I, Mesh II and Mesh III. Mesh I was produced using the basic coarse graining with nodes placed only at the centre of each sphere of the Tom1L1 SAXS envelope. Mesh II was produced using the basic coarse grainer with nodes places at the edges of the spheres in the SAXS envelope to better capture the surface. Mesh III was produced using the advanced coarse grainer with nodes placed on the surface of the SAXS envelope. The three meshes and their corresponding X-ray scattering curves are shown in Figure 5.6.

Figure 5.6 demonstrates the problems with the basic coarse graining method and

Figure 5.6: Top: Tom1L1[138] SAXS envelope shown in red with the corresponding finite element mesh shown as a wireframe in white. Below: The X-ray scattering curves from the 3 meshes compared against the scattering from the Tom1L1 SAXS envelope visualised in PRIMUS[141].



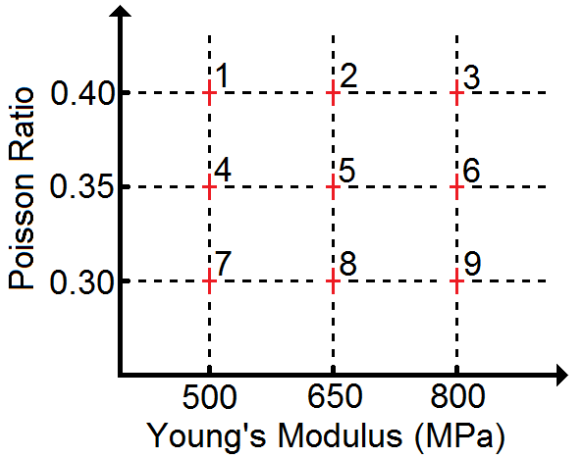
volume loss. It is clear that meshes I and II do not track the volume and shape of the Tom1L1 mesh well and this is reflected in the shape of the X-ray scattering curve. While Mesh II does show improvement, the scattering curves from mesh I and II differ significantly from that of the SAXS envelope. By contrast, Mesh III using the advanced coarse grainer is a much better fit to the Tom1L1 SAXS scattering curve especially for low values of $\mathbf{q}$ that correspond to long length scales. This makes sense as with coarse graining we desire to retain the same long length scale information while coarse graining over the short length scales.

We conclude from this that Mesh III gives a sufficiently similarly X-ray scattering curve to be a reasonable model of Tom1L1.

One question that remains unresolved is how this coarse grainer compares to already existing software available for meshing. As we have already used Netgen, I shall compare the performance of the Netgen's ability to coarse grain compared to the coarse grainer I have written. Netgen contains a variety of options for constructing a volume mesh of various degrees granularity ranging from very coarse to very fine with the definitions changing properties of what qualifies as an acceptable finite element for the mesh. Typically, the very coarse option will accept poorer elements than the very fine option and so the very fine option will produce a mesh containing more elements than the very fine option.

We can now perform a test, if we use the initial surface mesh that was coarse grained to construct Mesh III in Figure 5.6 and task Netgen to construct a volume mesh using the very fine and very coarse granularity options Netgen returns a Mesh with 4243 elements in either case. By contrast, Mesh III contains 425 elements. Thus, it would appear that for the Tom1L1 starting mesh the coarse graining options within Netgen do not have any affect for this particular mesh and secondly Netgen makes no attempt to coarse grain the surface mesh. Consequently, the edge lengths of the Netgen coarse grained mesh are all of order 2Å and thus too small to run a stochastic finite elment simulation on. While the finite element mesh coarse grained by Netgen will have a

Figure 5.7: Combinations of Young's moduli and Poisson ratios used in the simulations shown as red crosses.



X-ray scattering curve that better matches the Tom1L1 SAXS envelope than Mesh III this is because in effect no coarse graining has taken place in Netgen as the surface mesh is taken as a given and the volume mesh constructed around it.

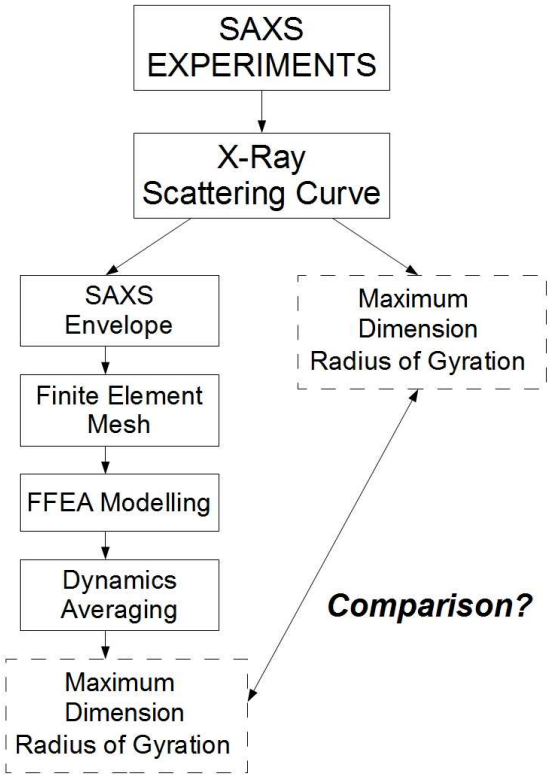
5.4.3 Simulations of Tom1L1

Using Mesh III we performed simulations of the dynamics of Tom1L1. The density, temperature and viscous parameters were set to be $1500Kgm^{-3}$, $300K$ and $10^{-3}Pas$ respectively, while the elastic parameters were different in each simulation.

As discussed in Section 4.2.4 the Young's modulus of biomolecules is expected to be around 600 MPa, and so in common with out studies in chapter 4 we choose the Young's modulus to be between $500MPa$ and $800MPa$ and Poisson ratios between 0.3 and 0.4. Figure 5.7 displays the combination of Young's moduli and Poisson ratios used giving nine sets of parameters.

Each simulation was run for a total of $500ns$ taking a total of 28 days on a single core processor. The primary objective of these simulations was to see the effect on the X-ray scattering curve, radius of gyration and the maximum dimension of Tom1L1 due to the elastic parameters.

Figure 5.8: Overall scheme of work. FFEA stands for Fluctuating Finite Element Analysis and means application of the model to the Tom1L1 SAXS envelope.



5.5 Results

In this section, we shall consider the results from the 9 simulations of Tom1L1. The overall scheme of work is shown in Figure 5.8. First the overall observed dynamics in section 5.5.1, the averaged X-ray scattering curve is discussed in section 5.5.2, the effect of the Young's modulus in section 5.5.3 and the effect of the Poisson ratio in section 5.5.4.

5.5.1 Observed Dynamics

We shall begin by considering the results from simulation 1. Qualitatively the dynamics of the other simulations are similar as only the maximal extent of the motions are affected by the different choice a elastic parameters. Nine different conformers are shown in Figure 5.9 from a total of 50 conformers extracted at 10 nanosecond intervals, to ensure statistical independence between conformers.

In general, the dynamics in the first simulation shows that the primary mode is a pincer movement wherein Lobes 1 and 2 are brought against the body then back again. This is best shown in the 17^{th} and 27^{th} conformer. Lobes 1 and 2 are not necessarily correlated in their motions and are in general independent of one another as shown in conformers 5 and 14. Other motions of Tom1L1 include overall extensions shown in conformers 15 and 41.

5.5.2 Dynamically Averaged X-Ray Scattering Curve

To produce the average X-ray scattering curve for each of the 9 sets of material parameters, a total of 50 conformers are selected from the dynamics every 10ns as discussed above. The X-ray scattering curve for each of these 50 conformers are then calculated and averaged to produce a time averaged signal from the simulation (See Figure 5.10).

When the X-ray scattering curves of the nine averages are compared against the experimental X-ray scattering curve shown in turquoise or the SAXS envelope shown

Figure 5.9: A total of nine conformers of Tom1L1 including the rest state are displayed along with the X-ray scattering curve from each conformer. The green scattering curves belong to the static structure shown in the top left and the brown scattering curves the deformed structure shown below. The colours represent the overall displacement of the rest structure to the static structure shown in the figure legend.

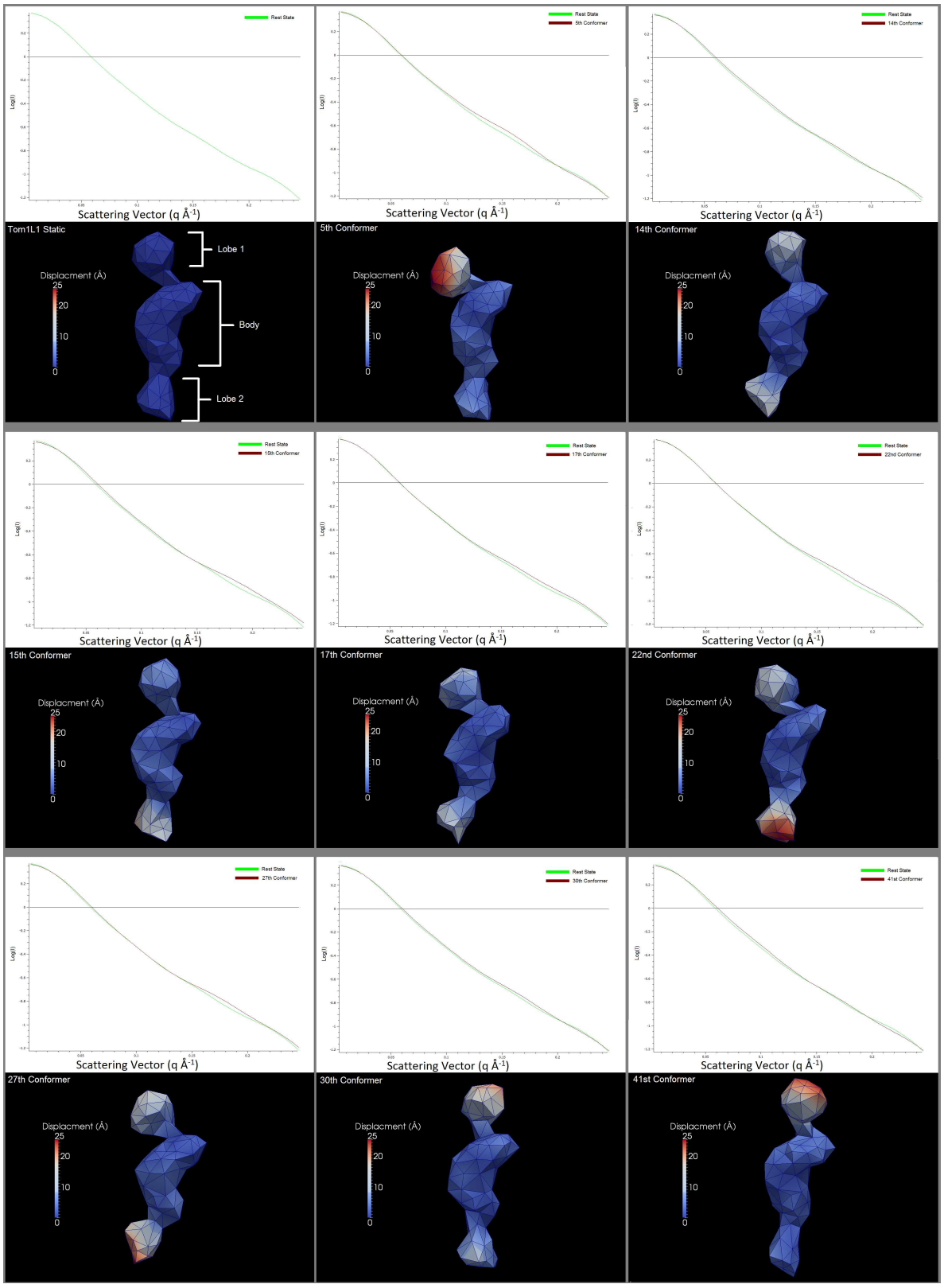
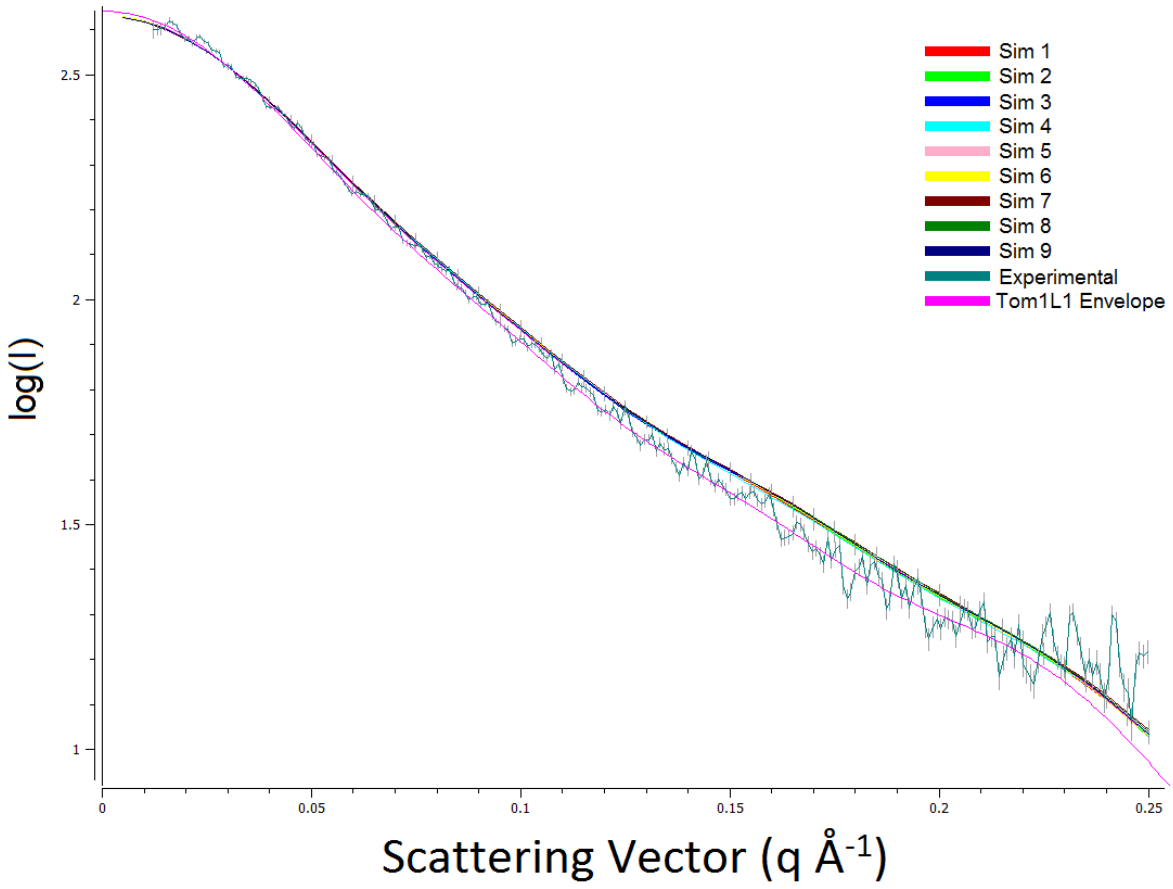


Figure 5.10: Average X-ray scattering resultsm.



in dark pink in Figure 5.6 they are broadly very similar. Furthermore, the difference observed between the averages and the SAXS envelope is comparable to the difference between the initial finite element mesh (Mesh III in Figure 5.6) and the SAXS envelope. We interpret this as meaning that the quality of the initial mesh is crucial and defines how close the final X-ray scattering curves will be to the experimental curves.

The simulations show that the average X-ray scattering curve for all nine simulations is approximately the same. In other words, regardless of the individual dynamics dependent on the elastic properties of each simulation the net scattering is unaffected to within the precision of the measurement. The initial finite element mesh shown in Figure 5.6 (Mesh III) is assumed to be the at rest structure. When thermal fluctuations are added, the structure oscillates around this rest structure. However, the average X-ray scattering signal remains close to that of the rest structure. While this conclusion suggests that it is not be possible to determine dynamics just from the average X-ray signals in the case of Tom1l1, it does demonstrate that neglecting the effects of fluctuations in the reconstruction of the SAXS envelope is reasonable, at least for this protein.

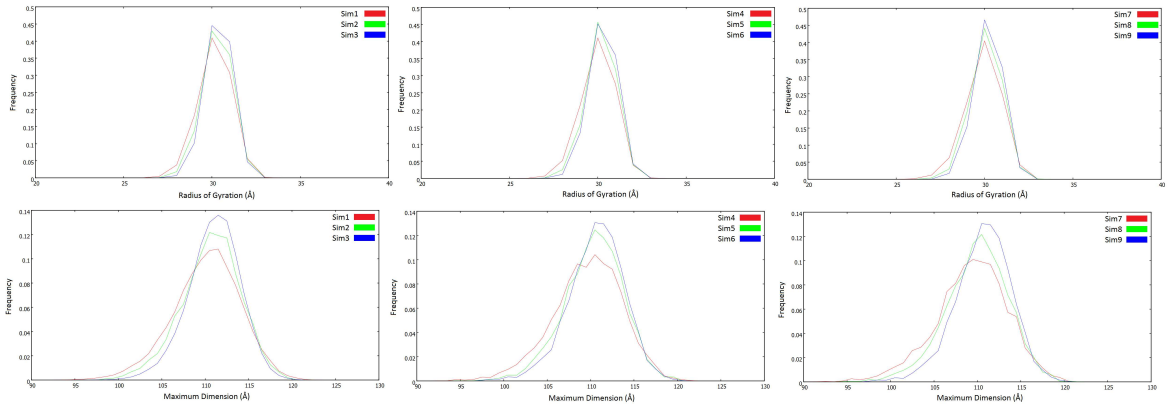
We now discuss the effect of the Young's modulus and Poisson ratio on the distributions of the maximum dimension and the radius of gyration.

5.5.3 Effect of the Young's modulus

The radius of gyration and maximum dimension are calculated for all conformers obtained from the 9 simulations of Tom1L1. These results are presented in the form of histograms in Figure 5.11 and 5.12 to display the distribution of the radius of gyration and maximum dimension.

The average radius of gyration from all simulations is approximately 30 Å while the maximum dimension of Tom1L1 is approximately 120 Å. This compares well with the radius of gyration found from the experimental data of 31 Å and the experimentally observed maximum dimension of 120 Å. The radius of gyration was obtained from the

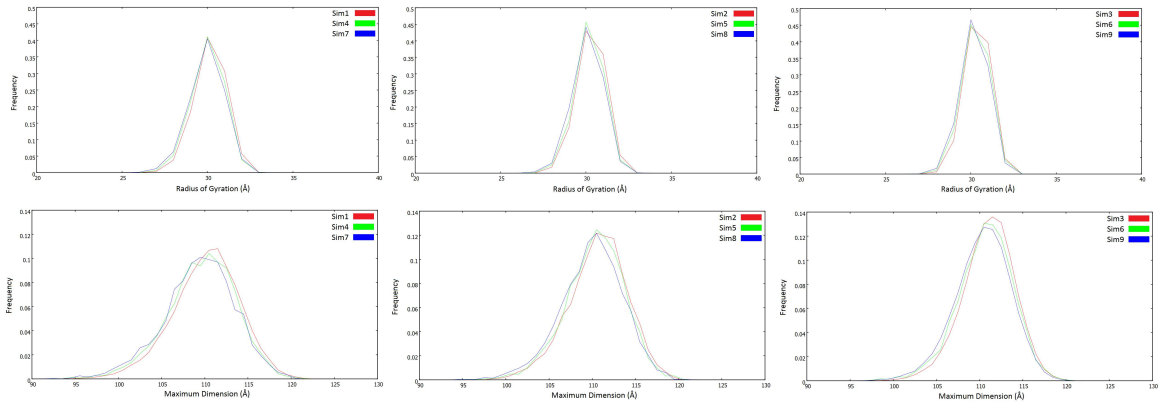
Figure 5.11: Effect of the Young's modulus on the radius of gyration (Above) and the maximum dimension (Below). From left to right the graphs show the Poisson ratio held at 0.40, 0.35 and 0.30. The graphs show that as the Young's modulus is decreased the width of the distribution of the radius of gyration and the maximum dimension increases.



calculated gradient of Guinier[142] region of the experimental X-ray scattering curve and by computing the pair distribution function from the experimental X-ray scattering curve using an inverse Fourier transform and obtaining the point where the distribution function tends to 0 respectively[102]. Care must be taken in calculating the inverse Fourier transform to ensure that the pair distribution function falls off smoothly and does not oscillate.

The effect of varying the Young's Modulus is shown in Figure 5.11 where the changes in the distribution for each of the 3 values of the Poisson ratio are compared. As the Young's modulus is decreased the average radius of gyration and maximum dimension decreases, while the width of the distributions increases. As the Young's modulus is decreased the FFEA model of Tom1L1 becomes more flexible. Thus, both Lobe 1 and Lobe 2 are able to move closer to the body of Tom1L1. This deformation will decrease the radius of gyration and maximum dimension of the finite element model because it is more compact (see Conformer 27 in Figure 5.9). These conformers are not present when the Young's modulus is increased due to the material stiffness.

Figure 5.12: Effect of the Poisson ratio on the radius of gyration (Above) and the maximum dimension (Below). From left to right the graphs show the Young's modulus held at 500 MPa, 650 MPa and 800 MPa. The graphs show that the Poisson ratio has a very small effect on the dynamics.



5.5.4 Effect of the Poisson ratio

The effect of the Poisson ratio on the distribution of the radius of gyration and the maximum dimension is shown in Figure 5.12. In comparison to the Young's modulus the effect of changing the Poisson ratio is very small. For the most part, the graphs in Figure 5.12 overlap and show no significant differences with change in Poisson ratio. One interpretation of this is that the dynamics of Tom1L1 is dominated by bending, which that is controlled by the Young's modulus and not by compressive deformations that are controlled by the Poisson ratio. This again correlates with the dynamics discussed in the analysis of simulation 1 where the most common mode of motion is a pincer movement. This motion is a pure bending motion and reliant on the Young's modulus.

5.6 Overview

The simulations of Tom1L1 agree with the experimental data in terms of the X-ray scattering curve, radius of gyration and the maximum dimension. However, the scattering signal was found to be insensitive to the values of the elastic parameters and so

it is not possible to obtain this information from this form of scattering experiment, at least for this particular biomolecule. Although the configuration space explored by the molecule does depend on the Young's modulus this does not significantly affect the averaged value of the scattering intensity.

This study does demonstrate that we can derive a finite element mesh from this type of experiment, and also demonstrates what features of a biomolecule need to be preserved when coarse-graining, so as to retain compatibility with the SAXS envelope.

Chapter 6

Finite Element Simulations of the Molecular Motor Dynein

In this chapter, I apply the stochastic finite element method discussed in this thesis to model the motion of the molecular motor dynein. Dynein is a large, geometrically complex molecule whose atomic structure is only partially known[143], and hence cannot be studied using molecular dynamics. However, by using low resolution imaging data[144] we can perform a simulation using the stochastic finite element model. Here, I shall give a brief description of the biological function and importance of dynein to eukaryotic life[145] (Section 6.1), a review of the experimental data (Section 6.2) and construction of the finite element meshes for dynein in different biochemical states (Section 6.3), homogeneous and inhomogeneous simulations of dynein and results (Section 6.4).

6.1 Molecular Motors and Biological Function

Molecular motors are nanomachines capable of performing a variety of cellular functions from driving the beating of cilia[146], transporting vesicles[145] in cells and muscle contractions[147]. Molecular motors work by binding Adenosine Triphosphate (ATP) to a region of the motor called the motor domain. ATP is a chemical that effectively stores energy within the phosphate bonds. Molecular motors release this energy by breaking

one of the phosphate bonds by hydrolysing ATP and forming Adenosine Diphosphate (ADP). This process alters the biochemical state of the motor, inducing conformational changes in the motor structure and potentially changing its binding characteristics[148]. Molecular motors cycle through this process of binding and hydrolysing ATP, known as priming the motor, and releasing ADP to perform a power stroke. This process gives rise to the functionality[145][144] of the motor.

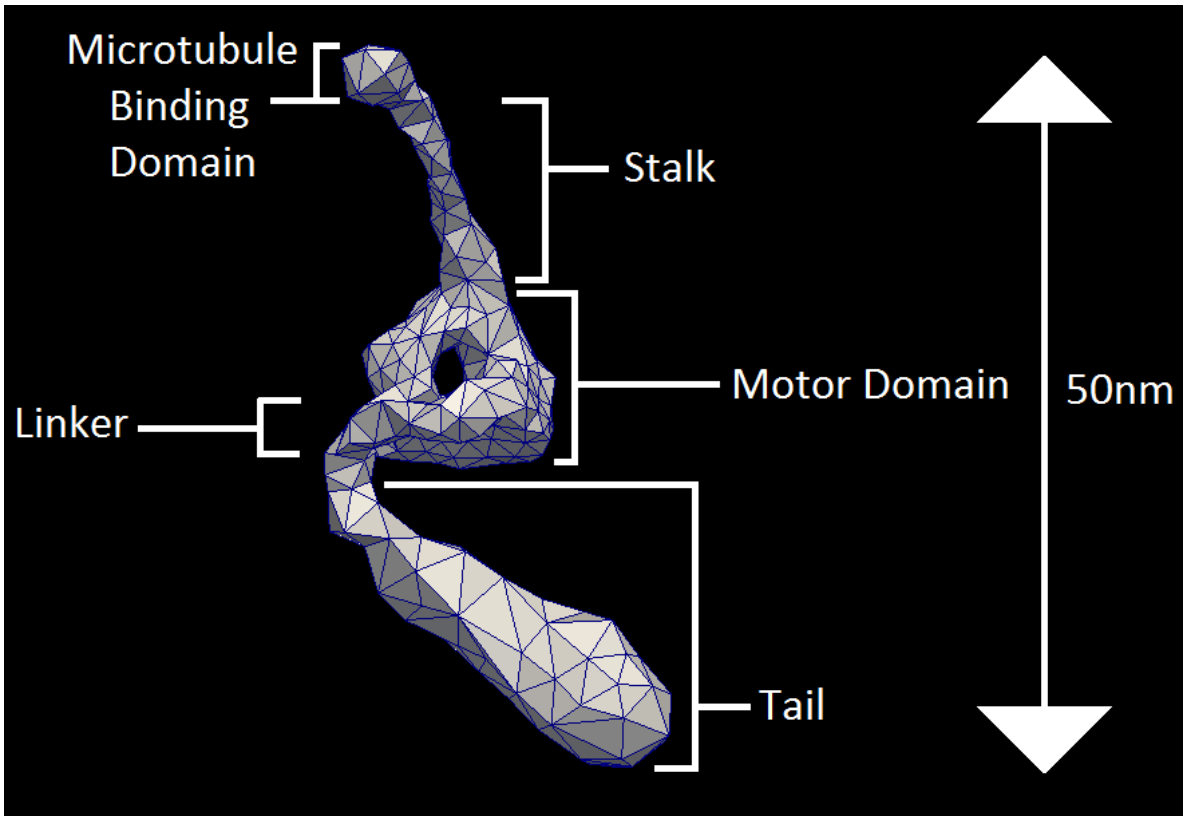
In eukaryotes, there are three cytoskeletal superfamilies of molecular motors: kinesin, myosin and dynein[145]. Of these superfamilies, kinesin and myosin are similar in structure and both have molecular weights on the order of $100kDa$[145]. However, dynein is significantly larger with a molecular weights for the cytoplasmic variant exceeding $1MDa$[145]. In the cell, the molecular motor super families all have different functionalities and bind to different cellular structures. Myosin is involved in muscle contraction and binds to actin filaments, kinesin is a microtubule walker and drags vesicles from the cell nucleus to the membrane and the cytoplasmic dynein variant is another microtubule walker that travels from the cell membrane to the nucleus.

6.1.1 Structure and Biological Function of Dynein

The molecular motor we shall consider is dynein. Dynein binds to microtubules. There are two primary types of dynein found in eukaryotes: cytoplasmic dynein and flagellar dynein[148]. Cytoplasmic dynein is a dimer with a molecular weight of $1.5MDa$[149] and is involved in vesicle transport from the cell membrane to the nucleus and mitosis[148].

For the purposes of this thesis, we are only concerned with flagellar dynein[148]. Flagellar dynein is found in the axoneme. The axoneme is a cytoskeletal structure in eukaryotes found in cilia and sperm tails[82]. Flagellar dynein is the molecular motor that drives the beating motion of the axoneme giving rise to waves that generate propulsion for sperm and the oscillating motion of cilia[144]. There are two experimentally imaged states of dynein called the APO state and the ADPVI state. The APO state corresponds to there being no ATP or ADP bound to the motor domain of dynein.

Figure 6.1: Flagellar dynein in the ADPVI state showing the microtubule binding domain, stalk consisting of a coiled coil of alpha helices, the motor domain containing 6 ATP binding sites, a linker region and a tail that anchors the motor in situ. Structure built from data provided by Roberts et al[144].



While the ADPVI state is a transition state mimic of the ADPPI state that has a very high affinity for the binding sites of the dynein motor domain[150]. A picture of the ADPVI state of dynein is shown in Figure 6.1.

Flagellar dynein is larger and more complex than either kinesin or mysoin. In dynein the ATP binding site and microtubule binding domain are separated by a coiled coil unlike in either mysoin or kinesin[145]. This separation of the ATP binding site from the microtubule binding point requires a mechanism for long range signalling from the motor domain to the stalk when ATP is bound and unbound. There are other complications such as dynein contains six ATP binding sites, only four of which are active in the motor domain as well as a long extended tail and linker regions[148].

In the axoneme, dynein is anchored by its tail to a microtubule in the axoneme

while the microtubule binding domain on the stalk is free to explore conformational space. On hydrolising ATP, dynein undergoes a major conformational change wherein the linker moves across the face of the motor domain and allows dynein to pull on the neighbouring microtubule[144][148]. Dynein motors work cooperatively to generate a wave in the axoneme that drives sperm tail propulsion.

For the remainder of this thesis, we will be exclusively concerned with flagellar dynein and this will be referred to as dynein from this point forth. I will now discuss the relevant experimental dynamics and imaging data that have allowed me to apply the stochastic finite element method to dynein.

6.2 Review of Experimental Data on the Dynamics of Dynein

In this section, I will review two papers that provide measurements on the dynamics of the dynein molecular motor (Section 6.2.1) and reveal the three dimensional shape of the dynein motor (Section 6.2.2).

6.2.1 Dynamics of Dynein

Burgess et al[148] measured the conformations of dynein in two different biochemical states. These are the APO state and the ADPVI state. The APO state corresponds to a post power stroke conformation while the ADPVI state replicates the pre power stroke state where ADPPI is bound.

Burgess et al[148] used negative staining and electron microscopy to obtain low resolution images of many thousands of dynein motors in different conformations. These conformations are combined through image processing to produce a set of conformers that represent the extent of motion due to thermal fluctuations in the ADPVI and APO states. From these images, metrics such as the overall length and angular distributions of various parts of the dynein motor can be measured.

Figure 6.2: Length and angle distributions for the dynein molecular motor in the APO and ADPVI state measured from negative staining experiments. Data provided by Burgess et al[148]

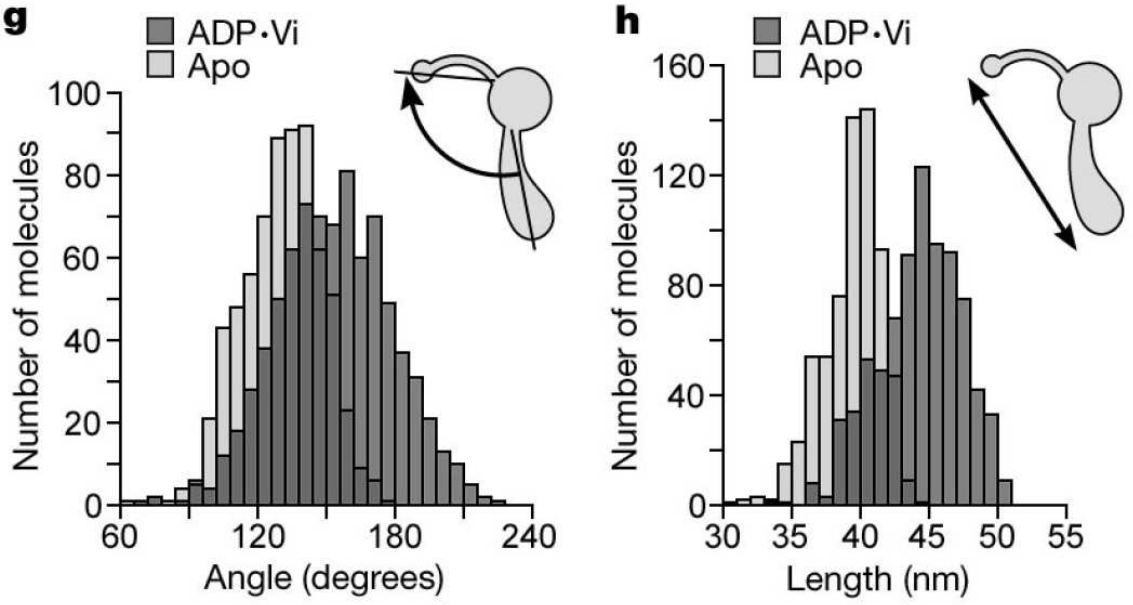


Figure 6.2 shows the experimentally measured distributions of the length of the dynein molecule from the microtubule binding domain to the tip of the tail and the distribution of the angle between a vector running from the base of the stalk to the microtubule binding domain and the linker to the tip of the tail. The experiment shows that in going from the pre power stroke state (ADPVI) to the post power stroke state, there is a significant change in the overall reach of the motor and the conformational space explored by the motor.

Burgess et al[148] also gives the overall flexibilities of the stalk independently of the tail motion and the tail motion independent of the stalk. In the ADPVI state the standard deviations for the stalk and tail are 20 degrees and 18 degrees respectively while for the APO state the standard deviations are 11 degrees for the stalk and 16 degrees for the tail. This data shows that the stalk of dynein stiffens considerably between the APO and ADPVI state whereas the stiffness of the tail would seem to be relatively constant over both states. These observations will allow us to fit the dynamics of the simulations discussed in Section 6.4 to the experimental data. However, before

this can be done we require a 3D structure of dynein.

6.2.2 Cryo-EM Imaging of Dynein

In a separate experiment Roberts et al[144] use flash freezing rather than negative staining to prepare the samples for electron microscopy. The primary difference between these techniques is that in the negative staining technique of Burgess et al the biomolecules are dried onto a film whereas in Cryo-EM the biomolecules are frozen in vitrified ice. This helps to prevent damage to the biomolecules as they are kept intact during the flash-freezing. The molecules can then be imaged under a micrograph. While the signal from each individual motor in the frozen assay is quite weak, by image processing and averaging, the signal to noise ratio can be increased.

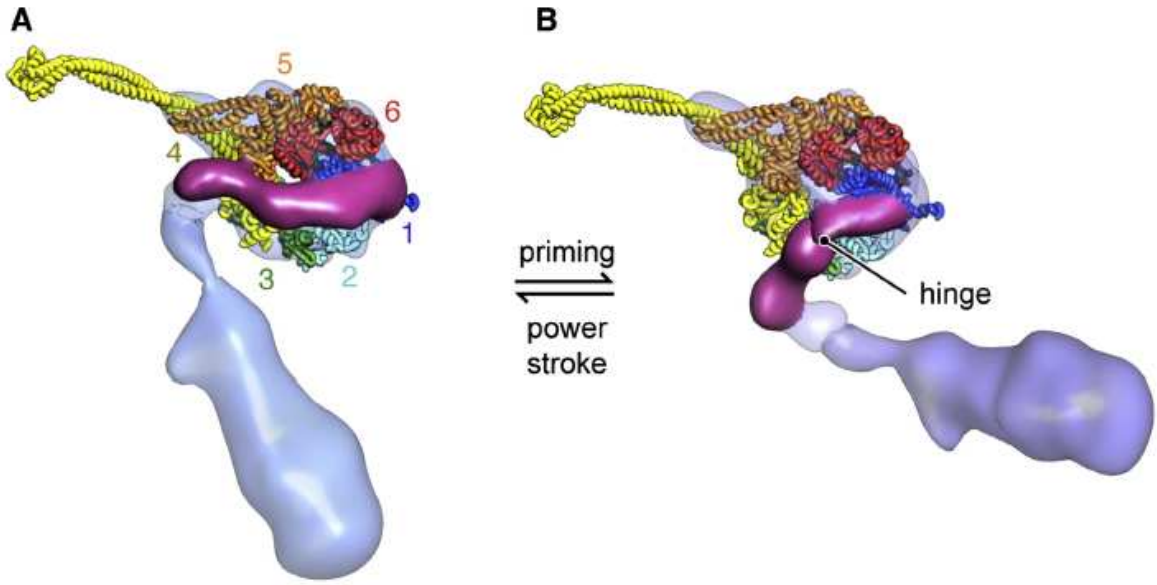
From these images a 3 dimensional structure of dynein can be built up in both biochemical states. This is achieved by using specialist software such as System for Processing Image Data from Electron microscopy and Related fields (SPIDER)[151] that builds a 3 dimensional density map that reflects the imaging data. From this process two Cryo-EM maps of dynein have been produced, one in the APO state and one in the ADPVI state. An image of these is shown below with a partial crystal structure in Figure 6.3 from Roberts et al[144].

Using these density maps, I have constructed a finite element mesh of dynein discussed in the next section.

6.3 Overview of Dynein Simulations: Mesh Construction and Theoretical Issues

In this section, I discuss the issues behind the simulations of dynein including how to construct the finite element meshes Section (6.3.1), how to accelerate the exploration of conformational space (Section 6.3.2) and the theoretical expectations of the simulations (Section 6.3.3).

Figure 6.3: Cryo-Em structures of dynein in the APO state (left) and ADPVI state (right). Image provided by Roberts et al[144]



6.3.1 Construction of the Finite Element Meshes

I have constructed a finite element mesh that represents the shape of the dynein molecular motor in both biochemical states. These meshes were constructed from the Cryo-Electron density maps obtained from Roberts et al[144]. Dynein is made of distinct parts such as the linker, tail, motor domain and stalk in the case of the ADPVI state. From the density maps of each of these sections, an isosurface of equal densities can be constructed in SPIDER[151]. The isosurfaces from each section of dynein can then be positioned and oriented correctly relative to one another to build an overall model of the shape of the full motor. From these oriented and positioned isosurfaces of each section of the dynein motor we then construct a surface mesh from which the volume mesh is generated. I will now give more precise details on the nature of the meshing for both biochemical states.

ADPVI State Finite Element Mesh

The ADPVI state configuration is formed from 4 separate density maps for different sections of the dynein motor. The construction of the final mesh of ADPVI dynein takes

place in three stages. Firstly, the surface meshes of the four different parts of dynein are extracted from the density maps of each part respectively by the Chimera[152] package. The advanced coarse grainer, discussed in Chapter 4, is then used to coarse grain the surface mesh of each of the four sections (See Figure 6.4). These four separate sections then have to be combined to form a single surface mesh for the entire molecule. Each section has to be positioned and oriented correctly and the four surface meshes were then stitched together at their intersections. This process was completed by hand. This yields the mesh shown at the bottom of Figure 6.4.

APO State Finite Element Mesh

The construction of the mesh for the APO state follows the same process as the ADPVI state. The APO state uses the same coarse grained meshes as the ADPVI for the tail and stalk as the changes in the motion and function of dynein are mostly due to reshaping of the motor domain. A mesh is constructed for the motor domain using the density map and Chimera that is then coarse grained using the advanced coarse grainer. The individual pieces are oriented correctly and can then be stitched by hand. This process is detailed in Figure 6.5 with the final APO mesh shown at the bottom.

6.3.2 Critical Damping of the Dynein Motor

In absence of an external fluid the thermal motion of the dynein molecule is under-damped due to the relatively low internal viscosity. This means that the molecule takes a long time to fully explore conformation space.

In order to accelerate the exploration of conformational space, I have used the external solvent described in Chapter 2 and 3 to critically damp the motion of the tail of dynein. The motion of the tail gives the longest timescales because it is the largest moving object in the simulations. Assuming ergodicity, critical damping will not change the conformational space explored by dynein for a particular set of material parameters because it does not affect the energetics, only the rates of timescales within the system. This damping is achieved by modeling the motion of the tail as a 2^{nd} order ordinary

Figure 6.4: The four finite element meshes of the different parts of ADPVI dynein are shown at various stages of mesh construction. In the top row the meshes extracted from the density maps are shown, in the middle after coarse graining with the advanced coarse grainer and bottom shows the final product after stitching.

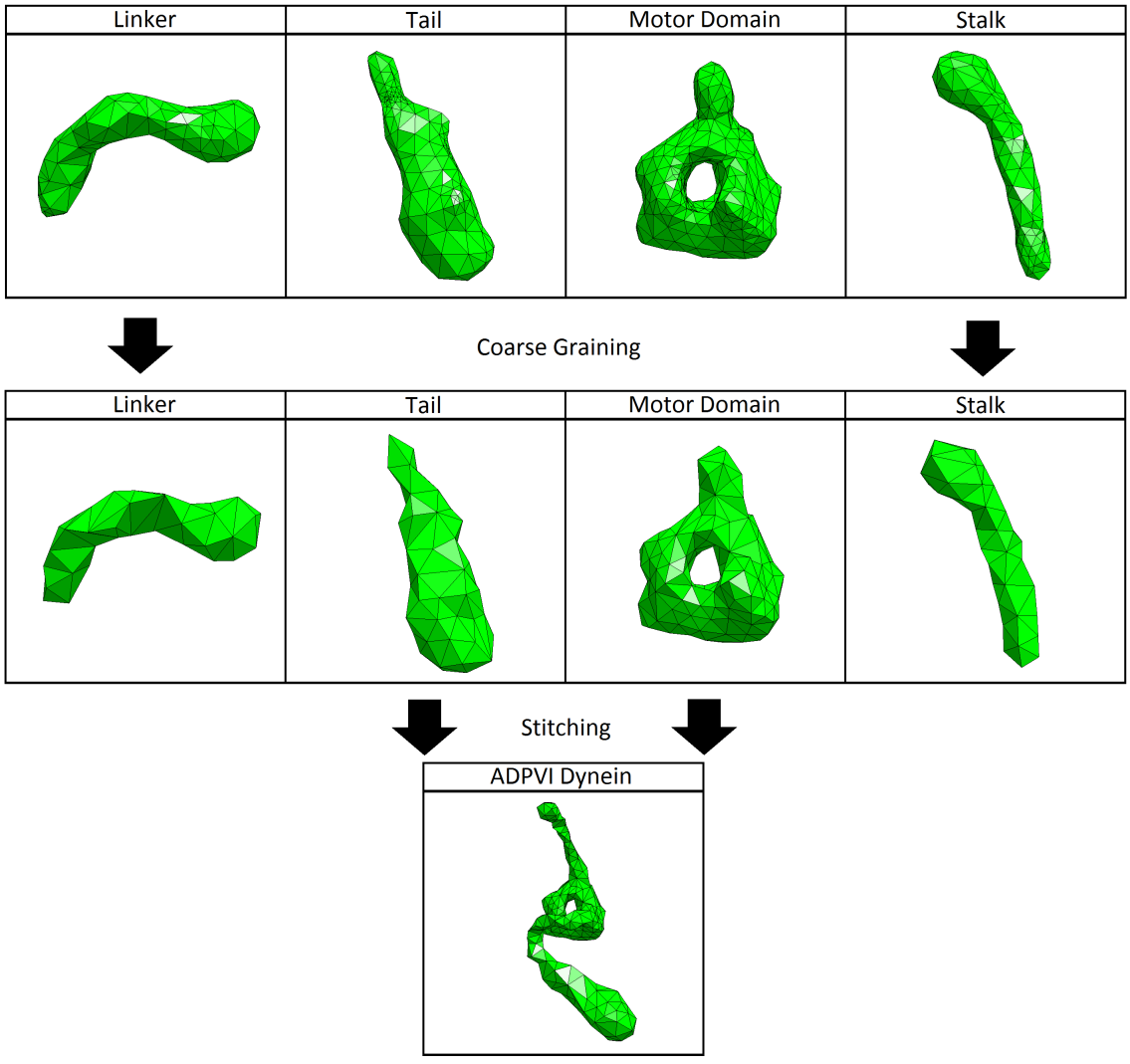
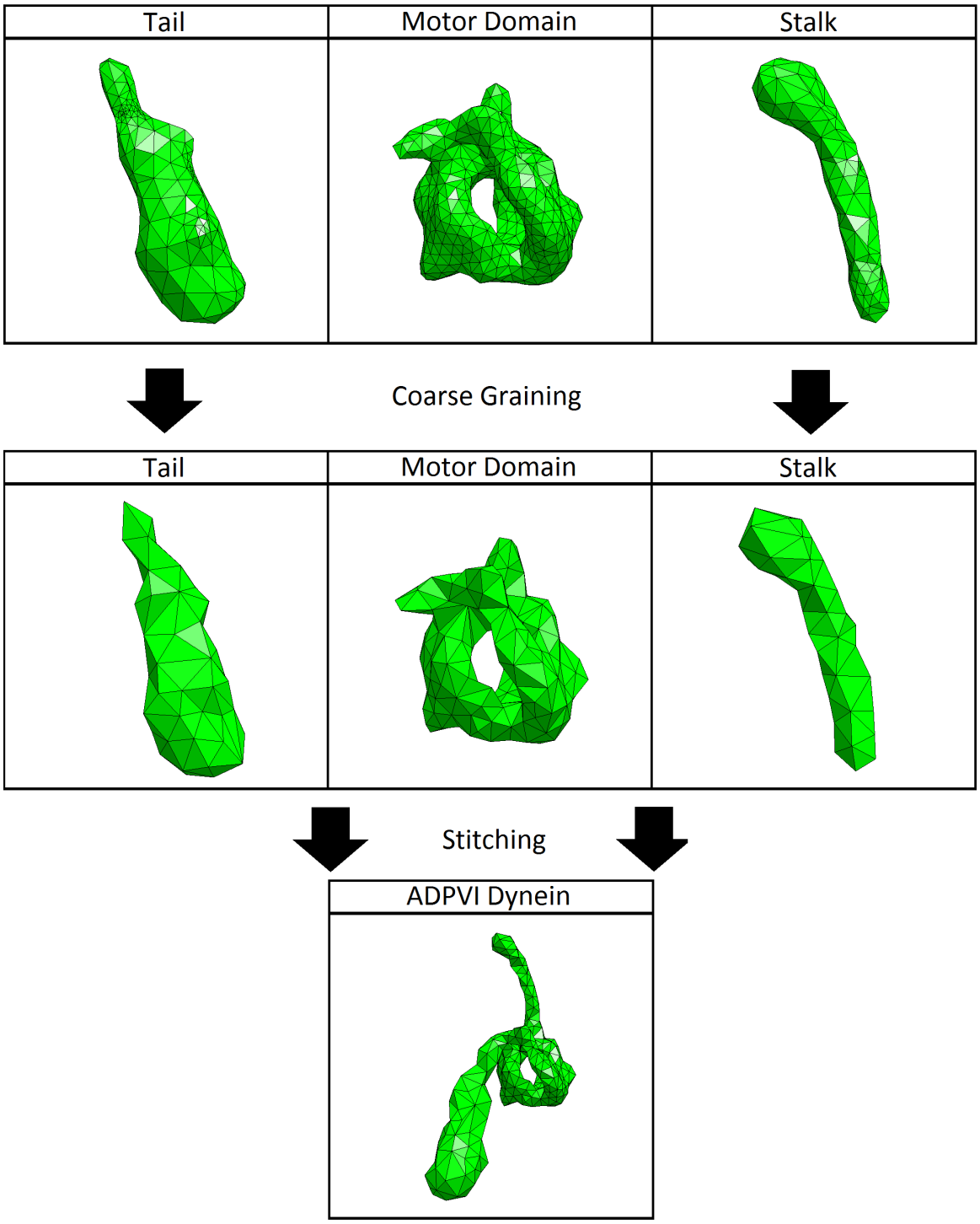


Figure 6.5: The three finite element meshes of the different parts of APO dynein are shown at various stages of mesh construction. The stalk and tail are the same meshes used for the ADPVI state in both initially and after coarse graining. The three meshes can then be stitched together by hand to form the final APO state mesh.



differential equation:

$$m\frac{d^2x}{dt^2} + \nu\frac{dx}{dt} + kx = 0, \tag{6.1}$$

where m is the mass of the tail ν is the tail's drag coefficient and k is a spring constant.

The coefficients m, ν and k can be estimated by characterising the tail as a sphere of radius $5nm$ that moves through a background solvent that induces Stokes' drag on the sphere with a maximum motion of about $15nm$ obtained from experiment (See Figure 6.6). Thus coefficients are given by:

$$m = \rho V \tag{6.2}$$

where $V = \frac{4\pi r^3}{3}$,

$$\nu = 6\pi\eta r, \tag{6.3}$$

and the spring constant by,

$$k = \frac{k_B T}{\langle x^2 \rangle} \tag{6.4}$$

using the equipartition theorem and where $\langle x^2 \rangle$ is the mean square distance moved by the sphere.

The roots l_1 and l_2 of the characteristic equation of equation (6.1) are then given by:

$$l_i = \frac{-\nu + \sqrt{\nu^2 - 4mk}}{2m}. \tag{6.5}$$

We seek the critically damped solution such that the discriminator is 0 so that:

$$\eta = \frac{\sqrt{4mk}}{6\pi r}. \tag{6.6}$$

All the quantities in equation (6.6) are known and yield a solvent viscosity for critical

Figure 6.6: Simple model of the motion of the tail through a background solvent. This model can then be used to attribute to coefficients from equation (6.1).

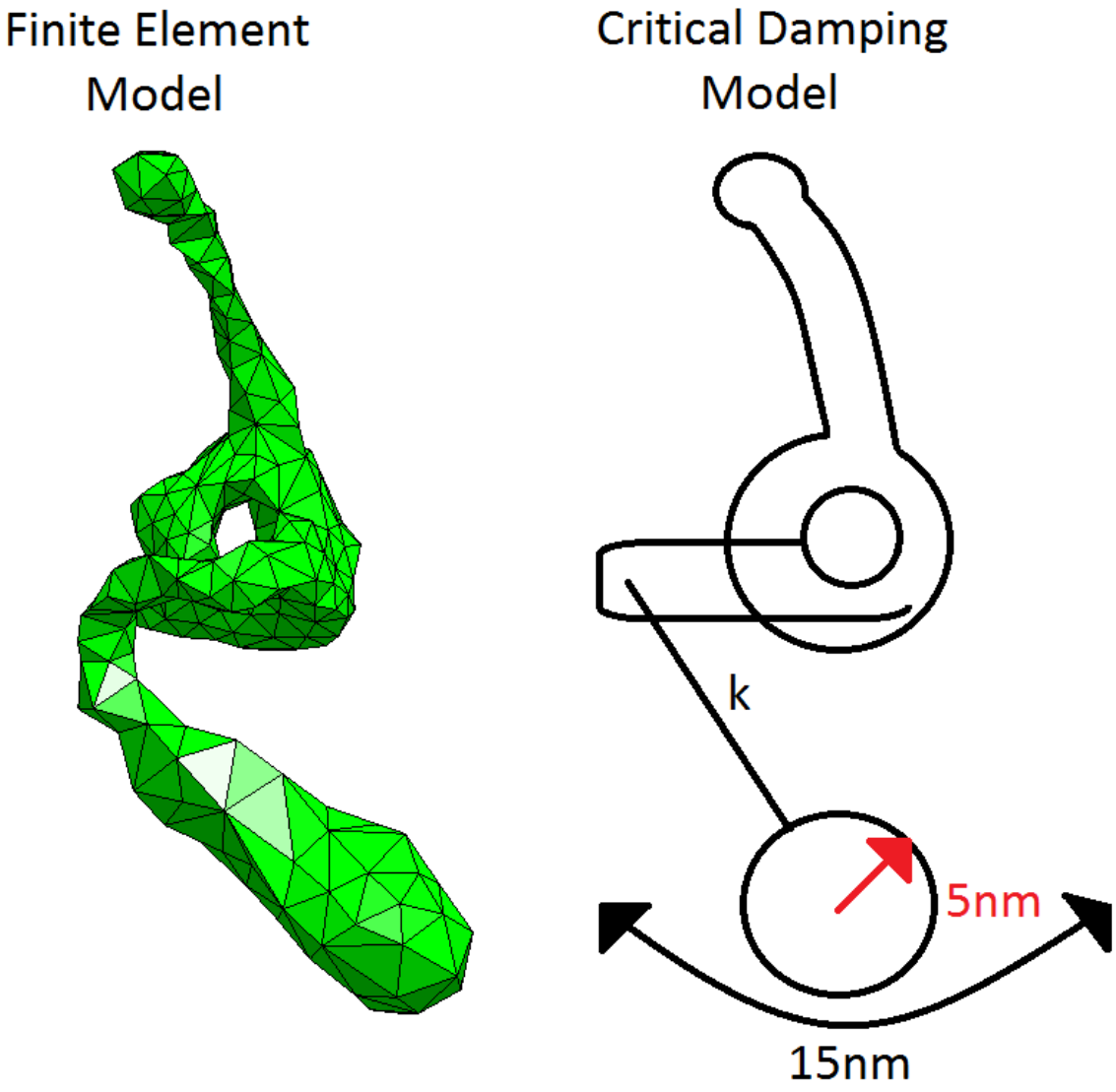
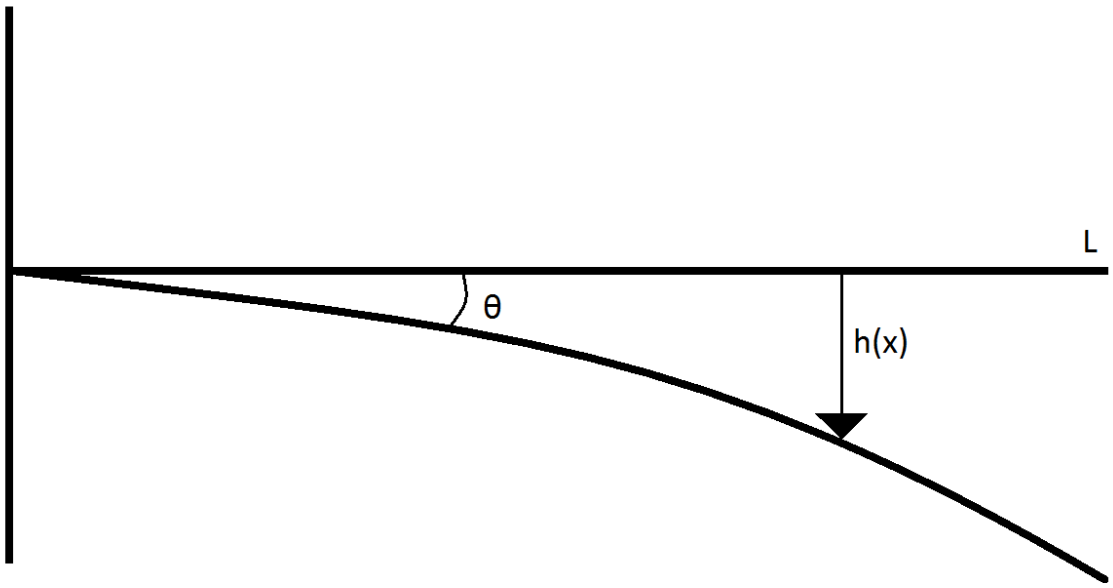


Figure 6.7: Diagram showing the deflection $h(x)$ of a beam with one end fixed.



damping of $10^{-6}Pas$ which is three orders of magnitude less than that of water. The two timescales of such a system are given by $\tau_1 = \frac{m}{\nu}$ and $\tau_2 = \frac{\nu}{k}$. At critical damping, these time scales are approximately equal with a time of $5ns$.

If we were to use the viscosity of water for the solvent, then the motion would be strongly over damped. In other words the the Reynolds number is low.

6.3.3 Theoretical Modeling of the Stalk and Tail

From the structure of the dynein molecule the principle forms of the motion are flexing of the stalk and tail. Since these are long thin structures, we would expect their motions to be analogous to that of beams. However, since both the stalk and tail are effectively pinned to the motor domain and linker rather than a free beam, we have a beam where one end is fixed, as shown in Figure 6.9. This gives rise to a highly simplified model for model for the motion of the stalk and tail that is a modification to the beam theory discussed in Chapter 3[153]. This results in a power law in the Young's modulus that will be important to the interpretation of the results in Section 6.4.2.

The equation on motion describing the deflection $h(x,t)$ of the uniform beam is given by[153]:

$$EI\frac{d^4h(x,t)}{dx^4} - \mu\frac{d^2h(x,t)}{dt^2} = q(x), \tag{6.7}$$

where μ is the mass per unit length of the beam and $q(x)$ is the external load subject to the boundary conditions. We are going to use a cantilevered beam wherein one end is fixed so that $\hat{h}(0) = 0$ and $\frac{d\hat{h}(0)}{dx} = 0$. The other end is subject to a free torque boundary condition so that $\frac{d^2\hat{h}(L)}{dx^2} = 0$ and $\frac{d^3\hat{h}(L)}{dx^3} = 0$ where L is the length of the beam[153].

The modes of vibration are found by seeking a solution of the form $h(x,t) = Re(\hat{h}(x)e^{i\omega t})$ so that:

$$\frac{d^4\hat{h}(x)}{dx^4} = k^4\hat{h}(x), \tag{6.8}$$

where $k^2 = \left(\frac{\mu\omega^2}{EI}\right)^{\frac{1}{2}}$.

The general solution of equation (6.8) is given by[153]:

$$\hat{h}(x) = A_1\cosh(kx) + A_2\sinh(kx) + A_3\cos(kx) + A_4\sin(kx). \tag{6.9}$$

Imposing the boundary conditions, the solutions for $h(x)$ are found to be of the form:

$$\hat{h}(x) = \cosh(kx) - \cos(kx) + \left(\frac{\cosh(kL) + \cos(kL)}{\sin(kL) + \sinh(kL)}\right)(\sin(kx) - \sinh(kx). \tag{6.10}$$

Where k must satisfy:

$$\cosh(kL)\cos(kL) + 1 = 0, \tag{6.11}$$

Equations (6.10) and (6.11), called the frequency equation, describe the shape and the modes of a cantilever beam. The roots of the frequency equation provide the sup-

ported wavenumbers of the beam from now on termed k_n[153]. Each mode contributes to the overall deflection of the beam such that $\hat{h}(x) = \sum_n \hat{h}_n(x)$ where,

$$\hat{h}_n(x) = A_n \left(\cosh(k_n x) - \cos(k_n x) + \left(\frac{\cosh(k_n L) + \cos(k_n L)}{\sin(k_n L) + \sinh(k_n L)} \right) (\sin(k_n x) - \sinh(k_n x) \right). \tag{6.12}$$

We can now proceed with our derivation by utilizing equation (3.9) from chapter 3:

$$W = \frac{EI}{2} \int_0^L \left(\frac{\partial^2 h(x)}{\partial x^2} \right)^2 dx. \tag{6.13}$$

Since the modes are orthogonal[153], we can determine the energy contributed by each mode as:

$$W_n = \frac{A_n^2 EI}{2} \int_0^L \left(\frac{\partial^2 S_n}{\partial x^2} \right)^2 dx. \tag{6.14}$$

Where $\hat{h}_n(x) = A_n S_n$. In principle, we can integrate (6.14) analytically however, this is not necessary for the purpose of the discussion here. We shall denote by c_n the integral in (6.14) so that W_n is of the form:

$$W_n = \frac{A_n^2 c_n EI}{2}. \tag{6.15}$$

Equation (6.15) is quadratic in the amplitude A_n while the other terms are all constants determined by the properties of the beam. Thus, at thermal equilibrium, the following relationship holds:

$$\langle W_n \rangle = \frac{k_B T}{2} = \frac{\langle A_n^2 \rangle c_n EI}{2}. \tag{6.16}$$

We can now re-arrange equation (6.16) to yield:

$$\langle A_n^2 \rangle = \frac{k_B T}{c_n EI}, \tag{6.17}$$

where $\langle A_n \rangle = 0$ due to symmetry. Hence, since the mode shape is not affected by averaging it follows that:

$$\langle \hat{h}_n^2(x) \rangle = \langle A_n^2 \rangle S_n^2(x). \tag{6.18}$$

By evaluating the position at the end of the beam, we can obtain the variance of any individual mode of the system. This can be summed to produce the variance $\langle \hat{h}^2(L) \rangle$ because each of the modes are independent of one another[153]. Thus:

$$\langle \hat{h}^2(L) \rangle = \sum_n \langle A_n^2 \rangle S_n^2(L) = \frac{k_B T}{EI} \sum_n \frac{S_n^2}{c_n}. \tag{6.19}$$

From equation (6.19) the standard deviation of the distribution is proportional to $E^{-\frac{1}{2}}$. This can be converted to an angular distribution using Figure 6.7 and a small angle approximation such that $\theta = \frac{h(L)}{L}$ so that the standard deviation of the angular distribution is given by:

$$\sigma = \left(\frac{k_B T}{EIL^2} \right)^{\frac{1}{2}} \left(\sum_n \frac{S_n^2(L)}{c_n} \right)^{\frac{1}{2}} \tag{6.20}$$

Thus, the angular distribution of the stalk and tail should follow a power law in terms of the Young's modulus. Note that although this result was derived for a uniform beam, the result extends to beams of non-uniform cross-section as this only affects the shape $S_n(x)$ and the values of the eigenvalues k_n.

6.4 Dynein Simulations

In this section, I describe the results of the simulations of dynein and compare the range of motion with the distribution of conformers found by Burgess[148]. There are two distinct sets of simulations run with homogeneous (Section 6.4.1) and inhomogeneous material parameters (Section 6.4.3). In the homogeneous simulations, we assume that the material properties are uniform throughout the molecule and explore how the range

of motion of the stalk and tail depends upon the values of the elastic parameters. From these simulations we are able to identify the material parameters that best match the experimental range of motion of the stalk and tail of dynein independently (Section 6.4.2). Finally in Section 6.4.3, we perform simulations with differing material parameters for the tail and stalk in order to match the range of motion of the tail and stalk in the same simulation (Section 6.4.4).

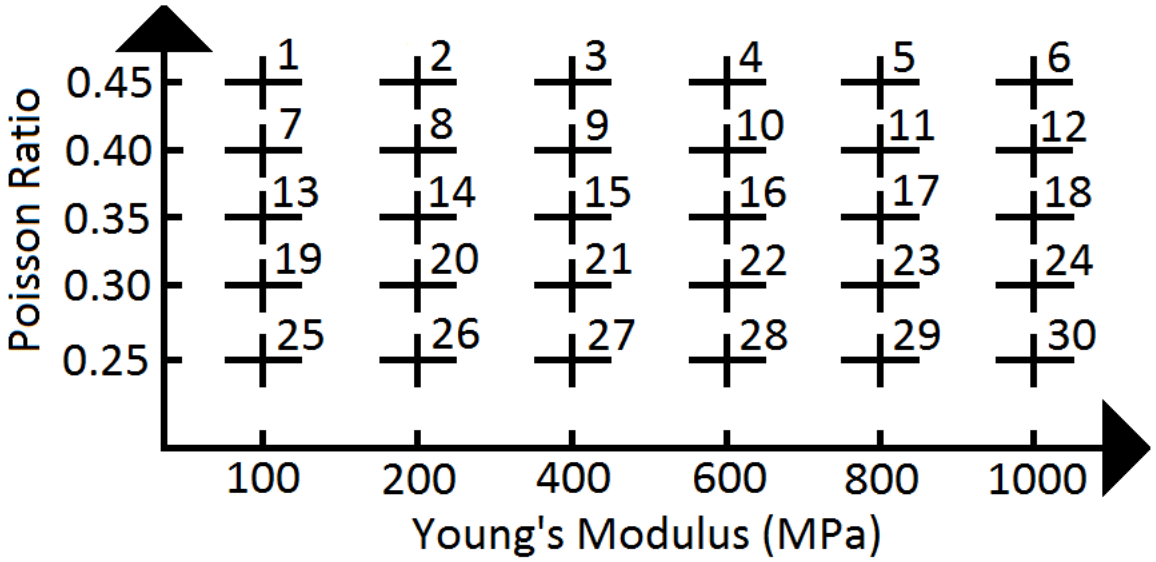
6.4.1 Homogeneous Dynein Simulations

The purpose of this set of simulations is to determine how the range of motion of the molecule depends upon the material parameters. As the overall amount of deformation at a particular temperature is going to be purely controlled by the elasticity of model, we must understand how the statistics of the motion of different sections of dynein in both biochemical states vary with the elastic parameters. From the experimental data, the standard deviation of the distribution for the tail and stalk is given in both biochemical states of dynein. We aim to find the elastic parameters that yield the same deviations.

In total, 60 different cases were run; 30 for the APO state and 30 for the ADPVI state as shown in Figure 6.8. Each simulation was run for sufficient time for the standard deviation of the stalk and tail angle distributions to converge to within approximately half a degree. All simulations used the following parameter values: Density $1500kgm^{-3}$, internal viscosities $0.001Pas$ (the same as water), temperature $300K$ and external viscosity $10^{-6}PaS$ (as discussed in Section 6.3.2). The elastic parameters in the model range between a Poisson ratio of 0.45 and 0.25 with Young's moduli varying between $100MPa$ and $1000MPa$, therefore covering the entire biological range of interest (See Figure 6.8).

The elastic parameters are assumed to be homogenous in each simulation so that each element has the same Young's Modulus and Poisson ratio with the values given in Figure 6.8. Thus, each simulation will have a different range of accessible conformers

Figure 6.8: Table showing the elastic parameters used in the 30 simulations for the ADPVI and the APO state of dynein. The crosses mark parameter values used for the Young's modulus and Poisson ratio that is set to be equal in all elements in the simulation. The elastic parameters are the only variables changed between the different simulations and thus each simulation will have a different accessible range of motion. The aim is to then match these ranges of motion against the experimentally observed distributions in Burgess et al[148].

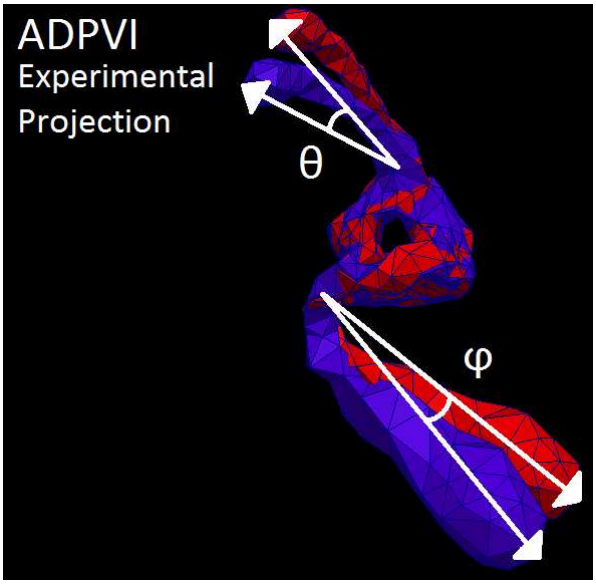


governed by these different elastic parameters. Thus, we can then simulate the motion of the dynein motor in both the ADPVI and APO states until enough independent conformers have been observed to characterise the overall range of motion of the dynein motor for a particular set of elasticites. These ranges of motion can then be compared to the experimentally known distributions found in Burgess et al[148] and the correct set of elasticities required to required to repliocate the experimental data extracted.

In these simulations, we are only modelling the thermal fluctuations of the dynein motor about the elastic energy minimum corresponding to the APO and ADPVI states. We are not intending to switch between the two states. This is a good match to the conditions of the experiments in Burgess et al[148] where the dynein motors are effectively locked in one biochemical state or the other.

In these simulations, I measured the angular distribution of both the stalk and tail, first having removed the effects of translation and rotation by aligning the system

Figure 6.9: The rest state of the ADPVI state (in red) with a deformed state (in blue). The stalk angle θ and tail angle ϕ are visually defined.

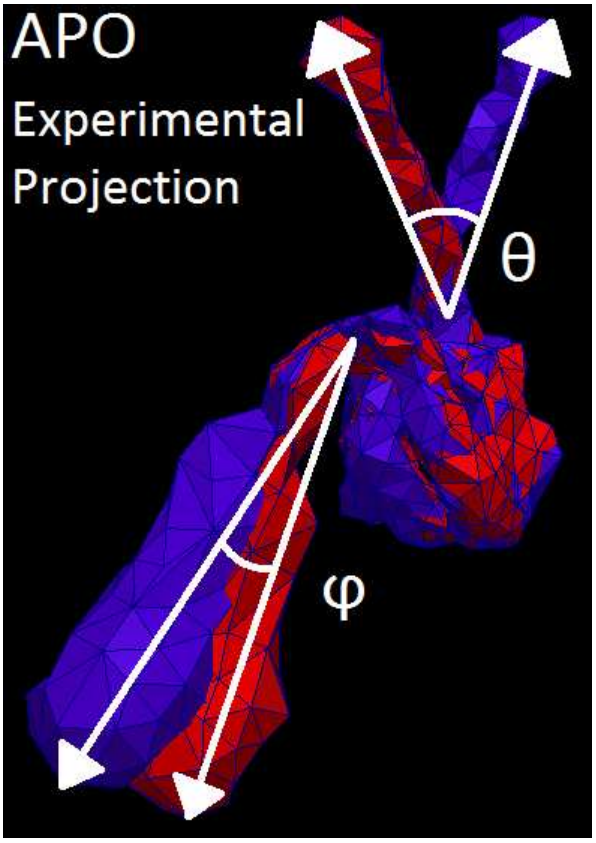


against a fixed axis within the motor domain. This is done by using a vector normal to the hole through the motor domain, that we define as the x-axis, and a second vector from the centre of the hole to the base of the stalk. The cross product of these two vectors then forms the third vector and a complete coordinate system that any conformation of dynein can be referenced to. This effectively pins the motor domain and shows the motion of the tail and stalks respectively, akin to the experimental data[148].

This allows for a consistent definition of the tail angle and stalk angle to be made between different time steps in the simulations as shown in Figure 6.9 and 6.10. The stalk angle θ and the tail angle ϕ are defined using the rest state and a deformed state of dynein in both the APO and ADPVI state. The stalk angle is defined as the change in angle of the vector from the tip of the stalk to the base of the stalk. Similarly the tail angle is defined from the change in angle of the vector from the tip to the base of the tail.

Note that in the experiments the angle measured is the projection onto the plane perpendicular to the hole axis, and therefore we shall report angles measured in this

Figure 6.10: The rest state of the APO state (in red) with a deformed state (in blue). The stalk angle θ and tail angle ϕ are visually defined.



projection. I have also examined the angular distribution in the perpendicular plane but the results are broadly the same and so not shown here.

6.4.2 Homogeneous Dynein Simulation Results

The standard deviations of the stalk and tail angles for the two biochemical states of dynein have been analysed for all conformers in all simulations to produce the following elasticity maps.

Figure 6.11 shows the APO state stalk and tail angle standard deviations. The results show that the flexibility of both the stalk and tail is highly dependent on the Young's modulus but less sensitive to the Poisson ratio. This is demonstrated by the near vertical nature of the colour gradient on the graphs in Figure 6.11. This indicates that just as in Chapter 5, the primary modes of APO dynein are driven by bending and not volume change.

The experimentally observed standard deviations for the APO state are 11 degrees for the stalk and 16 degrees for the tail[148]. To match these flexibilities the simulations require a stalk Young's modulus of approximately $350 - 400MPa$ and tail Young's Modulus of approximately $100MPa$.

Figure 6.12 shows the flexibilities stalk and tail of the ADPVI state. The results are very similar to the APO state results in that again the Young's modulus is a far more important quantity in defining the flexibility of the stalk and tail. The experimental data for the ADPVI state indicates that the stalk has a standard deviation of 20 degrees and a tail standard deviation of 18 degrees[148]. This corresponds to Young's moduli of approximately $175MPa$ for the stalk and $150MPa$ for the tail. This suggests that in that going from the ADPVI state to the APO, the stalk gets stiffer and the tail gets slightly softer.

This result reflects the experimental observation that the range of motion in the stalk is reduced in the experimental data in going from the ADPVI state to the APO state. The relationship between the tail in the two states is far less clear. The range of

Figure 6.11: Elasticity maps of the APO state stalk angle standard deviation (above) and tail angle distribution (below).

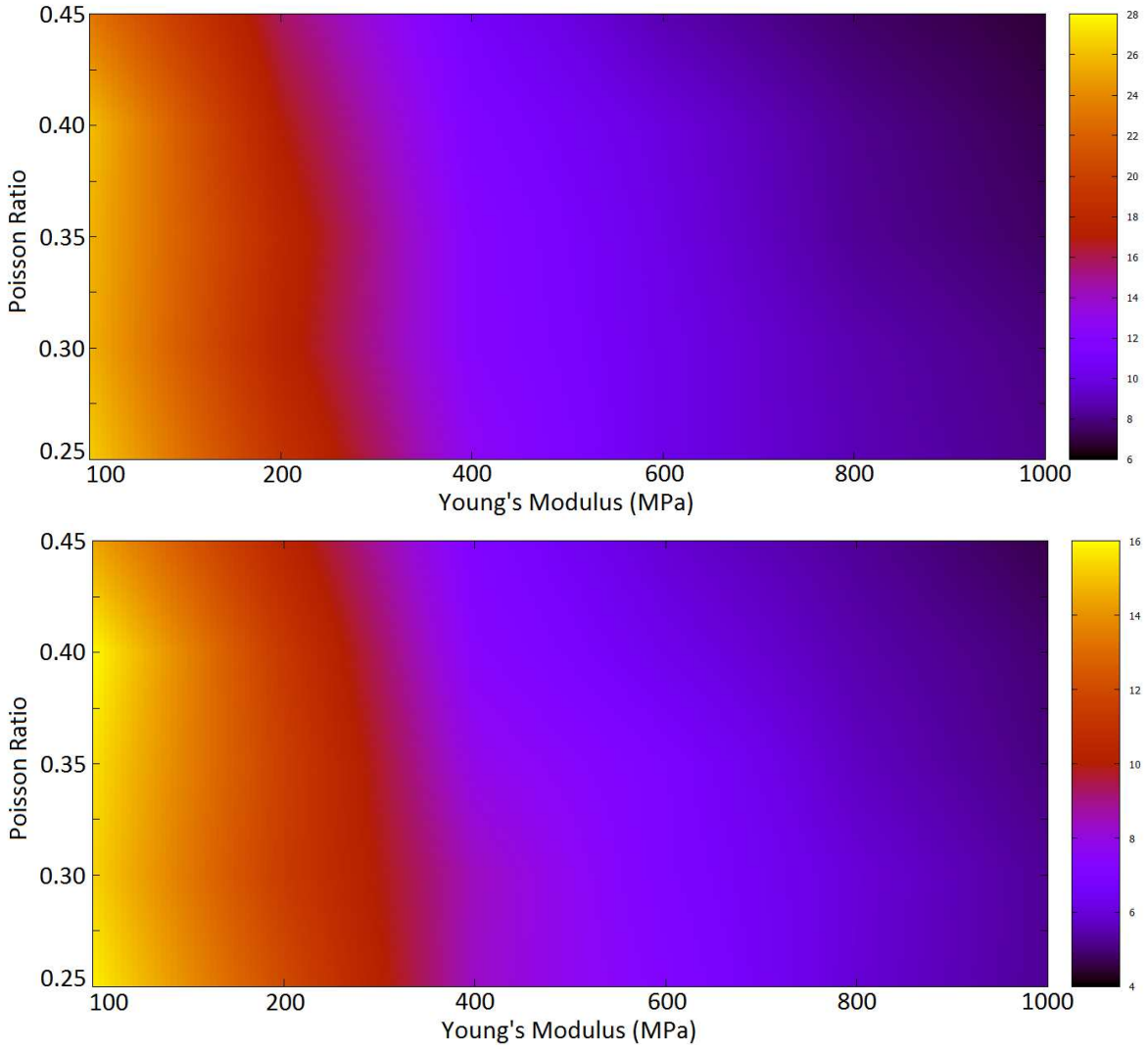


Figure 6.12: Elasticity maps of the ADPVI state stalk angle standard deviation (above) and tail angle distribution (below).

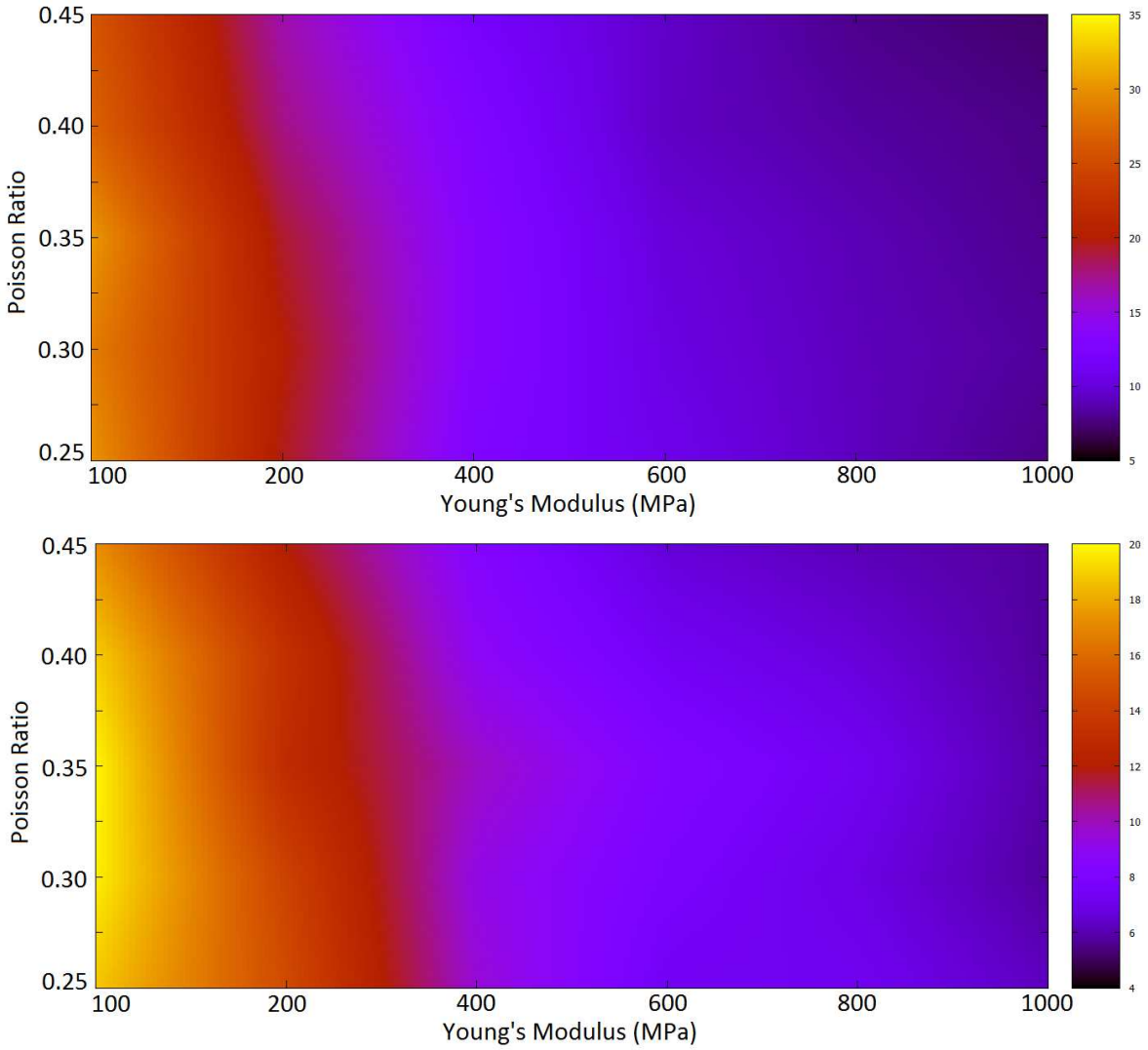


Table 6.1: Fitting parameters for the ADPVI state in the plane perpendicular the the hole for the stalk and tail.

Poisson Ratio	a Stalk	b Stalk	a Tail	b Tail
0.25	0.58 ± 0.02	6.06 ± 0.10	0.50 ± 0.03	5.26 ± 0.02
0.30	0.56 ± 0.01	5.94 ± 0.08	0.53 ± 0.02	5.45 ± 0.10
0.35	0.58 ± 0.01	6.08 ± 0.09	0.50 ± 0.02	5.27 ± 0.15
0.40	0.55 ± 0.02	5.82 ± 0.13	0.52 ± 0.02	5.32 ± 0.09
0.45	0.56 ± 0.01	5.81 ± 0.08	0.49 ± 0.02	5.10 ± 0.13

motion between the two states is very similar with standard deviations of 18 degrees for the ADPVI state and 16 degrees for the APO state. However, the simulations indicate the Young's modulus differs between the two states by a factor of 2. The reason for this is that in the ADPVI state the tail is effectively longer than in the APO state due to the position where the linker is bound to the motor domain. In the APO state much more of the linker is directly bound to the motor domain, which shortens the length of tail and thus reduces the effective range of motion. This is then compensated for by the change in the Young's modulus of the material.

These maps can be analysed further by looking at lines of constant Poisson ratio and plotting a graph of the log of the Young's modulus against the log of angle. If the mode of motion is essentially that of a beam flexing from a fixed point then this should reveal the power law discussed in Section 6.2.3. Figure 6.13 shows a selection of results for the APO and ADPVI states.

The data shown in Figure 6.13 is a small snapshot of all the simulations and a complete analysis of the data is given in Tables 6.1-6.4 using a fit to the line $f(x) = -ax+b$. The fitting parameters are based on a χ squared fit to the data using Gnuplot.

Tables show that the fitting parameter a is generally very close to the predicted value of $\frac{1}{2}$ and usually within the error. However, the case of the stalk in the ADPVI state in the plane perpendicular to the hole, a is always above the trend by at least ten percent. In this case, we conclude that while the basic behaviour is that of a flexing beam there are secondary effects that modify the results compared to beam bending

Figure 6.13: Log graphs of the observed angle against the Young's Modulus showing the observed power law. The data for these graphs is drawn from the APO with Poisson ratio 0.30 for the stalk and 0.45 from the tail (Top). While the ADPVI data (bottom) is drawn from simulations with Poisson ratios 0.45 for the stalk data and 0.40 for the tail data.

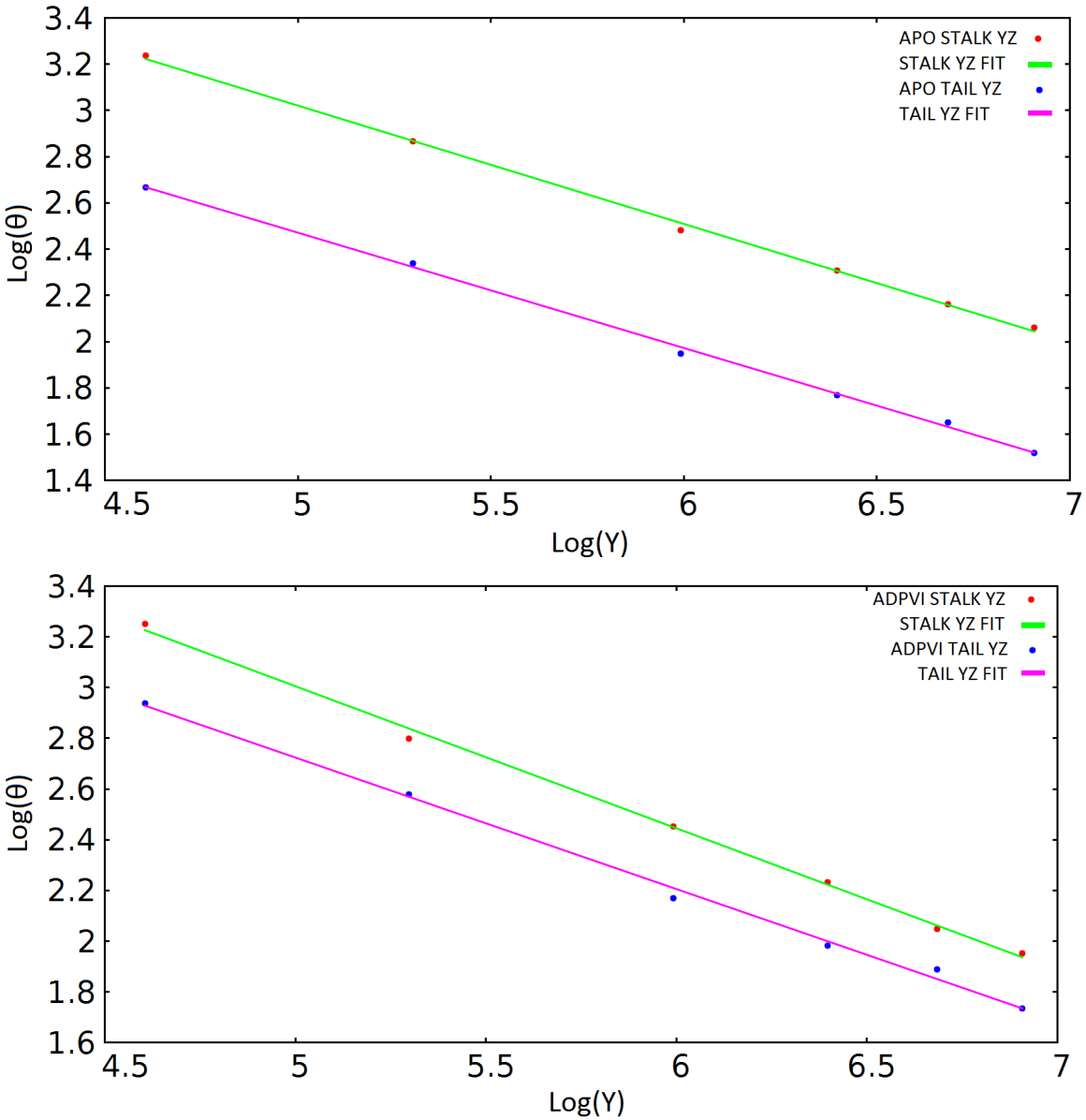


Table 6.2: Fitting parameters for the ADPVI state in the plane parallel to the hole for the stalk and tail.

Poisson Ratio	a Stalk	b Stalk	a Tail	b Tail
0.25	0.52 ± 0.01	5.63 ± 0.05	0.53 ± 0.02	6.02 ± 0.09
0.30	0.51 ± 0.01	5.60 ± 0.04	0.55 ± 0.02	6.13 ± 0.10
0.35	0.53 ± 0.01	5.69 ± 0.07	0.56 ± 0.01	6.17 ± 0.04
0.40	0.50 ± 0.01	5.44 ± 0.08	0.52 ± 0.03	5.87 ± 0.15
0.45	0.50 ± 0.01	5.39 ± 0.05	0.52 ± 0.01	5.81 ± 0.09

Table 6.3: Fitting parameters for the APO state in the plane perpendicular the the hole for the stalk and tail.

Poisson Ratio	a Stalk	b Stalk	a Tail	b Tail
0.25	0.53 ± 0.01	5.72 ± 0.07	0.48 ± 0.02	5.01 ± 0.09
0.30	0.51 ± 0.01	5.58 ± 0.06	0.46 ± 0.01	4.88 ± 0.07
0.35	0.54 ± 0.01	5.75 ± 0.05	0.49 ± 0.02	5.02 ± 0.09
0.40	0.55 ± 0.01	5.78 ± 0.08	0.52 ± 0.02	5.15 ± 0.09
0.45	0.54 ± 0.01	5.60 ± 0.03	0.50 ± 0.01	4.96 ± 0.06

Table 6.4: Fitting parameters for the APO state in the plane parallel to the hole for the stalk and tail for the stalk and tail.

Poisson Ratio	a Stalk	b Stalk	a Tail	b Tail
0.25	0.52 ± 0.01	5.83 ± 0.05	0.52 ± 0.01	5.27 ± 0.08
0.30	0.52 ± 0.01	5.76 ± 0.06	0.51 ± 0.01	5.17 ± 0.07
0.35	0.51 ± 0.01	5.73 ± 0.04	0.51 ± 0.01	5.15 ± 0.06
0.40	0.53 ± 0.02	5.79 ± 0.10	0.50 ± 0.01	5.06 ± 0.05
0.45	0.51 ± 0.01	5.63 ± 0.04	0.51 ± 0.01	5.12 ± 0.07

theory.

As the Young's modulus is increased, the beam appears to be less stiff than one would expect for its Young's modulus. This rules out the effect of non-linear elasticity since these would increase the effective stiffness. Volume effects can also be ruled out because the observed power is not dependent on the Poisson ratio. The most likely cause is that the base of the stalk is not fixed but moves slightly accommodating larger bends by violating the static base boundary condition. This would increase the range of motion making the beam appear less stiff for a given Young's modulus.

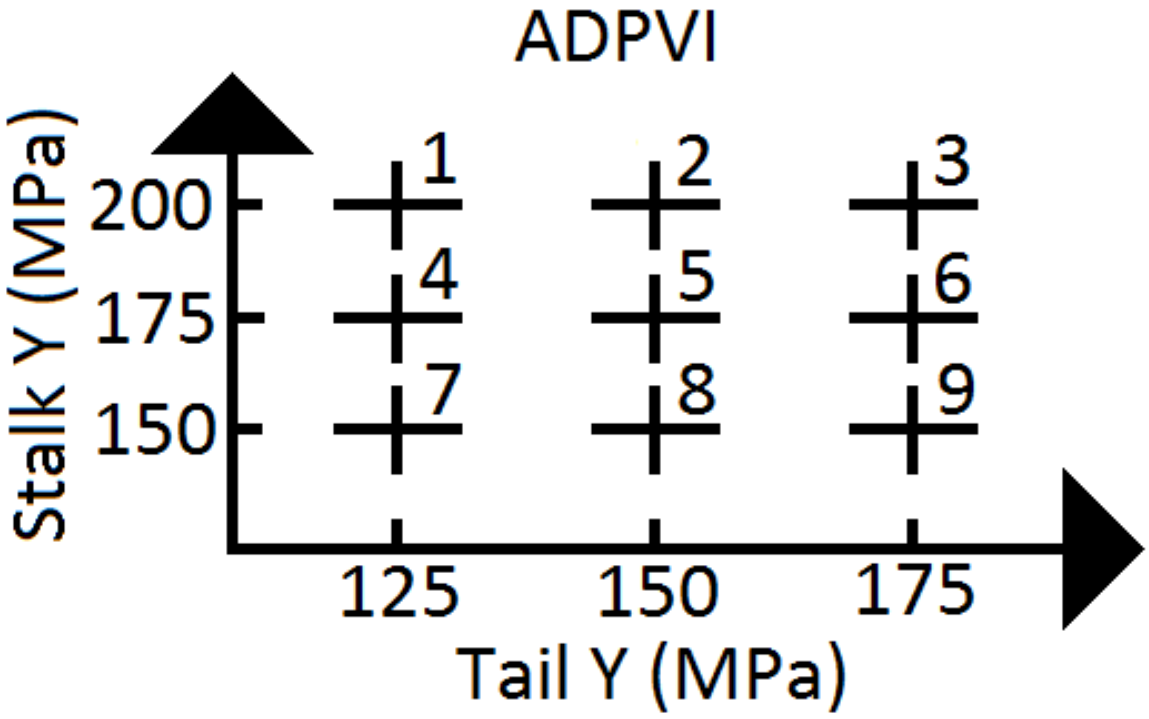
From these homogeneous simulations of dynein we have been able to obtain the elastic parameters that best fit the motion of the stalk and tail. These suggest that the elastic properties of these two sections are not the same, which is expected given their different secondary structure[143]. Therefore, in order to most closely match the motion of dynein we need to use different values for the elastic parameters in the two parts of the molecule.

6.4.3 Inhomogeneous Dynein Simulations

The inhomogeneous simulations use the results from the homogeneous simulations to parameterise the elasticities of the stalk and tail of dynein in both biochemical states. Under the assumption that there is no long range communication through the biomolecule between the stalk and tail, then the Young's Moduli observed in Section 6.4.2 should yield the correct angular distributions for the stalk and tail. We can use the results from the elasticity maps in Figures 6.11 and 6.12 to give elasticities that will give the correct standard deviations.

The results discussed in Section 6.4.2 show that the standard deviations of the distributions of the head and tail are not significantly affected by the Poisson ratio. Instead the Young's modulus plays a much more important role. On this basis, I have fixed the Poisson ratio of these simulations to be 0.35 and alter only the Young's modulus of the material between finite elements. From Figures 6.11 and 6.12 the

Figure 6.14: Young's Moduli for the Stalk and Tail for the ADPVI state.



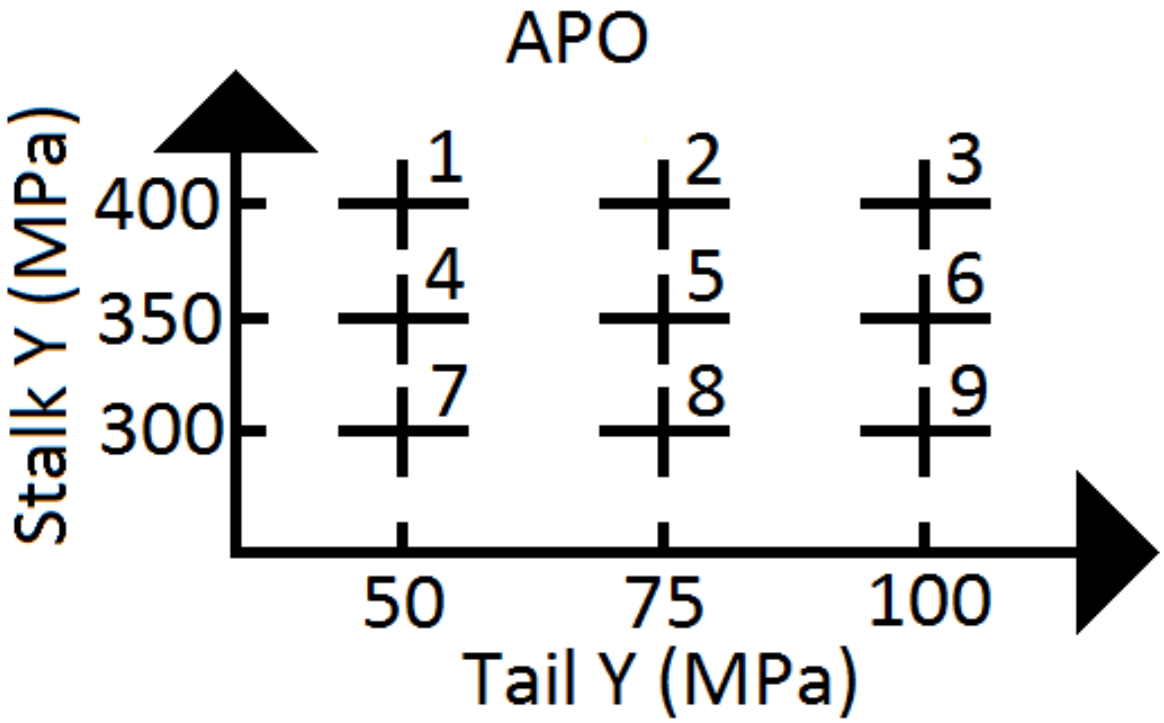
correct Young's moduli for the stalk and tail of the ADVPI state are around $175MPa$ and $150MPa$ respectively. For the APO state the correct Young's moduli are $350MPa$ and $75MPa$.

A total of 18 simulations, 9 for each state of dynein, have been run with parameters detailed in Figures 6.14 and 6.15. The simulations are designed such that at least one of the simulations should pick up the correct combination of Young's Moduli.

In order to define the Young's modulus of each finite element, I have used a simple rule. Any element in the tail or linker will use the tail Young's modulus in a specific simulation and any element in the stalk will use the stalk's Young's modulus. In the motor domain the Young's modulus is varied linearly between the connection point at the linker to the stalk. This ensures that there is a smooth transition between material properties and avoids any problems with a hard barrier between the stalk and tail that might have unexpected effects.

In terms of the remaining parameters: The density was set to be $1500kgm^{-3}$, internal

Figure 6.15: Young's Moduli for the Stalk and Tail for the APO state.



viscosities set to be $0.001 Pas$, temperature $300K$ and external viscosity $10^{-6} Pas$ the same as the homogeneous simulations. Each simulation as run for a total of $500ns$. This provides approximately 100 independent conformers based on the results in Section 6.2.2.

6.4.4 Inhomogeneous Dynein Simulation Results

The inhomogeneous simulations were analysed in the same way as the homogeneous simulations. First I have checked that the expected flexibilities of the stalk and tail in both the ADPVI state and APO state match the predictions from Section 6.4.3. Figure 6.16 shows the distributions of the flexibilities for the stalk and tail in the APO state and Figure 6.17 shows the flexibilities in the ADPVI state.

The results for the APO simulation show that there is no communication between the stalk and tail that affects their overall dynamics. This is shown by stalk distribution being constant as the Young's modulus of the tail is varied and vice versa. Furthermore,

Figure 6.16: Distributions of the stalk (Top) and tail (Bottom) as a function of the inhomogeneous input parameters for the APO state.

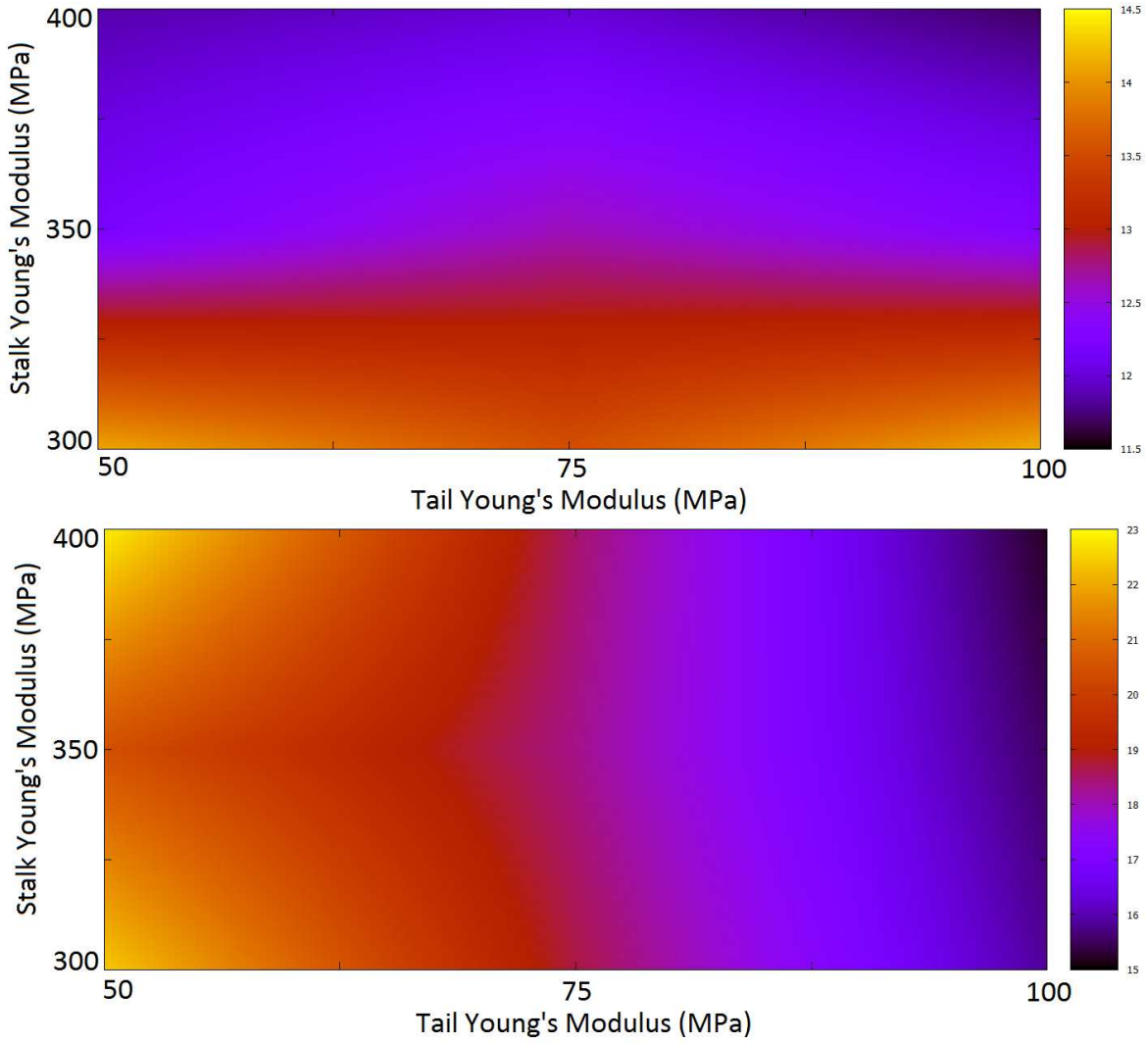
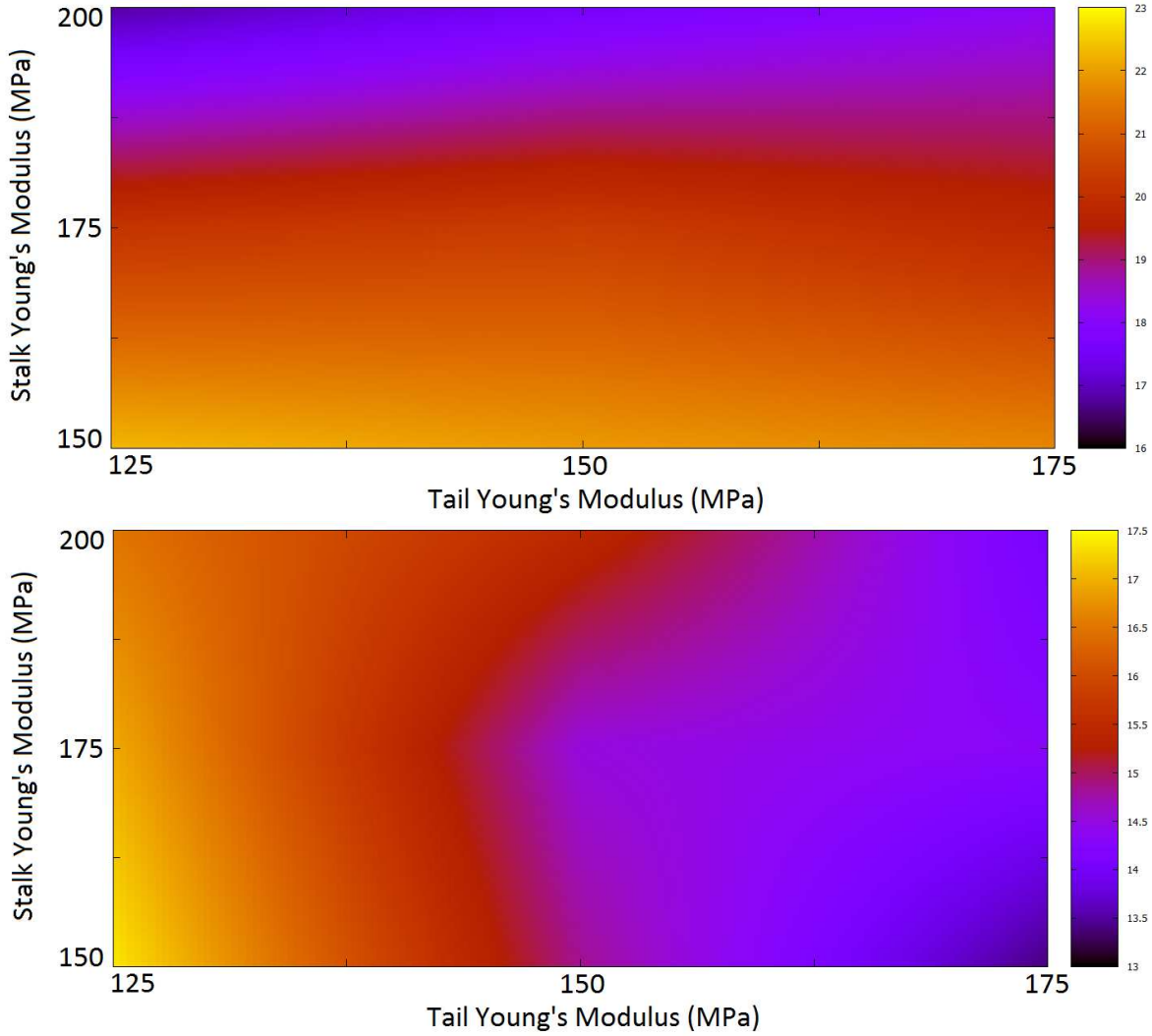


Figure 6.17: Distributions of the stalk (Top) and tail (Bottom) as a function of the inhomogeneous input parameters for the ADPVI state.



from this data we can see that simulation 3 with a Young's modulus $400MPa$ for the stalk and $100MPa$ gives the closest match to the experimental data[148] with a stalk standard deviation of 11.7 degrees and tail standard deviation of 15.2 degrees.

The ADPVI results reinforce the argument that there is no communication between the stalk and tail as there is no evidence of the tail Young's modulus affecting the flexibility of the stalk or vice versa. Simulation 4 yielded the closest match to the experimental data with a Young's modulus for the stalk of $175MPa$ and tail modulus of $125MPa$. These parameters yields a stalk standard deviation of 20.1 degrees a tail

standard deviation of 17.0 degrees.

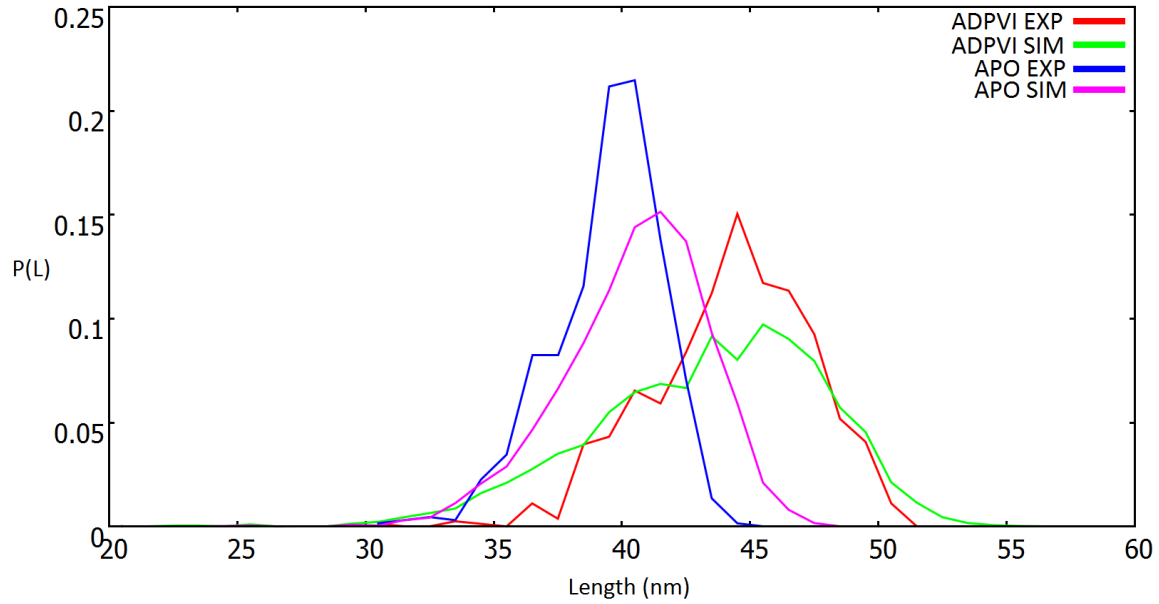
Having established the elastic properties based upon the standard deviation of the angular distribution, I will now compare the simulation results with the full set of measurements of Burgess et al[148]. From APO simulation 3 and ADPVI simulation 4 we can now calculate the length and angle distributions in the same manner as the experimental data. This is shown in Figure 6.20 with the length data shown above and the angle data shown below.

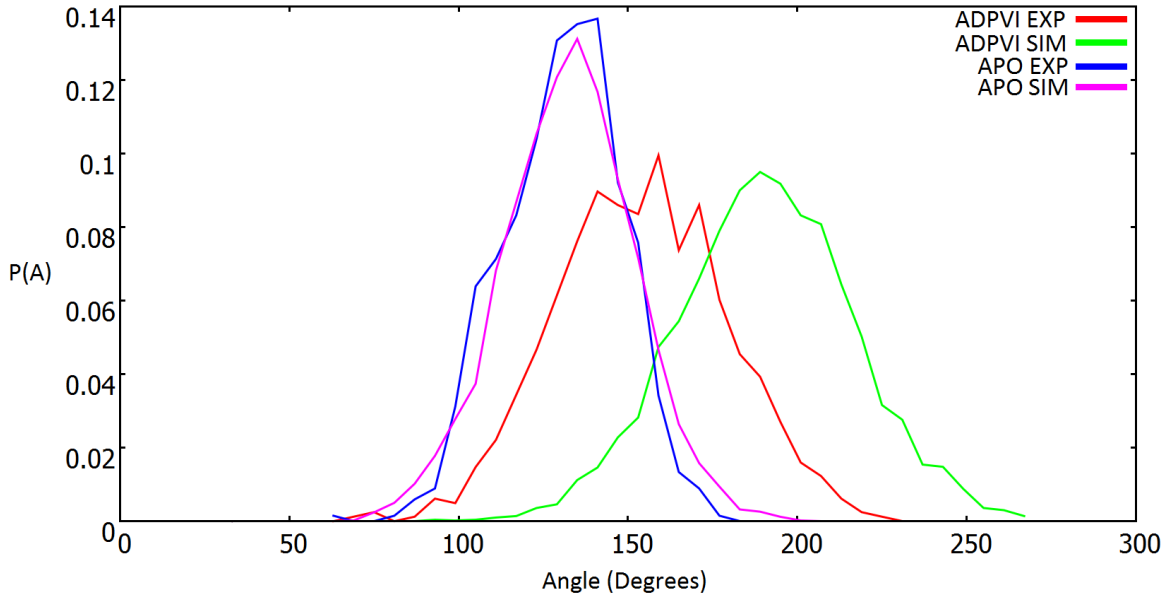
These distributions match qualitatively the experimental data; the mode for the APO state is $42nm$ and for the ADPVI state $46nm$ this is slightly longer than the experimental distributions by $2nm$[148]. Both distributions show a minimum length of approximately $30nm$, which compares well for the APO case but is slightly too short in the case of the ADPVI case by $3nm$. The maximum length for the APO state is $48nm$ and $56nm$ for the ADPVI state[148]. Again, this is slightly broader by a few nanometers than the experimental observation. However, overall the length distributions are a reasonable approximation of the experimental data.

We now examine the angle between the stalk and tail for both the APO and ADPVI state. In the simulations, the APO state has a mode of 130 degrees with a minimum angle of 70 degrees to a maximum angle of 200 degrees. The width of the APO angular distribution is a good match to the experimental data indicating that the standard deviations of the simulated and experimental data are very similar. The main difference between the two data sets is that the maximum angle of the simulated data is approximately 20 degrees too large[148]. The ADPVI state does not have such a good fit. The mode of the simulated ADPVI state is 180 degrees while the mode for the experimental data is 160 degrees. However, the width of the experimental and simulated distributions is the same indicating that the flexibilities are correct but consistently out by approximately 20 degrees[148].

From this, we conclude that the overall fit to the experimental data seems to be reasonable. Properties such as the mode and standard deviations of the distributions

Figure 6.18: Total length (Top) and angle (Bottom) distributions for dynein using the closest fitting simulations to the experimental data .





between experimental and simulated data are conserved reasonably well. The main exception to this is the ADPVI angular distribution where the mode is shifted by about 20 degrees between the simulated and experimental data. This would suggest that there is something wrong with the base state of the ADPVI finite element mesh in that the tail is in the wrong place. We can conclude this because the tail undergoes the largest change in position going from the APO to the ADPVI state and thus is the most likely source of any error in the base state. An alternative explanation is that there is a difference between the negative staining experimental data and the Cryo-EM data. The negative stating data is 2D because of the drying process onto a graphite film[148] this might alter the conformers observed in that experiment. By contrast the Cryo-Em data[144] is 3D and does not rely on a drying process onto a film. Thus, the shift in the angle in the ADPVI state might simply be an artifact of the 2D and 3D experimental data.

The only other significant deviation from the simulated data and experimental data is that the model predicts longer lengths between the stalk and tail and larger angles between the stalk and tail. A possible reason for this is that there is a feature called the buttress[144] that would impede this kind of motion of the stalk, but it is not modelled in the simulations as it is a fine scale feature removed by coarse-graining.

6.5 Conclusion

In this section, I have demonstrated how low resolution imaging data can be used to provide a simulation of the motor protein dynein. Three dimensional electronic density measurements are used to construct the finite element mesh, and then separate experimental measurements of the standard deviation of the angular distribution are used to establish the elastic properties. The resulting simulations are then able to reproduce the experimental distributions to reasonable accuracy. These simulations can now be used to study the details of the power-stroke of the dynein motor, as discussed in the conclusion chapter. In particular, they provide dynamic information

not previously available as the experiments provide only static conformers.

Chapter 7

Conclusion

The purpose of this thesis was to develop a novel coarse grained simulation method capable of tackling problems of mesoscopic length and time scales. In chapter 1, we discuss both the need for such a simulation in order to access biologically important length and time scales and also the physics that must be included to model motion at this scale [46][47][48][49]. Chapters 2-3 shows the numerical and mathematical details of the model including the validation, while chapters 4-6 show examples of how the model can be applied to specific molecules.

Chapter 1 discusses the range of simulation tools available for studying biological systems. This shows that there are no ubiquitous methods to deal with length and time scales in the range of $10nm$ to $100nm$ and $100ns$ to microseconds. This is marked contrast to the macroscale where finite element[3] and finite volume methods are well established and the nanoscale where and molecular dynamics[16] and quantum chemistry[9] are used. While there are many simulation methods in the mesoscale, they are mainly adapted to solving fluid type problems such as dissipative particle dynamics and not the individual dynamics of a protein[57]. However, it is the dynamics and ability of proteins to change shape structure governs their functionality.

Chapter 2 and 3 detail the derivation and validation of the stochastic finite element model. Specifically, chapter 2 shows how the fluctuation dissipation relation can be written down and solved for a Kelvin-Voigt material within the finite element framework.

This is the key result from chapter 2 that gives rise to an efficient and computationally inexpensive algorithm that can model the important thermal fluctuations that give rise to protein dynamics. Chapter 3 is concerned with validating the algorithm described in chapter 2 against thermodynamic principles and understanding the numerical model of the mathematical framework.

Chapter 4 uses the validated numerical code to build a biological simulation and addresses the issues of how to build a finite element mesh of a protein from low resolution imaging data as well as determining the material parameters from Atomic Force Microscopy and measurements of protein viscosity using solvent based folding methods. Using these parameter values we have simulated the protein CoA[103] ligase and compare its RMSD with the RMSD found for proteins undergoing thermal fluctuations obtained from molecular dynamics[127]. The RMSD found from the coarse grained method and molecular dynamics are comparable and we conclude that the coarse grained model is producing reasonable dynamics for the molecule.

In chapter 5, I discussed using simulations to calculate the X-ray scattering curve from a biomolecule termed Tom1L1[138] and comparing it to the experimental curve. The results showed that the population average X-ray scattering curve from the Small Angle X-ray Scattering data was comparable to the simulated dynamically averaged X-ray scattering curve within the range of parameter values appropriate for biomolecules the X-ray scattering curve was independent of material parameters. This shows that at least in the case of this neglecting thermal fluctuations in building the low resolution structure from the experimental data is reasonable. However, it also shows that the experimental X-ray scattering curve does not contain dynamical information that could be used to further analyse the simulations.

In order to compare to dynamical information, chapter 6 considers using the molecular motor dynein[148][144] for which a low resolution structure exists and dynamical information about the flexibility of the motor. This allows for a more thorough comparison of the experimental data and simulated data and allowed me to extract material

parameters that match the experimental data. From this an inhomogeneous simulation of dynein was run that matches the overall reported dynamics for both of the motors biochemical states. A Young's modulus can be extracted for both the stalk and tail of dynein. These simulations show that the coarse grained model can replicate the correct distribution conformers of large globular proteins.

We can take the simulations of dynein further and actually place the molecular motor in situ within the axoneme[82] as shown in Figure 7.1. The image of the axoneme shown in Figure 7.1 is obtained from electron microscopy but highly flexible regions such as the stalk of dynein are averaged out in the image protein and are missing from the image. By using higher resolution imaging of dynein and the dynamics to parameterise dynein, the position of the stalk can be modelled back into the structure. This has important biological applications as the stalk binds to the neighbouring microtubule in order to generate motility in the axoneme.

These simulations allow us to monitor the position of the stalk head and in particular how close it comes to the neighbouring micro tubule in both biochemical states (See Figure 7.2). This gives hints as to the detailed motion of the motor. In the future, we will be able to generate a finite element mesh for the axoneme and resolve external forces to the motor and axoneme such as Van Der Waals, hydrodynamics and electrostatics as well as swap between the two biochemical states of dynein within a single simulation. Such a simulation would give a detailed description of how the motor functions, generates force and provide a window into how biology works in the mesoscale.

Figure 7.1: Image of the axoneme obtained through electron microscopy. The position of the dynein motor we have modelled in chapter 6 is identified along with the microtubule it binds to to generate force. Image obtained from Pigino et al[82].

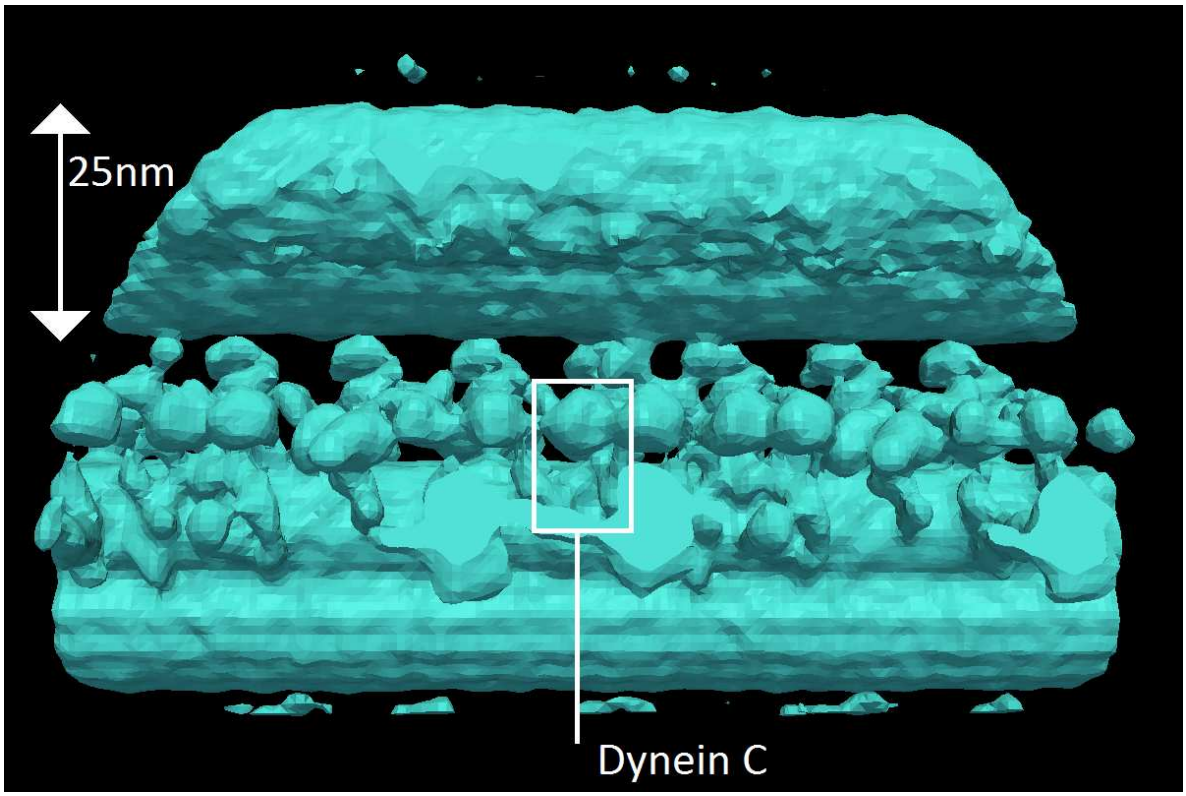
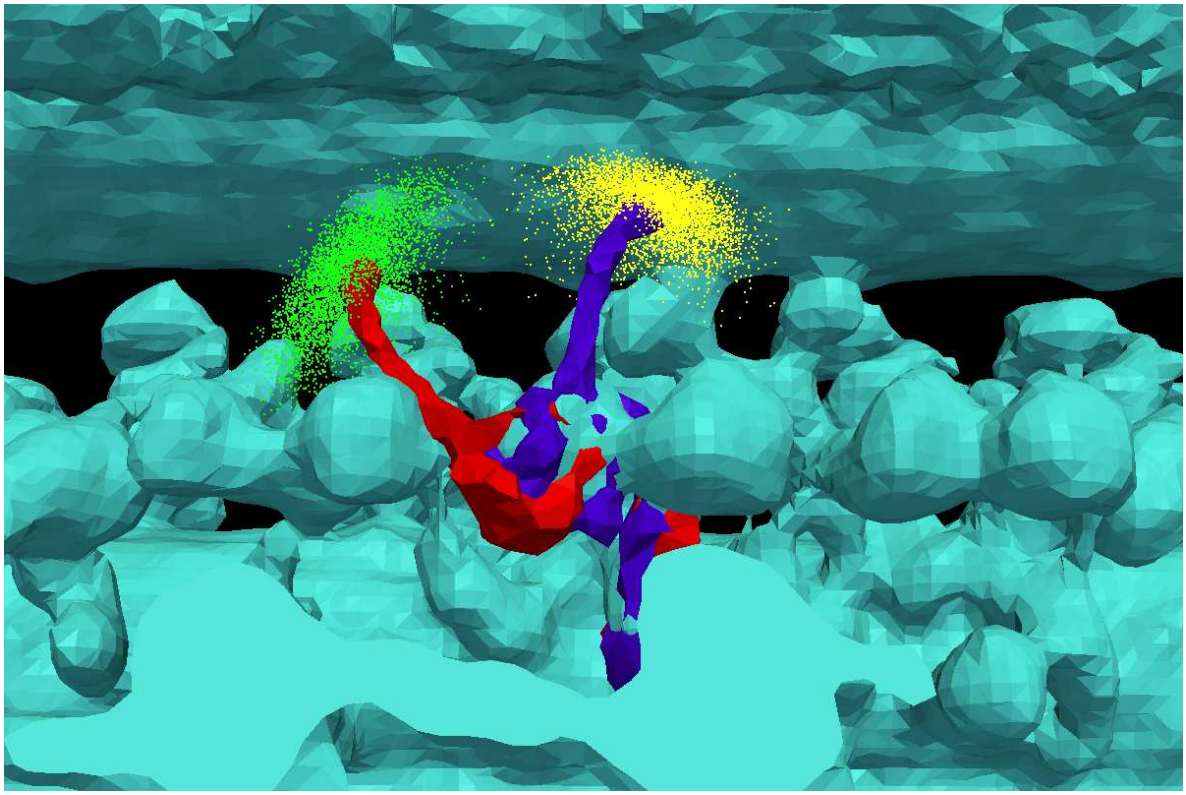


Figure 7.2: The axoneme[82] with simulations of APO (Blue) and ADPVI (Red) dynein shown with the stalk. The squares track the position of the APO stalk tip (Yellow) and ADPVI stalk tip (Green). The two distributions collide on different parts of the microtubule showing the potential for the motor to drag the microtubule during a power stroke.



Appendix A

Relationship to the Fokker-Planck Equation

In chapter 2, I introduced the Fokker-Planck equation as an additional means of testing the validity of the derived thermal noise terms in equations (2.29) and (2.36). I will now provide more detail on this derivation and the physical meaning of the Fokker-Planck equation.

A.1 The Fokker-Planck Equation

Let us consider a N-dimensional random vector X_p subject to a stochastic differential equation of the form:

$$dX_p = \mu_p(X_p, t)dt + B_{p\alpha}(X_p, t)dW_\alpha. \tag{A.1}$$

Then the time evolution of the probability distribution function $\psi(X_p, t)$ of the random vector X_p is given by the Fokker-Planck equation[88]:

$$\frac{\partial \psi}{\partial t} = -\frac{\partial}{\partial x_\alpha}(\mu_\alpha \psi) + \frac{1}{2}\frac{\partial}{\partial x_\alpha}\frac{\partial}{\partial x_\beta}(D_{\alpha\beta}\psi). \tag{A.2}$$

Where μ is called the drift vector and D_{pq} is called the diffusion tensor defined as:

$$D_{pq} = B_{p\alpha} B^T_{\alpha q}. \tag{A.3}$$

Equation (A.1) introduces the vector of Weiner processes dW_p, these are stochastic objects that represent the random processes of the system. Weiner processes have the following statistical properties:

$$\langle dW_p \rangle = 0, \text{and} \tag{A.4}$$

$$\langle dW_p dW_q \rangle = \delta_{pq} dt. \tag{A.5}$$

The Fokker-Planck equation is thus a kinetic theory approach to statistical mechanics and explicitly describes the time evolution of the probability distribution function of a random set of vectors X_p[154]. In order to use the Fokker-Planck equation to test the validity of the derived thermal noise, we must recast equation (2.14) using Weiner processes in the same form as equation (A.1) and then obtain the appropriate forms of the diffusivity matrix D_{pq} and the drift vector μ.

A.2 Recasting the Finite Element Equation of Motion

After finite element discretisation, the equation of motion for the system is given by:

$$M_{p\alpha} \frac{\partial v_\alpha}{\partial t} = N_p - K_{p\gamma} v_\gamma - \nabla_p U(\mathbf{x}). \tag{A.6}$$

Equation (A.6) can be re-arranged to give the infinitesimal change in velocity dv_p by inverting the mass matrix as follows:

$$dv_p = -M^{-1}_{p\alpha} K_{\alpha\beta} v_\beta dt - M^{-1}_{p\alpha} \nabla_\alpha U(\mathbf{x})\, dt + M^{-1}_{p\alpha} N_\alpha. \tag{A.7}$$

Equation (A.7) only characterises half the available space as not only are the velocities of the nodes stochastic the node positions are as well. The relationship between node position and velocity is given by:

$$dx_p = v_p dt. \tag{A.8}$$

The union of equations (A.7) and (A.8) provides a complete set of stochastic vectors that obey the Fokker-Planck equation wherein the drift vector μ is given by:

$$\mu_p = -M^{-1}_{p\alpha} K_{\alpha\beta} v_\beta dt - M^{-1}_{p\alpha} \nabla_\alpha U(\mathbf{x}) + v_p, \tag{A.9}$$

note that the drift vector picks up components from the velocity space and position space.

The derivation for the diffusion tensor D_{pq} is slightly more complicated and is achieved by rewriting equation (A.6) in terms of Weiner processes as follows:

$$dv_p = -M^{-1}_{p\alpha} K_{\alpha\beta} v_\beta dt - M^{-1}_{p\alpha} \nabla_\alpha U(\mathbf{x})\, dt + B_{p\alpha} dW_\alpha. \tag{A.10}$$

Wherein the relationship between the tensor B_{pq} and the noise force vector N_P is given by:

$$M^{-1}_{p\alpha} N_\alpha = B_{p\alpha} dW_\alpha. \tag{A.11}$$

The overall statistics of both the left and right hand side of equation (A.11) must be equivalent so that:

$$\langle B_{p\alpha} dW_\alpha B_{q\beta} dW_\beta \rangle = \langle M^{-1}_{p\alpha} N_\alpha dt M^{-1}_{q\beta} N_\beta dt \rangle, \tag{A.12}$$

this simplifies, using equation (2.29), to:

$$B_{p\alpha} B^T_{\alpha q} = k_B T M^{-1}_{p\alpha} M^{-1}_{q\beta} (K_{\alpha\beta} + K_{\beta\alpha}). \tag{A.13}$$

We have now derived the form of diffusion matrix D_{pq} in equation (A.13) and the drift vector μ in equation (A.9). The Fokker-Planck equation for the equation of motion (A.6) is therefore given by:

$$\frac{\partial \psi}{\partial t} = \frac{\partial}{\partial v_p}\left[\left(M^{-1}_{p\alpha}K_{\alpha\beta}v_\beta + M^{-1}_{p\alpha}\frac{\partial U(\mathbf{x})}{\partial x_\alpha}\right)\psi\right] - \frac{\partial}{\partial x_p}(v_p\psi) + \frac{1}{2}\frac{\partial}{\partial v_p}\frac{\partial}{\partial v_q}(D_{pq}\psi). \quad \text{(A.14)}$$

A.3 Solution of the Fokker-Planck Equation

At equilibrium, the probability distribution ψ is given by the Boltzmann distribution[87]:

$$\psi(\mathbf{x},\mathbf{v}) = A\exp\left(-\frac{U(\mathbf{x})}{k_\mathrm{B}T} - \frac{v_pM_{pq}v_q}{2k_\mathrm{B}T}\right). \quad \text{(A.15)}$$

All that is left is to show that the probability distribution ψ given in equation (A.15) satisfies the Fokker-Planck equation for the system. At equilibrium the probability distribution should not change in time and therefore the right hand side of equation (A.14) should be zero. Thus, we proceed by substitution and evaluate the terms of the Fokker-Planck equation:

$$\frac{\partial}{\partial v_p}\left(M^{-1}_{p\alpha}K_{\alpha\beta}v_\beta\psi\right) = M^{-1}_{p\alpha}K_{\alpha\beta}\delta_{p\beta}\psi - \frac{K_{\alpha\beta}v_\alpha}{k_BT}\psi, \quad \text{(A.16)}$$

$$\frac{\partial}{\partial v_p}\left(M^{-1}_{p\alpha}\frac{\partial U(\mathbf{x})}{\partial x_\alpha}\psi\right) = -\frac{v_\alpha}{k_BT}\frac{\partial U(\mathbf{x})}{\partial x_\alpha}\psi, \quad \text{(A.17)}$$

$$\frac{\partial}{\partial x_p}(v_p\psi) = -\frac{v_\alpha}{k_BT}\frac{\partial U(\mathbf{x})}{\partial x_\alpha}\psi, and \quad \text{(A.18)}$$

$$\frac{1}{2}\frac{\partial}{\partial v_p}\frac{\partial}{\partial v_q}(D_{pq}\psi) = -M^{-1}_{p\alpha}K_{\alpha\beta}\delta_{p\beta}\psi + \frac{K_{\alpha\beta}v_\alpha}{k_BT}\psi. \quad \text{(A.19)}$$

Equations (A.16)-(A.19) cancel out such that:

$$\frac{\partial \psi}{\partial t} = 0. \tag{A.20}$$

The result in equation (A.20) shows that the expected probability distribution function ψ for our stochastic system given in equation (A.15) is the static solution of the Fokker-Planck equation. This is entirely expected as the probability distribution function ψ was obtained by considering the case of thermal equilibrium and thus the distribution function should not be time dependent. From this, we conclude that the stochastic system constructed in Chapter 2 of this thesis conforms to the Fokker-Planck equation at thermal equilibrium.

A.4 Stochastic Finite Element Fortran Code

For completeness the main driving algorithm for the stochastic finite element model is provided. To compile the code the BLAS libraries are required as well as the Harwell subroutines for sparse matrix inversion and sparse matrix multiplication.

Figure A.1: Fortran 90 stochastic finite element code.

```
module RandomCons
implicit none
INTEGER, PARAMETER :: K4B=selected_int_kind(9)
REAL*8, SAVE :: am
INTEGER(K4B), SAVE :: ix=-1,iy=-1,KRAN
end module RandomCons

Use RandomCons
Implicit None
! Written by Robin Oliver, University of Leeds
! Program Divided into 3 main Sections, Main: Time Stepper + Outputs, Matrix Generators: Build the Matrices for finite element analysis and Subroutines. Augment the primary and secondary functions

!Main, Parameters that set the size of the system, NNodes = Number of Nodes, NElem = Number of Elements, n = total dimension of the system.

Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes

! Sparse Matrix Inverter, NMax= Maximum Dimension of the Mass Matrix, NZMAX= Total Number non-zero elements in the mass Matrix, MAXSP and MAXIW define the real and integer working space for the
Matrix Inverter if they're too small they should yield an error.

Integer, parameter :: NMAX=1000,NZMAX=6000,MAXSP=50000,MAXIW=50000

!Sparse Matrix Multiplier: LAELT: Maxiumum number of non zero elements in the matrix, NMAX2: Dimension of the Matrix
Integer, parameter :: LAELT=NElem*100 ,NMAX2=n

!Primary Definitions that allow the program to perform finite elements analysis
Real*8, Dimension(n,n) :: Visco
Real*8, Dimension(3) :: COM,ANGMOM
Real*8, Dimension(NNodes,6) :: Stress
Real*8, Dimension(n) :: Flucstress,ForceNV,ForceV,Vel,Vmid,VelD
Integer, dimension(NElem,4) :: Elem
Integer, dimension(NElem) :: Dummy
Real*8, dimension(NNodes,3) :: Nodes,TEMPNODES,ROT
Real*8, Dimension(n) :: Hyper
!Timer
Real*8, Dimension(16,5,3) :: WCor
Real*8, Dimension(18,5,3) :: BinT
Real*8, Dimension(16,5,3) :: Cor
Integer I,J,K,L,idum,Y
Real*8 DT,T,KB,ETA,ETAB,ENA,EN,EX,EY,EZ,EXA,EYA,EZA,ENPOT,EPA
Real*8, dimension(NElem) :: ACons,GCons

!Sparse Solver Definitions: Consult MA41 Help PDF for more info.
integer NE,JOB
integer, dimension(NZMAX) :: IRN,JCN
real*8, dimension(NZMAX) :: ASPK
real*8, dimension(NMAX) :: DuForX,DuForY,DuForZ,COLSCA,ROWSCA,VelDX,VelDY,VelDZ
integer, dimension(MAXIW) :: IW
integer, dimension(20) :: ICNTL,INFO
real*8, dimension(20) :: RINFO
integer, dimension(50) :: KEEP
Real*8, dimension(10) :: CNTL
Real*8, dimension(MAXSP) :: S

!Sparse Multiplier Definitions: Consult MC57 for more info.
Real*8, Dimension(LAELT) :: ASTOR
Integer, Dimension(LAELT) :: IRN2
Integer, Dimension(NMAX2+1) :: IP
Integer K1,K2,IKM

!Restart Code
Integer NStep,Restart
real*8 Time,Dummy1,Dummy2,Dummy3,Dummy4

!angular/Rotate
real*8 length,angle

!VTKwriter
CHARACTER(len=20) :: FN
integer NWRITE,ALWRITE

!Stokes_Drag
Real*8, Dimension(n) :: SDV,SPHR
real*8 etastokes

ALWRITE=0
NWRITE=0
NSTEP=0
Time=0.0d0

do I=1,n
SDV(I)=0.0d0
SPHR(I)=0.0d0
end do

open(400,file='ErrorEul.dat',status='unknown')

!IKM exists to perform simulations for different timesteps without recompiling
do IKM=1,1

!Primary Variables These should be stripped and read from an external file.

open(1110,file='ADPVI_Elastic_Corrected_1.dat',status='unknown')
do I=1,NElem
read(1110,*) GCons(I),ACons(I)
end do
close(1110)
```

Figure A.2: Fortran 90 stochastic finite element code.

```
do I=1,NElem
ACons(I)=ACons(I)*1.0d6
GCons(I)=GCons(I)*1.0d6
end do

DT=1.0d-14
T=3.0d2
KB=1.3806503d-23
ETA=1.0d-3
ETAB=1.0d-3
idum=-37267379
ENA=0.0d0
EPA=0.0d0
EXA=0.0d0
EYA=0.0d0
EZA=0.0d0
etastokes=0.001d-3
NE=0

open(1500,file='SPHR_LINK.dat',status='old')
do I=1,NNodes
read(1500,*) SPHR(I)
end do
close(1500)

do I=1,NNodes
SPHR(I+NNodes)=SPHR(I)
SPHR(I+2*NNodes)=SPHR(I)
end do

open(20,file='Vel.DAT',status='unknown')
open(30,file='Pos.DAT',status='unknown')
open(40,file='EnergyVer.out',status='unknown')
open(50,file='PositionsVer.d',status='unknown')
open(70,file='Str.str',status='unknown')
open(100,file='EnergyCorX.out',status='unknown')
open(101,file='EnergyCorY.out',status='unknown')
open(102,file='EnergyCorZ.out',status='unknown')
open(1000,file='Length_angle.dat',status='unknown')

!Read Mesh.dat:  See Read subroutine

Call Read(Elem,Dummy,Nodes)

!Prep for the first timestep: Establish the Rest Configuration of the Nodes in TEMPNODES and then set the initial velocity

do I=1,NNodes
   do J=1,3
      Nodes(I,J)=1.0d-10*Nodes(I,J)
   end do
end do

do I=1,NNodes
        do J=1,3
        TEMPNODES(I,J)=Nodes(I,J)
        end do
end do

do I=1,n
Vel(I)=0.0d0
end do

call Mass(Nodes,Elem,COM,TEMPNODES)

open(500,file='Restart.dat',status='old')
read(500,*) Restart
close(500)

if (Restart .eq. 1) then
        open(510,file='Restart.node',status='old')
        open(520,file='Restart.vel',status='old')
        open(530,file='RestartRan.dat',status='old')
        open(540,file='RestartValues.dat',status='old')

        do I=1,NNodes
        read(510,*) (Nodes(I,J),J=1,3)
        end do

        do I=1,3*NNodes
        read(520,*) Vel(I)
        end do

        read(530,*) idum,am,ix,iy,KRAN
        read(540,*) NStep,NWRITE,ALWRITE,TIME,ENA,EPA,EXA,EYA,EZA

        close(510)
        close(520)
        close(530)
        close(540)

        do I=1,ALWRITE
        read(1000,*)
        end do
end if
```

Figure A.3: Fortran 90 stochastic finite element code.

```
!Build the Matrices and obtain the Sparsity Pattern for the Mass Matrix and Viscosity Matrix
Y=1
call Matrices(Elem,Nodes,TEMPNODES,Visco,Hyper,Flucstress,DT,T,KB,ETA,ETAB,idum,NE,IRN,JCN,ASPK,Y,ASTOR,IP,IRN2,ACons,GCons)
Y=0

call stokesdrag(Vel,SPHR,etastokes,KB,T,DT,idum,SDV)

if (NE .gt. NZMAX) then
write(6,*) 'Too many non-zero matrix elements, increase Nmax size'
stop
end if

!Sparse Matrix Multiplier Note this version works only with a Matrices call preceeding

Call Multi3(Astor,IP,IRN2,Vel,VelD)

!Split Velocity into 3 components and invert based on block diagonal construction of the mass matrix

do I=1,NNodes
VelDX(I)=VelD(I)
VelDY(I)=VelD(I+NNodes)
VelDZ(I)=VelD(I+2*NNodes)
end do

do I=1,NNodes
VelDX(I)=1.0d0*VelDX(I)+1.0d0*Hyper(I)+1.0d0*Flucstress(I)-1.0d0*SDV(I)
VelDY(I)=1.0d0*VelDY(I)+1.0d0*Hyper(I+NNodes)+1.0d0*Flucstress(I+NNodes)-1.0d0*SDV(I+NNodes)
VelDZ(I)=1.0d0*VelDZ(I)+1.0d0*Hyper(I+2*NNodes)+1.0d0*Flucstress(I+2*NNodes)-1.0d0*SDV(I+2*NNodes)
end do

!Call to MA41 Control Routines Consult MA41 Help
call MA41ID(CNTL,ICNTL,KEEP)
!Set the Properties of the MA41 Inverter
ICNTL(10)=10
ICNTL(11)=1
JOB=6

!Invert The Mass MAtrix

call MA41AD (JOB,NNodes,NE,IRN,JCN,ASPK,VelDX,COLSCA,ROWSCA,KEEP,IW,MAXIW,S,MAXSP,CNTL,ICNTL,INFO,RINFO)
call MA41AD (3,NNodes,NE,IRN,JCN,ASPK,VelDY,COLSCA,ROWSCA,KEEP,IW,MAXIW,S,MAXSP,CNTL,ICNTL,INFO,RINFO)
call MA41AD (3,NNodes,NE,IRN,JCN,ASPK,VelDZ,COLSCA,ROWSCA,KEEP,IW,MAXIW,S,MAXSP,CNTL,ICNTL,INFO,RINFO)

!Euler Timestep of Velocity
do I=1,NNodes
Vel(I)=Vel(I)-DT*VelDX(I)
Vel(I+NNodes)=Vel(I+NNodes)-DT*VelDY(I)
Vel(I+2*NNodes)=Vel(I+2*NNodes)-DT*VelDZ(I)
end do

        !Call Analysis Routines: Note COM AND ANGMOM should be conserved
        call Mass(Nodes,Elem,COM,TEMPNODES)
        call Angmomentum(Nodes,TEMPNODES,Elem,Vel,COM,ANGMOM)

        write(6,*) 'COM',(COM(I),I=1,3)
        write(6,*) 'ANGMOM',(ANGMOM(I),I=1,3)


!Euler Timestep of Position
do I=1,NNodes
        do J=1,3
        Nodes(I,J)=Nodes(I,J)+DT*Vel(I+(J-1)*NNodes)
        end do
end do

NSTEP=NSTEP+1
TIME=TIME+DT

! Proceed into the main Dynamics Loop, The above is merely a timestep to obtain the sparsity patterns
do K=1,5000000
        NSTEP=NSTEP+1
        Time=Time+DT


        ! Sparsity Pattern Has Been Calculated So Call Matrices 2
        call Matrices2(Elem,Nodes,TEMPNODES,Visco,Hyper,Flucstress,DT,T,KB,ETA,ETAB,idum,NE,IRN,JCN,ASPK,Y,Stress,Vmid,ACons,GCons,ENPOT)
        call stokesdrag(Vel,SPHR,etastokes,KB,T,DT,idum,SDV)

        !Proceed with Euler TimeStep
        do I=1,n
        ForceNV(I)=1.0d0*Hyper(I)+1.0d0*Flucstress(I)
        end do

        !Matrices 2 Called so Use Multi4 since we have the Sparsity Pattern of the viscosity matrix
        Call Multi4(Visco,IP,IRN2,Vel,ForceV)

        do J=1,NNodes
        DuForX(J)= 1.0d0*ForceV(J)+1.0d0*ForceNV(J)-1.0d0*SDV(J)
        DuForY(J)= 1.0d0*ForceV(J+NNodes)+1.0d0*ForceNV(J+NNodes)-1.0d0*SDV(J+NNodes)
        DuForZ(J)= 1.0d0*ForceV(J+2*NNodes)+1.0d0*ForceNV(J+2*NNodes)-1.0d0*SDV(J+2*NNodes)
        end do

        !MAtrix Inversion
        call MA41AD (3,NNodes,NE,IRN,JCN,ASPK,DuForX,COLSCA,ROWSCA,KEEP,IW,MAXIW,S,MAXSP,CNTL,ICNTL,INFO,RINFO)
        call MA41AD (3,NNodes,NE,IRN,JCN,ASPK,DuForY,COLSCA,ROWSCA,KEEP,IW,MAXIW,S,MAXSP,CNTL,ICNTL,INFO,RINFO)
```

Figure A.4: Fortran 90 stochastic finite element code.

```
call MA41AD (3,NNodes,NE,IRN,JCN,ASPK,DuForZ,COLSCA,ROWSCA,KEEP,IW,MAXIW,S,MAXSP,CNTL,ICNTL,INFO,RINFO)

do I=1,NNodes
DuForX(I)=DuForX(I)
DuForY(I)=DuForY(I)
DuForZ(I)=DuForZ(I)
end do

!Euler Timestep Velocity
do J=1,NNodes
Vel(J)=Vel(J)-(DT)*DuForX(J)
Vel(J+NNodes)=Vel(J+NNodes)-(DT)*DuForY(J)
Vel(J+2*NNodes)=Vel(J+2*NNodes)-(DT)*DuForZ(J)
end do

call Mass(Nodes,Elem,COM,TEMPNODES)
call Angmomentum(Nodes,TEMPNODES,Elem,Vel,COM,ANGMOM)

!Euler Timestep Position
do I=1,NNodes
        do J=1,3
        Nodes(I,J)=Nodes(I,J)+DT*Vel(I+(J-1)*NNodes)
        end do
end do

do I=1,NNodes
        do J=1,3
        Nodes(I,J)=Nodes(I,J)-COM(J)
        end do
end do

!Calculations the Kinetic Energy, Potential Energy is calculated in the Matrices subroutine
call Energy(ASPK,IRN,JCN,Vel,En,EX,EY,EZ)

if ((mod(NSTEP,1000000) .eq. 0))  then
write(6,*) NSTEP
write(6,*) ENA,EN,EXA,EYA,EZA,EPA,ENPOT
write(6,*) EN+ENPOT
write(6,*) 'COM',(COM(I),I=1,3)
write(6,*) 'ANGMOM',(ANGMOM(I),I=1,3)
end if

!Energy Averages
if (NSTEP .ge. 400000) then
ENA= (ENA*(NSTEP-400000)+En)/((NSTEP-399999)*1.0d0)
EXA= (EXA*(NSTEP-400000)+EX)/((NSTEP-399999)*1.0d0)
EYA= (EYA*(NSTEP-400000)+EY)/((NSTEP-399999)*1.0d0)
EZA= (EZA*(NSTEP-400000)+EZ)/((NSTEP-399999)*1.0d0)
        if (NSTEP .ge. 400000) then
        EPA= (EPA*(NSTEP-400000)+ENPOT)/((NSTEP-399999)*1.0d0)
        end if
end if

if (mod(NSTEP,1000) .eq. 0) then
call Coordinate(Nodes,Elem,length,angle,ROT)
ALWRITE=ALWRITE+1
write(1000,*) NStep,length,angle
end if

if (mod(NSTEP,10000) .eq. 0) then
NWRITE=NWRITE+1
WRITE(FN,10) NWRITE
WRITE(6,*)FN
OPEN(1,FILE=FN)

write(1,'(a)') '# vtk DataFile Version 1.0'
write(1,'(a)') '2D Unstructured Grid of Linear Triangles'
write(1,'(a)') 'ASCII'
write(1,*)
write(1,'(a)') 'DATASET UNSTRUCTURED_GRID'
write(1,'("POINTS",1x,i3,1x,"float")') NNodes
do I=1,NNodes
write(1,'(f8.3,2X,f8.3,2X,f8.3)') (1.0d10*Rot(I,J),J=1,3)
end do
write(1,*)
write(1,'("CELLS",1x,I6,1x,I6)') NElem,5*NElem
do I=1,NELEM
write(1,'("4",2x,I3,2x,I3,2x,I3,2x,I3)') (Elem(I,J)-1,J=1,4)
end do
write(1,*)
write(1,'("CELL_TYPES",1x,i6)') NElem
do I=1,NELEM
write(1,'(I2)') 10
end do
write(1,*)
write(1,'("POINT_DATA",1x,i6)') NNodes
write(1,'(a)') 'VECTORS velocity float'
do I=1,NNodes
write(1,'(f7.2,2X,f7.2,2X,f7.2)') (1.0d10*(Nodes(I,J)-TEMPNODES(I,J)),J=1,3)
end do

CLOSE(1)
10 FORMAT('Dyn',I8.8,'.vtk')
end if
```

Figure A.5: Fortran 90 stochastic finite element code.

```
        end if

        if (mod(NSTEP,10000) .eq. 0) then
        open(500,file='Restart.dat',status='old')
        write(500,*) 1
        close(500)

        open(510,file='Restart.node',status='unknown')
        open(520,file='Restart.vel',status='unknown')
        open(530,file='RestartRan.dat',status='unknown')
        open(540,file='RestartValues.dat',status='unknown')

        do I=1,NNodes
        write(510,*) (Nodes(I,J),J=1,3)
        end do

        do I=1,3*NNodes
        write(520,*) Vel(I)
        end do

        write(530,*) idum,am,ix,iy,KRAN
        write(540,*) NStep,NWRITE,ALWRITE,TIME,ENA,EPA,EXA,EYA,EZA

        call flush(500)
        call flush(510)
        call flush(520)
        call flush(530)
        call flush(540)
        call flush(1000)

        close(510)
        close(520)
        close(530)
        close(540)

        end if

end do
close(20)
close(30)
close(40)
close(70)
write(400,*) ENA,EPA,EPA+ENA,DT
end do
close(400)
close(1000)
end

Subroutine Matrices(Elem,Nodes,TEMPNODES,Visco,Hyper,Flucstress,DT,T,KB,ETA,ETAB,idum,NE,IRN,JCN,ASPK,Y,ASTOR,IP,IRN2,ACons,GCons)
!Call This Subroutine First in any solution, this subroutine calculates the sparsity patterns and the Mass Matrix.
implicit none
!Standard Definitions
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
Integer, parameter :: NZMAX=6000
Integer, parameter :: LAELT=NElem*100 ,NMAX2=n ,NZMAX2=NElem*78
Real*8, Dimension(n,n) :: Visco
Real*8, Dimension(n) :: Hyper,Flucstress
Integer, dimension(NElem,4) :: Elem
Real*8, dimension(NNodes,3) :: Nodes,TEMPNODES
Real*8, dimension(3,3) :: Jac
Real*8, dimension(3,3) :: JacIn
Real*8, dimension(3,3) :: Old,New,Oldin,A,F,FT,Random
Real*8, dimension(3,4) :: Grad
Real*8, dimension(3) :: BinA,BinB,BinC,BinD,Bin,OldA,OldB,OldC,OldD
Real*8, dimension(4) :: XCOMP,YCOMP,ZCOMP
Real*8 Vol,Vol0,T,KB,ETA,ETAB,DT,P,Rand,Tr,DUM,Contract,EnBulk,EnPres,Guage,EP
Real*8, dimension(NElem) :: ACons,GCons
Real*8 ran1
Integer I,J,K,Q,W,idum,Y

!Mass
real*8, dimension(NNodes,NNodes) :: Mass,Mass2
real*8 VolT

!Sparse Inverter
integer NE,D,O
integer, dimension(NZMAX) :: IRN,JCN
real*8, dimension(NZMAX) :: ASPK

!Sparse Multiplier See MC57 Help
Real*8, Dimension(LAELT) :: AELT, ASTOR
Integer, Dimension(NZMAX2) :: ELTVAR
Integer, Dimension(NElem+1) :: ELTPTR
Integer, Dimension(LAELT) :: IRN2
Integer, Dimension(10) :: INFO
Integer, Dimension(NMAX2+1) :: IP
Integer, Dimension(2*NMAX2) :: IW
Logical, Dimension(10) :: LCNTL
integer NE2,NE3,LIRN,LA,NORD,LP

EP=0.0d0
VolT=0.0d0

!Control Parameters for the Sparse Multiplier
LP=6
NE2=0
NE3=0
```

Figure A.6: Fortran 90 stochastic finite element code.

```
LCNTL(1)= .TRUE.
LCNTL(2)= .TRUE.
LCNTL(3)= .TRUE.
LCNTL(4)= .TRUE.
LIRN=NElem*78
LA=NElem*78

do I=1,NELEM+1
ELTPTR(I)=1+12*(I-1)
end do

!Sparse Matrix Setup
if (Y .eq. 1) then
NE=0
do I=1,NZMAX
IRN(I)=0
JCN(I)=0
ASPK(I)=0.0d0
end do
do I=1,LAELT
AELT(I)=0.0d0
ASTOR(I)=0.0d0
end do
do I=1,NZMAX2
ELTVAR(I)=0
end do

end if

!Set 3x3 matrices used form element calculations to zero
do I=1,3
        do J=1,3
        Jac(I,J)=0.0d0
        Old(I,J)=0.0d0
        New(I,J)=0.0d0
        Oldin(I,J)=0.0d0
        Random(I,J)=0.0d0
        end do
end do

!Set arrays and full matrices to zero
do I=1,n
        do J=1,n
        Visco(I,J)=0.0d0
        end do
Hyper(I)=0.0d0
flucstress(I)=0.0d0
end do
do I=1,NNodes
        do J=1,NNodes
        Mass(I,J)=0.0d0
        Mass2(I,J)=0.0d0
        end do
end do

!Main Section of Matrices subroutine, calculate all matrices and arrays associated with one element then add and loop over all elements
do J=1,NELEM
        !Calculate Jacobian Transformation between the parametrised coordinate system and cartesian coordiante system
        do I=1,3
        BinA(I)=Nodes(Elem(J,1),I)-Nodes(Elem(J,2),I)
        BinB(I)=Nodes(Elem(J,1),I)-Nodes(Elem(J,3),I)
        BinC(I)=Nodes(Elem(J,1),I)-Nodes(Elem(J,4),I)
        Jac(1,I)=-BinA(I)
        Jac(2,I)=-BinB(I)
        Jac(3,I)=-BinC(I)
        end do

        !Calculate the Transpose the of the Jacobian used to calculate the Deformation Gradient Tensor
        do I=1,3
                do K=1,3
                New(I,K)=Jac(K,I)
                end do
        end do

        !Calculate the Volume of an Element
        call VP(BinA,BinB,BinD)

        Vol=(1.0d0/6.0d0)*Dot_Product(BinD,BinC)
                if (Vol .LT. 0.0d0) then
                call VP(BinB,BinA,BinD)
                Vol=(1.0d0/6.0d0)*Dot_Product(BinD,BinC)
                end if

        !Invert the Jacobian using the method of minor determinants
        call inverse(Jac, JacIn)

        ! Compute the Original Position Vectors between node sin an element, used to Calculate the deformation gradient tensor
        do I=1,3
        Old(I,1)=-TEMPNODES(Elem(J,1),I)+TEMPNODES(Elem(J,2),I)
        Old(I,2)=-TEMPNODES(Elem(J,1),I)+TEMPNODES(Elem(J,3),I)
        Old(I,3)=-TEMPNODES(Elem(J,1),I)+TEMPNODES(Elem(J,4),I)
        OldA(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,2),I)
        OldB(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,3),I)
        OldC(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,4),I)
        end do

        !Compute Original Volume
```

Figure A.7: Fortran 90 stochastic finite element code.

```
        call VP(OldA,OldB,OldD)

        Vol0=(1.0d0/6.0d0)*Dot_Product(OldD,OldC)
                if (Vol0 .LT. 0.0d0) then
                call VP(OldB,OldA,OldD)
                Vol0=(1.0d0/6.0d0)*Dot_Product(OldD,OldC)
                end if

        !Invert of the old matrix
        call inverse (Old,Oldin)

        !Compute F for element J
        call Multiply(New,Oldin,F)


        !Compute F transpose
        do I=1,3
                do K=1,3
                FT(I,K)=F(K,I)
                end do
        end do

        !Compute A, this gives us the rubber elasicity matrix for a single element
        call Multiply(F,FT,A)

        !First Term in Bulk
        do I=1,3
                do K=1,3
                A(I,K)= 1.0d0*(Vol0/Vol)*A(I,K)
                end do
        end do

        VolT=Vol0+VolT

        !Subtract Identity in line witht he equation of motion
        A(1,1)=(A(1,1)-1.0d0)
        A(2,2)=(A(2,2)-1.0d0)
        A(3,3)=(A(3,3)-1.0d0)

        do I=1,3
                do K=1,3
                A(I,K)= GCons(J)*A(I,K)
                end do
        end do

        !Subtract Pressure term
        A(1,1)= A(1,1)+ACons(J)*((Vol-Vol0)/Vol0)
        A(2,2)= A(2,2)+ACons(J)*((Vol-Vol0)/Vol0)
        A(3,3)= A(3,3)+ACons(J)*((Vol-Vol0)/Vol0)

        call Elastic(F,Vol0,Vol,ACons,GCons,EP,J)

        !Fluctuating stress terms (Shear Viscosity)
        Random(1,1)=sqrt((48.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
        Random(2,2)=sqrt((48.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
        Random(3,3)=sqrt((48.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
        Random(1,2)=sqrt((24.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
        Random(1,3)=sqrt((24.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
        Random(2,3)=sqrt((24.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
        Random(2,1)=Random(1,2)
        Random(3,1)=Random(1,3)
        Random(3,2)=Random(2,3)

        !Bulk Viscosity Fluctuating Term
        P=sqrt((24.0d0*KB*T*ETAB)/(Vol*DT))*(ran1(idum)-0.5d0)
        Random(1,1)=Random(1,1)+P
        Random(2,2)=Random(2,2)+P
        Random(3,3)=Random(3,3)+P

        !XCOMP, YCOMP and ZCOMP are essentially the shape functions differentiated with respect to x,y and z. EG. YCOMP(3) is the third shape function phi=t          differenetiated with
respect to Y
        XCOMP(1)=-JacIn(1,1)-JacIn(1,2)-JacIn(1,3)
        XCOMP(2)=JacIn(1,1)
        XCOMP(3)=JacIn(1,2)
        XCOMP(4)=JacIn(1,3)
        YCOMP(1)=-JacIn(2,1)-JacIn(2,2)-JacIn(2,3)
        YCOMP(2)=JacIn(2,1)
        YCOMP(3)=JacIn(2,2)
        YCOMP(4)=JacIn(2,3)
        ZCOMP(1)=-JacIn(3,1)-JacIn(3,2)-JacIn(3,3)
        ZCOMP(2)=JacIn(3,1)
        ZCOMP(3)=JacIn(3,2)
        ZCOMP(4)=JacIn(3,3)

        !Now Compute the Mass and Viscosity Matrix in a form readable by the Spare MAtrix subroutines
        ! The Mass MAtrix Sparsity pattern is done by computing the relevant entry and reading its row and column number
        ! The Row and Column Numbers are then checked to see if there is an enrty already existing in that location
        ! IF an entry isn't found the actual matrix entry is placed into ASPK and the row and column number stored in IRN and JCN
        ! IF an Entry is found with the same row and column number the new value is added to that value already stored in ASPK
        do W=1,4
                do Q=1,4
                        ! Sparse Matrix Multiplier Entry
                        ! This is done by considering the 9 4x4 submatrices of the full Viscosity matrix
                        ! We aim to read the lower triangular portion of the full viscosity matrix only
                        ! So we Read column 1 row by row then column 2 etc.
```

Figure A.8: Fortran 90 stochastic finite element code.

```
                        ! This data structure mimicks that by entering data in that order.
                        ! See MC57 details for how the routine requires data to be entered
                        if (Q .ge. W) then
                        NE3=NE3+1
                        DUM=(XCOMP(Q)*XCOMP(W)+YCOMP(Q)*YCOMP(W)+ZCOMP(Q)*ZCOMP(W))*Vol*ETA
                        AELT(NE3)=(ETA+ETAB)*Vol*(XCOMP(Q)*XCOMP(W))+DUM
                                if (W .eq. 1) then
                                NE2=NE2+1
                                ELTVAR(NE2) = Elem(J,Q)

                                end if
                        end if
                end do

                do Q=1,4
                NE3=NE3+1
                AELT(NE3)=(ETAB)*Vol*YCOMP(Q)*XCOMP(W)+(ETA)*Vol*YCOMP(W)*XCOMP(Q)
                        if (W .eq. 1) then
                        NE2=NE2+1
                        ELTVAR(NE2) = Elem(J,Q)+NNodes
                        end if
                end do

                do Q=1,4
                NE3=NE3+1
                AELT(NE3)=(ETAB)*Vol*ZCOMP(Q)*XCOMP(W)+(ETA)*Vol*ZCOMP(W)*XCOMP(Q)
                        if (W .eq. 1) then
                        NE2=NE2+1
                        ELTVAR(NE2) = Elem(J,Q)+2*NNodes

                        end if
                end do

        !Compute The Elastic Vector (Hyper) and the Fluctuating Stress Vector
        Hyper(Elem(J,W))=Vol*XCOMP(W)*A(1,1)+Hyper(Elem(J,W))
        Hyper(Elem(J,W))=Vol*YCOMP(W)*A(2,1)+Hyper(Elem(J,W))
        Hyper(Elem(J,W))=Vol*ZCOMP(W)*A(3,1)+Hyper(Elem(J,W))
        Hyper(Elem(J,W)+NNodes)=Vol*XCOMP(W)*A(1,2)+Hyper(Elem(J,W)+NNodes)
        Hyper(Elem(J,W)+NNodes)=Vol*YCOMP(W)*A(2,2)+Hyper(Elem(J,W)+NNodes)
        Hyper(Elem(J,W)+NNodes)=Vol*ZCOMP(W)*A(3,2)+Hyper(Elem(J,W)+NNodes)
        Hyper(Elem(J,W)+2*NNodes)=Vol*XCOMP(W)*A(1,3)+Hyper(Elem(J,W)+2*NNodes)
        Hyper(Elem(J,W)+2*NNodes)=Vol*YCOMP(W)*A(2,3)+Hyper(Elem(J,W)+2*NNodes)
        Hyper(Elem(J,W)+2*NNodes)=Vol*ZCOMP(W)*A(3,3)+Hyper(Elem(J,W)+2*NNodes)

        Flucstress(Elem(J,W))=Vol*XCOMP(W)*Random(1,1)+Flucstress(Elem(J,w))
        Flucstress(Elem(J,W))=Vol*YCOMP(W)*Random(2,1)+Flucstress(Elem(J,W))
        Flucstress(Elem(J,W))=Vol*ZCOMP(W)*Random(3,1)+Flucstress(Elem(J,W))
        Flucstress(Elem(J,W)+NNodes)=Vol*XCOMP(W)*Random(1,2)+Flucstress(Elem(J,W)+NNodes)
        +lucstress(Elem(J,W)+NNodes)=Vol*YCUM+(W)*Random(2,2)++Lucstress(Elem(J,W)+NNodes)
        Flucstress(Elem(J,W)+NNodes)=Vol*ZCOMP(W)*Random(3,2)+Flucstress(Elem(J,W)+NNodes)
        Flucstress(Elem(J,W)+2*NNodes)=Vol*XCOMP(W)*Random(1,3)+Flucstress(Elem(J,W)+2*NNodes)
        Flucstress(Elem(J,W)+2*NNodes)=Vol*YCOMP(W)*Random(2,3)+Flucstress(Elem(J,W)+2*NNodes)
        Flucstress(Elem(J,W)+2*NNodes)=Vol*ZCOMP(W)*Random(3,3)+Flucstress(Elem(J,W)+2*NNodes)

        end do

        !Complete the Viscosity Pattern Entry
        do W=1,4
                do Q=1,4
                if (Q .ge. W) then
                NE3=NE3+1
                DUM=(XCOMP(Q)*XCOMP(W)+YCOMP(Q)*YCOMP(W)+ZCOMP(Q)*ZCOMP(W))*Vol*ETA
                AELT(NE3)=(ETA+ETAB)*Vol*YCOMP(Q)*YCOMP(W)+DUM
                end if
                end do

                do Q=1,4
                NE3=NE3+1
                AELT(NE3)=(ETAB)*Vol*ZCOMP(Q)*YCOMP(W)+(ETA)*Vol*ZCOMP(W)*YCOMP(Q)
                end do
        end do

        do W=1,4
                do Q=1,4
                if (Q .ge. W) then
                NE3=NE3+1
                DUM=(XCOMP(Q)*XCOMP(W)+YCOMP(Q)*YCOMP(W)+ZCOMP(Q)*ZCOMP(W))*Vol*ETA
                AELT(NE3)=(ETA+ETAB)*Vol*ZCOMP(Q)*ZCOMP(W)+DUM
                end if
                end do
        end do

        do Q=1,4
                do W=1,4
                if (Q .eq. W) then
                Mass(Elem(J,Q),Elem(J,W))=Mass(Elem(J,Q),Elem(J,W))+1.5d3*Vol0/10.0d0
                elseif (Q .ne. W) then
                Mass(Elem(J,Q),Elem(J,W))=Mass(Elem(J,Q),Elem(J,W))+1.5d3*Vol0/20.0d0
                end if
                end do
        end do

end do

call MC57AD(LCNTL,NMAX2,NELEM,ELTVAR,ELTPTR,AELT,NORD,LIRN,IRN2,IP,LA,ASTOR,IW,LP,INFO)

do J=1,NNodes
```

Figure A.9: Fortran 90 stochastic finite element code.

```
        do K=1,NNodes
                if (abs(Mass(J,K)) .gt. 0.0d0) then
                NE=NE+1
                IRN(NE)=J
                JCN(NE)=K
                ASPK(NE)=Mass(J,K)
                end if
        end do
end do

end subroutine

Subroutine Matrices2(Elem,Nodes,TEMPNODES,Visco,Hyper,Flucstress,DT,T,KB,ETA,ETAB,idum,NE,IRN,JCN,ASPK,Y,Stress,Vmid,ACons,GCons,EP)
!Call Matrices 2 After Matrices 1.
!No longer need to calculate the sparsity pattern we simply calculate matrices
implicit none
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
Integer, parameter :: NZMAX=6000
Real*8, Dimension(n,n) :: Visco
Real*8, Dimension(NNodes,6) :: Stress
Real*8, Dimension(n) :: Hyper,Flucstress,Vmid
Integer, dimension(NElem,4) :: Elem
Real*8, dimension(NNodes,3) :: Nodes,TEMPNODES
Real*8, dimension(3,3) :: Jac
Real*8, dimension(3,3) :: JacIn
Real*8, Dimension(3,3) :: Old,New,Oldin,A,F,FT,Random
Real*8, dimension(3,4) :: Grad
Real*8, dimension(3) :: BinA,BinB,BinC,BinD,Bin,OldA,OldB,OldC,OldD
Real*8, dimension(4) :: XCOMP,YCOMP,ZCOMP
Real*8 Vol,Vol0,T,KB,ETA,ETAB,DT,P,Rand,Tr,DUM,Contract,EnBulk,EnPres,Guage,EP
Real*8, dimension(NElem) :: ACons,GCons
Real*8 ran1
Integer I,J,K,Q,W,idum,Y

integer NE,D,O
integer, dimension(NZMAX) :: IRN,JCN
real*8, dimension(NZMAX) :: ASPK

EP=0.0d0

!Same Steup as Before See Matrices 1 for comments.

if (Y .eq. 1) then
NE=0
do I=1,NZMAX
IRN(I)=0
JCN(I)=0
ASPK(I)=0.0d0
end do
end if

do I=1,NNodes
        do J=1,6
        Stress(I,J)=0.0d0
        end do
end do

do I=1,3
        do J=1,3
        Jac(I,J)=0.0d0
        Old(I,J)=0.0d0
        New(I,J)=0.0d0
        Oldin(I,J)=0.0d0
        Random(I,J)=0.0d0
        end do
end do

do I=1,n
        do J=1,n
        Visco(I,J)=0.0d0
        end do
Hyper(I)=0.0d0
Flucstress(I)=0.0d0
end do

!Main Algorithm, computes volume of an element and builds the mass/stiffness matrices.
do J=1,NElem
        do I=1,3
        BinA(I)=Nodes(Elem(J,1),I)-Nodes(Elem(J,2),I)
        BinB(I)=Nodes(Elem(J,1),I)-Nodes(Elem(J,3),I)
        BinC(I)=Nodes(Elem(J,1),I)-Nodes(Elem(J,4),I)
        Jac(1,I)=-BinA(I)
        Jac(2,I)=-BinB(I)
        Jac(3,I)=-BinC(I)
        end do

        do I=1,3
                do K=1,3
                New(I,K)=Jac(K,I)
                end do
        end do

        call VP(BinA,BinB,BinD)

        Vol=(1.0d0/6.0d0)*Dot_Product(BinD,BinC)
                if (Vol .LT. 0.0d0) then
                call VP(BinB,BinA,BinD)
```

Figure A.10: Fortran 90 stochastic finite element code.

```
        if (Vol .LT. 0.0d0) then
        call VP(BinB,BinA,BinD)
        Vol=(1.0d0/6.0d0)*Dot_Product(BinD,BinC)
        end if

call inverse(Jac, JacIn)

! Compute the stable 'Old' F matrix
do I=1,3
Old(I,1)=-TEMPNODES(Elem(J,1),I)+TEMPNODES(Elem(J,2),I)
Old(I,2)=-TEMPNODES(Elem(J,1),I)+TEMPNODES(Elem(J,3),I)
Old(I,3)=-TEMPNODES(Elem(J,1),I)+TEMPNODES(Elem(J,4),I)
OldA(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,2),I)
OldB(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,3),I)
OldC(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,4),I)
end do

call VP(oldA,oldB,oldD)

Vol0=(1.0d0/6.0d0)*Dot_Product(OldD,oldC)
        if (Vol0 .LT. 0.0d0) then
        call VP(oldB,oldA,oldD)
        Vol0=(1.0d0/6.0d0)*Dot_Product(oldD,oldC)
        end if

!Invert of the old matrix
call inverse (Old,Oldin)

!Compute F for element J
call Multiply(New,oldin,F)

!Compute F transpose
do I=1,3
        do K=1,3
        FT(I,K)=F(K,I)
        end do
end do

!Compute A
call Multiply(F,FT,A)

!First Term in Bulk
do I=1,3
        do K=1,3
        A(I,K)= 1.0d0*(Vol0/Vol)*A(I,K)
        end do
end do

!Subtract Identity
A(1,1)=(A(1,1)-1.0d0)
A(2,2)=(A(2,2)-1.0d0)
A(3,3)=(A(3,3)-1.0d0)

do I=1,3
        do K=1,3
        A(I,K)= GCons(J)*A(I,K)
        end do
end do

!Subtract Pressure term
A(1,1)= A(1,1)+ACons(J)*((Vol-Vol0)/Vol0)
A(2,2)= A(2,2)+ACons(J)*((Vol-Vol0)/Vol0)
A(3,3)= A(3,3)+ACons(J)*((Vol-Vol0)/Vol0)

call Elastic(F,Vol0,Vol,ACons,GCons,EP,J)

!Fluctuating stress terms (Shear Viscosity)
Random(1,1)=sqrt((48.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
Random(2,2)=sqrt((48.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
Random(3,3)=sqrt((48.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
Random(1,2)=sqrt((24.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
Random(1,3)=sqrt((24.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
Random(2,3)=sqrt((24.0d0*KB*T*ETA)/(Vol*DT))*(ran1(idum)-0.5d0)
Random(2,1)=Random(1,2)
Random(3,1)=Random(1,3)
Random(3,2)=Random(2,3)

!Bulk Viscosity
P=sqrt((24.0d0*KB*T*ETAB)/(Vol*DT))*(ran1(idum)-0.5d0)
Random(1,1)=Random(1,1)+P
Random(2,2)=Random(2,2)+P
Random(3,3)=Random(3,3)+P

do Q=1,4
        do W=1,4
                !Under Most circumstances this isn't accessed as the Mass Matrix is constant in time.
                !IF for any reason the mass Matrix needs to be recalculated then set Y=1 in the Integrator Routine In Main
                XCOMP(1)=-JacIn(1,1)-JacIn(1,2)-JacIn(1,3)
                XCOMP(2)=JacIn(1,1)
                XCOMP(3)=JacIn(1,2)
                XCOMP(4)=JacIn(1,3)
                YCOMP(1)=-JacIn(2,1)-JacIn(2,2)-JacIn(2,3)
                YCOMP(2)=JacIn(2,1)
                YCOMP(3)=JacIn(2,2)
                YCOMP(4)=JacIn(2,3)
                ZCOMP(1)=-JacIn(3,1)-JacIn(3,2)-JacIn(3,3)
                ZCOMP(2)=JacIn(3,1)
```

Figure A.11: Fortran 90 stochastic finite element code.

```
                        ZCOMP(3)=JacIn(3,2)
                        ZCOMP(4)=JacIn(3,3)

                        !Calculate Lower Triangle of the Viscosity matrix
                        if (Elem(J,Q) .ge. Elem(J,W)) then
                        DUM=(XCOMP(Q)*XCOMP(W)+YCOMP(Q)*YCOMP(W)+ZCOMP(Q)*ZCOMP(W))*Vol*ETA
                        Visco(Elem(J,Q),Elem(J,W))=(ETA+ETAB)*Vol*(XCOMP(Q)*XCOMP(W))+DUM+Visco(Elem(J,Q),Elem(J,W))
                        Visco(Elem(J,Q)+NNodes,Elem(J,W)+NNodes)=(ETA+ETAB)*Vol*YCOMP(Q)*YCOMP(W)+DUM+Visco(Elem(J,Q)+NNodes,Elem(J,W)+NNodes)
                        Visco(Elem(J,Q)+2*NNodes,Elem(J,W)+2*NNodes)=(ETA+ETAB)*Vol*ZCOMP(Q)*ZCOMP(W)+DUM+Visco(Elem(J,Q)+2*NNodes,Elem(J,W)+2*NNodes)
                        end if

                        Visco(Elem(J,Q)+NNodes,Elem(J,W))=(ETAB)*Vol*YCOMP(Q)*XCOMP(W)+Visco(Elem(J,Q)+NNodes,Elem(J,W))
                        Visco(Elem(J,Q)+2*NNodes,Elem(J,W))=(ETAB)*Vol*ZCOMP(Q)*XCOMP(W)+Visco(Elem(J,Q)+2*NNodes,Elem(J,W))
                        Visco(Elem(J,Q)+2*NNodes,Elem(J,W)+NNodes)=(ETAB)*Vol*ZCOMP(Q)*YCOMP(W)+Visco(Elem(J,Q)+2*NNodes,Elem(J,W)+NNodes)

                        Visco(Elem(J,Q)+NNodes,Elem(J,W))=(ETA)*Vol*YCOMP(W)*XCOMP(Q)+Visco(Elem(J,Q)+NNodes,Elem(J,W))
                        Visco(Elem(J,Q)+2*NNodes,Elem(J,W))=(ETA)*Vol*ZCOMP(W)*XCOMP(Q)+Visco(Elem(J,Q)+2*NNodes,Elem(J,W))
                        Visco(Elem(J,Q)+2*NNodes,Elem(J,W)+NNodes)=(ETA)*Vol*ZCOMP(W)*YCOMP(Q)+Visco(Elem(J,Q)+2*NNodes,Elem(J,W)+NNodes)

                end do
        !Compute Elasicity and Fluctuating Stress Vectors
        Hyper(Elem(J,Q))=Vol*XCOMP(Q)*A(1,1)+Hyper(Elem(J,Q))
        Hyper(Elem(J,Q))=Vol*YCOMP(Q)*A(2,1)+Hyper(Elem(J,Q))
        Hyper(Elem(J,Q))=Vol*ZCOMP(Q)*A(3,1)+Hyper(Elem(J,Q))
        Hyper(Elem(J,Q)+NNodes)=Vol*XCOMP(Q)*A(1,2)+Hyper(Elem(J,Q)+NNodes)
        Hyper(Elem(J,Q)+NNodes)=Vol*YCOMP(Q)*A(2,2)+Hyper(Elem(J,Q)+NNodes)
        Hyper(Elem(J,Q)+NNodes)=Vol*ZCOMP(Q)*A(3,2)+Hyper(Elem(J,Q)+NNodes)
        Hyper(Elem(J,Q)+2*NNodes)=Vol*XCOMP(Q)*A(1,3)+Hyper(Elem(J,Q)+2*NNodes)
        Hyper(Elem(J,Q)+2*NNodes)=Vol*YCOMP(Q)*A(2,3)+Hyper(Elem(J,Q)+2*NNodes)
        Hyper(Elem(J,Q)+2*NNodes)=Vol*ZCOMP(Q)*A(3,3)+Hyper(Elem(J,Q)+2*NNodes)

        Flucstress(Elem(J,Q))=Vol*XCOMP(Q)*Random(1,1)+Flucstress(Elem(J,Q))
        Flucstress(Elem(J,Q))=Vol*YCOMP(Q)*Random(2,1)+Flucstress(Elem(J,Q))
        Flucstress(Elem(J,Q))=Vol*ZCOMP(Q)*Random(3,1)+Flucstress(Elem(J,Q))
        Flucstress(Elem(J,Q)+NNodes)=Vol*XCOMP(Q)*Random(1,2)+Flucstress(Elem(J,Q)+NNodes)
        Flucstress(Elem(J,Q)+NNodes)=Vol*YCOMP(Q)*Random(2,2)+Flucstress(Elem(J,Q)+NNodes)
        Flucstress(Elem(J,Q)+NNodes)=Vol*ZCOMP(Q)*Random(3,2)+Flucstress(Elem(J,Q)+NNodes)
        Flucstress(Elem(J,Q)+2*NNodes)=Vol*XCOMP(Q)*Random(1,3)+Flucstress(Elem(J,Q)+2*NNodes)
        Flucstress(Elem(J,Q)+2*NNodes)=Vol*YCOMP(Q)*Random(2,3)+Flucstress(Elem(J,Q)+2*NNodes)
        Flucstress(Elem(J,Q)+2*NNodes)=Vol*ZCOMP(Q)*Random(3,3)+Flucstress(Elem(J,Q)+2*NNodes)

        !Constants for the Thermal Stress and Bulk Term? ( To Incorporate Flux Stress just add the tensor components)
Stress(Elem(J,Q),1)=(2.0d0*ETA+ETAB)*XCOMP(Q)*Vmid(Elem(J,Q))+A(1,1)+Stress(Elem(J,Q),1)
Stress(Elem(J,Q),2)=(2.0d0*ETA+ETAB)*YCOMP(Q)*Vmid(Elem(J,Q)+NNodes)+A(2,2)+Stress(Elem(J,Q),2)
Stress(Elem(J,Q),3)=(2.0d0*ETA+ETAB)*ZCOMP(Q)*Vmid(Elem(J,Q)+2*NNodes)+A(3,3)+Stress(Elem(J,Q),3)
Stress(Elem(J,Q),4)=ETA*(XCOMP(Q)*Vmid(Elem(J,Q)+NNodes)+YCOMP(Q)*Vmid(Elem(J,Q)))+A(1,2)+Stress(Elem(J,Q),4)
Stress(Elem(J,Q),5)=ETA*(YCOMP(Q)*Vmid(Elem(J,Q)+2*NNodes)+ZCOMP(Q)*Vmid(Elem(J,Q)+2*NNodes))+A(2,3)+Stress(Elem(J,Q),5)
Stress(Elem(J,Q),6)=ETA*(XCOMP(Q)*Vmid(Elem(J,Q)+2*NNodes)+ZCOMP(Q)*Vmid(Elem(J,Q)))+A(1,3)+Stress(Elem(J,Q),6)
        end do
end do
end subroutine

!Redundant
Subroutine Multi(Bin1,Bin2,Bin3)
implicit none
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
Real*8, Dimension(n,n) :: Bin1
Real*8, Dimension(n) :: Bin2
Real*8, Dimension(n) :: Bin3
integer I,J

do I=1,n
Bin3(I)=0.0d0
end do

do I=1,n
        do J=1,n
        Bin3(I) = Bin1(J,I)*Bin2(J)+Bin3(I)
        end do
end do

end subroutine

!Redundant
Subroutine Multi2(Bin1,Bin2,Bin3)
implicit none
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
Real*8, Dimension(n,n) :: Bin1
Real*8, Dimension(n) :: Bin2
Real*8, Dimension(n) :: Bin3
integer I,J

do I=1,n
Bin3(I)=0.0d0
end do

do I=1,n
        do J=I,n
        Bin3(I)=Bin1(I,J)*Bin2(J)+Bin3(I)
        end do
end do

do I=1,n-1
        do J=I+1,n
        Bin3(J)=Bin1(I,J)*Bin2(I)+Bin3(J)
        end do
end do
```

Figure A.12: Fortran 90 stochastic finite element code.

```
end subroutine

Subroutine Multi3(Astor,IP,IRN2,Bin1,Bin2)
!Sparse MAtrix Multiplier USE AFTER A MATRICES CALL
implicit none
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
Integer, parameter :: LAELT=NElem*100 ,NMAX2=n
Real*8, Dimension(n) :: Bin1,Bin2
Real*8, Dimension(LAELT) :: ASTOR
Integer, Dimension(LAELT) :: IRN2
Integer, Dimension(NMAX2+1) :: IP
integer I,J,K,K1,K2

do I=1,n
Bin2(I)=0.0d0
end do

!Read the data from MC57 Routines, consult the MC57 PDF for info on this calculation
DO J = 1,n
K1 = IP(J)
K2 = IP(J+1)-1
        DO K = K1,K2
        Bin2(IRN2(K))=Bin2(IRN2(K))+Astor(K)*BIN1(J)
        if (J .ne. IRN2(K)) then
        Bin2(J)=Bin2(J)+Astor(K)*Bin1(IRN2(K))
        end if
        end do
end do

end subroutine

Subroutine Multi4(Visco,IP,IRN2,Bin1,Bin2)
!Sparse MAtrix Multiplier USE AFTER A MATRICES2 CALL
implicit none
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
Integer, parameter :: LAELT=NElem*100 ,NMAX2=n
Real*8, Dimension(n) :: Bin1,Bin2
Real*8, Dimension(n,n) :: Visco
Integer, Dimension(LAELT) :: IRN2
Integer, Dimension(NMAX2+1) :: IP
integer I,J,K,K1,K2

do I=1,n
Bin2(I)=0.0d0
end do

!Multiplier Read the row and Column information provided by the MC57 Routine and uses that to locate the non-zero elements of the Viscosity Matrix for Matrix Multiplication
DO J = 1,n
K1 = IP(J)
K2 = IP(J+1)-1
        DO K = K1,K2
        Bin2(IRN2(K))=Bin2(IRN2(K))+Visco(IRN2(K),J)*BIN1(J)
        if (J .ne. IRN2(K)) then
        Bin2(J)=Bin2(J)+Visco(IRN2(K),J)*Bin1(IRN2(K))
        end if
        end do
end do

end subroutine

subroutine inverse(Jac,JacIn)
!3x3 Matrix Inverter using the matrices of minor determinants method.
implicit none
real*8, dimension(3,3) :: Jac
real*8, dimension(3,3) :: JacIn
Real*8 Det1,Det2,Det3,Det
integer I,J

Det1= Jac(1,1)*(Jac(2,2)*Jac(3,3)-Jac(2,3)*Jac(3,2))
Det2= Jac(1,2)*(Jac(2,3)*Jac(3,1)-Jac(2,1)*Jac(3,3))
Det3= Jac(1,3)*(Jac(2,1)*Jac(3,2)-Jac(2,2)*Jac(3,1))
Det= Det1+Det2+Det3

do I=1,3
        do J=1,3
                JacIn(J,I)=Jac(mod(I,3)+1,mod(J,3)+1)*Jac(mod(I+1,3)+1,mod(J+1,3)+1)-Jac(mod(I+1,3)+1,mod(J,3)+1)*Jac(mod(I,3)+1,mod(J+1,3)+1)
                JacIn(J,I)=(1.0d0/Det)*JacIN(J,I)
        end do
end do

end subroutine

Subroutine Multiply(D,B,C)
!3x3 Matrix Multiplier
implicit none
real*8, dimension(3,3) :: D
real*8, dimension(3,3) :: B
real*8, dimension(3,3) :: C
integer I,J


do I=1,3
        do J=1,3
```

Figure A.13: Fortran 90 stochastic finite element code.

```
        C(I,J)=D(I,1)*B(1,J)+D(I,2)*B(2,J)+D(I,3)*B(3,J)
        end do
end do
end

Subroutine Trace(A,Tr)
!3x3 Trace Calculator
real*8 Tr
real*8, dimension (3,3) :: A

Tr=0.0d0
Tr=A(1,1)+A(2,2)+A(3,3)

end subroutine

FUNCTION ran1(idum)
USE RandomCons
IMPLICIT NONE
INTEGER(K4B), INTENT(INOUT) :: idum
REAL*8 :: ran1
INTEGER(K4B), PARAMETER :: IA=16807,IM=2147483647,IQ=127773,IR=2836
if (idum <= 0 .or. iy < 0) then
am=nearest(1.0,-1.0)/IM
iy=ior(ieor(888889999,abs(idum)),1)
ix=ieor(777755555,abs(idum))
idum=abs(idum)+1
end if
ix=ieor(ix,ishft(ix,13))
ix=ieor(ix,ishft(ix,-17))
ix=ieor(ix,ishft(ix,5))
KRAN=iy/IQ
iy=IA*(iy-KRAN*IQ)-IR*KRAN
if (iy < 0) iy=iy+IM
ran1=am*ior(iand(IM,ieor(ix,iy)),1)
END FUNCTION ran1

subroutine Read(Elem,Dummy,Nodes)
!Reads Mesh.dat for inital MEsh configuration
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
integer I,J,A,B
Integer, dimension(NElem,4) :: Elem
Integer, dimension(NElem) :: Dummy
real*8, dimension(NNodes,3) :: Nodes

open(10,file='Dyn_link.dat',status='old')
read(10,*) A,B

if ((NNodes .ne. A) .or. (NElem .ne. B)) then
write(6,*) 'Check Number or Nodes and Number of Elements against Array sizes'
stop
end if

Do I=1,NNodes
        read(10,*) Dummy(I),(Nodes(I,J),J=1,3)
end do

Do I=1,NElem
        read(10,*) Dummy(I),(Elem(I,J),J=1,4)
end do
close(10)
end

Subroutine VP(Bin1,Bin2,Bin3)
!Cross Product Subroutine
implicit none
real*8, dimension(3) :: Bin1,Bin2,Bin3
!Standard cross product
Bin3(1)= Bin1(2)*Bin2(3)-Bin1(3)*Bin2(2)
Bin3(2)= Bin1(3)*Bin2(1)-Bin1(1)*Bin2(3)
Bin3(3)= Bin1(1)*Bin2(2)-Bin1(2)*Bin2(1)


end subroutine

subroutine Energy(ASPK,IRN,JCN,Vel,E,EX,EY,EZ)
!Kinetic Energy Calculator Using the Sparsity pattern for the mass MAtrix
implicit none
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
Integer, parameter :: NZMAX=6000
integer, dimension(NZMAX) :: IRN,JCN
real*8, dimension(NZMAX) :: ASPK
real*8, dimension(n) :: Vel
real*8 E,EX,EY,EZ
integer I,J

E=0.0d0
EX=0.0d0
EY=0.0d0
EZ=0.0d0

do I=1,NZMAX
        if ((IRN(I) .ne. 0.0d0) .and. (JCN(I) .ne. 0.0d0)) then
        EX=EX+0.5*Vel(IRN(I))*ASPK(I)*Vel(JCN(I))
        EY=EY+0.5*Vel(IRN(I)+NNodes)*ASPK(I)*Vel(JCN(I)+NNodes)
        EZ=EZ+0.5*Vel(IRN(I)+2*NNodes)*ASPK(I)*Vel(JCN(I)+2*NNodes)
        end if
end do
```

Figure A.14: Fortran 90 stochastic finite element code.

```
E=EX+EY+EZ

end subroutine

subroutine Elastic(F,Vol0,Vol,ACons,GCons,EnPot,J)
!Elastic Energy Calculator: Works on an elemenet by element basis called in the matrices subroutines
implicit none
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
real*8, dimension(3,3) :: F
Real*8 Vol0,Vol,EnPot,EnBulk,EnPres,Guage,Contract
Real*8, dimension(NElem) :: ACons,GCons
Integer I,J,K

Contract=0.0d0
Do I=1,3
        Do K=1,3
        Contract=F(I,K)*F(I,K)+Contract
        end do
end do

EnBulk=0.5d0*GCons(J)*Vol0*Contract
EnPres=0.5d0*ACons(J)*Vol0*((Vol/Vol0)-(1.0d0+GCons(J)/ACons(J)))*((Vol/Vol0)-(1.0d0+GCons(J)/ACons(J)))
Guage=1.5d0*GCons(J)*Vol0+0.5d0*ACons(J)*Vol0*(GCons(J)/ACons(J))*(GCons(J)/ACons(J))
EnPot=EnBulk+EnPres-Guage+EnPot

end subroutine

subroutine Mass(Nodes,Elem,COM,TEMPNODES)
!Centre of Mass Calculator
Implicit none
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
Real*8, Dimension(NNodes,3) :: Nodes,TEMPNODES
Integer, Dimension(NElem,4) :: Elem
Real*8, Dimension(NElem,3) :: ElMass
Real*8, Dimension(3) :: OldA,OldB,OldC,OldD
Real*8, dimension(3) :: COM
Real*8, dimension(NElem) :: MassEL
Real*8 Vol0,Density,MassTot,Vol
Integer I,J,K

!This routine works by calculating the cnetre of mass of each individual element and then averaging them.
!This is achieved using the fact the COM of a tetrahedron is the centroid of the tetrahedron

Density=1.0d0
MassTot=0.0d0
Vol=0.0d0

do I=1,3
COM(I)=0.0d0
end do

do J=1,NElem
        do I=1,3
        ElMass(J,I)=0.0d0
        end do
end do

do J=1,NElem
        do I=1,3
        OldA(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,2),I)
        OldB(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,3),I)
        OldC(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,4),I)
        end do

        call VP(OldA,OldB,OldD)

        Vol0=(1.0d0/6.0d0)*Dot_Product(OldD,OldC)
                if (Vol0 .LT. 0.0d0) then
                call VP(OldB,OldA,OldD)
                Vol0=(1.0d0/6.0d0)*Dot_Product(OldD,OldC)
                end if

        MassEL(J)=Vol0*Density

        do I=1,3
                do K=1,4
                ElMass(J,I)=Nodes(Elem(J,K),I)+ElMass(J,I)
                end do
        end do

        do I=1,3
        ElMass(J,I)=0.25d0*ElMass(J,I)
        end do
end do

do I=1,NElem
MassTot=MassTot+MassEL(I)
end do


do I=1,NElem
        do J=1,3
        COM(J)=MassEL(I)*ElMass(I,J)+COM(J)
        end do
end do

do I=1,3
```

Figure A.15: Fortran 90 stochastic finite element code.

```
COM(I)=COM(I)/(MassTot)
end do


end subroutine

subroutine Angmomentum(Nodes,TEMPNODES,Elem,Vel,COM,ANGMOM)
!Angular Momentum Calculator
implicit none
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
Integer, dimension(NElem,4) :: Elem
Real*8, dimension(NNodes,3) :: Nodes,TEMPNODES
Real*8, Dimension(n) :: Vel
Real*8, Dimension(3,3) :: Old
Real*8, dimension(3) :: OldA,OldB,OldC,OldD,COM,ANGMOM
Real*8, dimension(3) :: CMTrans1,CMTrans2,CMTrans3,CMTrans4,VNode,TEMP
Real*8 Vol0,rho
integer I,J,K,L

do I=1,3
ANGMOM(I)=0.0d0
end do

rho=1.5d3

!Calculates the angular momentum about the centre of mass of the system using finite element analysis solutions.
!See Derivation for explanation (Too long for a comment!)

do J=1,NElem
        do I=1,3
        CMTrans1(I)=Nodes(Elem(J,1),I)-COM(I)
        CMTrans2(I)=Nodes(Elem(J,2),I)-COM(I)
        CMTrans3(I)=Nodes(Elem(J,3),I)-COM(I)
        CMTrans4(I)=Nodes(Elem(J,4),I)-COM(I)
        end do

        do I=1,3
        Old(I,1)=-TEMPNODES(Elem(J,1),I)+TEMPNODES(Elem(J,2),I)
        Old(I,2)=-TEMPNODES(Elem(J,1),I)+TEMPNODES(Elem(J,3),I)
        Old(I,3)=-TEMPNODES(Elem(J,1),I)+TEMPNODES(Elem(J,4),I)
        OldA(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,2),I)
        OldB(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,3),I)
        OldC(I)=TEMPNODES(Elem(J,1),I)-TEMPNODES(Elem(J,4),I)
        end do

        call VP(OldA,OldB,OldD)

        Vol0=(1.0d0/6.0d0)*Dot_Product(OldD,OldC)
                if (Vol0 .LT. 0.0d0) then
                call VP(OldB,OldA,OldD)
                Vol0=(1.0d0/6.0d0)*Dot_Product(OldD,OldC)
                end if

        do K=1,4
                do I=1,3
                VNode(I)=Vel(Elem(J,K)+(I-1)*NNodes)
                end do

                call VP(CMTrans1,VNode,TEMP)

                if (K .eq. 1) then
                        do L=1,3
                        ANGMOM(L)=ANGMOM(L)+(Vol0*rho*TEMP(L))/(10.0d0)
                        end do
                elseif (K .ne. 1) then
                        do L=1,3
                        ANGMOM(L)=ANGMOM(L)+(Vol0*rho*TEMP(L))/(20.0d0)
                        end do
                end if


                call VP(CMTrans2,Vnode,TEMP)
                if (K .eq. 2) then
                        do L=1,3
                        ANGMOM(L)=ANGMOM(L)+(Vol0*rho*TEMP(L))/(10.0d0)
                        end do
                elseif (K .ne. 2) then
                        do L=1,3
                        ANGMOM(L)=ANGMOM(L)+(Vol0*rho*TEMP(L))/(20.0d0)
                        end do
                end if

                call VP(CMTrans3,Vnode,TEMP)
                if (K .eq. 3) then
                        do L=1,3
                        ANGMOM(L)=ANGMOM(L)+(Vol0*rho*TEMP(L))/(10.0d0)
                        end do
                elseif (K .ne. 3) then
                        do L=1,3
                        ANGMOM(L)=ANGMOM(L)+(Vol0*rho*TEMP(L))/(20.0d0)
                        end do
                end if

                call VP(CMTrans4,VNode,TEMP)
                if (K .eq. 4) then
                        do L=1,3
                        ANGMOM(L)=ANGMOM(L)+(Vol0*rho*TEMP(L))/(10.0d0)
                        end do
```

Figure A.16: Fortran 90 stochastic finite element code.

```
                end if

        end do
end do

end subroutine

subroutine determinant(Jac,det)
implicit none
real*8, dimension(3,3) :: Jac
real*8 det
integer I,J,K

det=0.0d0
det=jac(1,1)*(jac(3,3)*jac(2,2)-jac(2,3)*jac(3,2))
det=det-jac(1,2)*(jac(2,1)*jac(3,3)-jac(2,3)*jac(3,1))
det=det+jac(1,3)*(jac(2,1)*jac(3,2)-jac(2,2)*jac(3,1))

end subroutine

subroutine stokesdrag(Vel,SPHR,eta,KB,T,DT,idum,SDV)
USE RandomCons
implicit none
real*8, parameter :: Pi=3.141592653589793
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
real*8, dimension(n) :: Vel,SDV,SPHR
Real*8 ran1,eta,stokes,m,f,KB,T,DT
integer I,J,K,Idum

do I=1,n
stokes=6.0d0*Pi*SPHR(I)*eta
f=sqrt(24.0d0*KB*T*stokes/DT)*(ran1(idum)-0.5d0)
SDV(I)=f-stokes*Vel(I)
end do

end subroutine

subroutine Coordinate(Nodes,Elem,length,angle,ROT)
implicit none
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
real*8, dimension(NNodes,3) :: Nodes,ROT
integer, dimension(NElem,4) :: Elem
real*8, dimension(3) :: Front,Back,Top,Center
real*8, dimension(3) :: V1,V2,V3,U1,U2,U3,N1,N2,N3
real*8 L1,L2,L3,Q,W,E,length,angle
integer I,J,K,A,B

do I=1,3
do I=1,3
Front(I)=0.0d0
Back(I)=0.0d0
Top(I)=0.0d0
Center(I)=0.0d0
end do

do I=1,3
Back(I)=Nodes(168,I)+Nodes(167,I)+Nodes(137,I)+Nodes(157,I)+Nodes(183,I)+Nodes(176,I)
Front(I)=Nodes(134,I)+Nodes(129,I)+Nodes(130,I)+Nodes(139,I)+Nodes(143,I)+Nodes(173,I)+Nodes(164,I)+Nodes(239,I)
Top(I)=Nodes(204,I)+Nodes(205,I)+Nodes(198,I)+Nodes(199,I)+Nodes(200,I)+Nodes(205,I)
end do

do I=1,3
Back(I)=Back(I)/6.0d0
Front(I)=Front(I)/8.0d0
Top(I)=Top(I)/6.0d0
end do


do I=1,3
Center(I)=(Front(I)+Back(I))/2.0d0
end do

do I=1,3
V1(I)=Front(I)-Center(I)
V2(I)=Top(I)-Center(I)
end do

l1=0.0d0
l2=0.0d0

do I=1,3
L1=V1(I)**2.0d0+L1
L2=V2(I)**2.0d0+L2
end do

L1=sqrt(L1)
L2=sqrt(L2)

do I=1,3
V1(I)=V1(I)/L1
V2(I)=V2(I)/L2
end do

do I=1,3
U1(I)=V1(I)
end do
```

Figure A.17: Fortran 90 stochastic finite element code.

```
do I=1,3
U2(I)=V2(I)-(dot_product(V2,U1)/dot_product(U1,U1))*U1(I)
end do

call VP(U1,U2,U3)

L1=0.0d0
L2=0.0d0
L3=0.0d0

do I=1,3
L1=U1(I)**2.0d0+L1
L2=U2(I)**2.0d0+L2
L3=U3(I)**2.0d0+L3
end do

L1=sqrt(L1)
L2=sqrt(L2)
L3=sqrt(L3)

do I=1,3
U1(I)=U1(I)/L1
U2(I)=U2(I)/L2
U3(I)=U3(I)/L3
end do

L1=0.0d0
L2=0.0d0
L3=0.0d0

do I=1,3
L1=U1(I)**2.0d0+L1
L2=U2(I)**2.0d0+L2
L3=U3(I)**2.0d0+L3
end do

call Rotate(Nodes,Elem,U1,U2,U3,center,length,angle,ROT)

end

subroutine rotate(Nodes,Elem,X1,X2,X3,center,length,angle,ROT)
implicit none
integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
real*8, dimension(NNodes,3) :: Nodes,ROT
integer, dimension(NElem,4) :: Elem
real*8, dimension(3) :: X1,X2,X3,V,center,CROT,Head,Tail
real*8, dimension(3,3) :: M,MI
real*8 a,b,c,length,angle
integer I,J,K

do I=1,3
CROT(I)=0.0d0
Head(I)=0.0d0
Tail(I)=0.0d0
end do

do J=1,NNodes
        a=0.0d0
        b=0.0d0
        c=0.0d0

        do I=1,3
        M(I,1)=X1(I)
        M(I,2)=X2(I)
        M(I,3)=X3(I)
        end do

        call inverse(M,MI)

        do I=1,3
        a=MI(1,I)*Nodes(J,I)+a
        b=MI(2,I)*Nodes(J,I)+b
        c=MI(3,I)*Nodes(J,I)+c
        end do

        ROT(J,1)=a
        ROT(J,2)=b
        ROT(J,3)=c
end do

do J=1,3
        do I=1,3
        CRot(J)=MI(J,I)*Center(I)+CRot(J)
        end do
end do

do I=1,NNodes
        do J=1,3
        Rot(I,J)=Rot(I,J)-CRot(J)
        end do
end do

do I=1,3
Head(I)=Rot(301,I)
Tail(I)=Rot(44,I)
end do
```

Figure A.18: Fortran 90 stochastic finite element code.

```
length=0.0d0
do I=1,3
length=(Head(I)-Tail(I))**2.0d0+length
end do
length=sqrt(length)

call Angular(Head,Tail,angle)

end subroutine

subroutine angular(Head,Tail,angle)
implicit none
real*8, parameter :: Pi=3.14159265
Integer, parameter :: NNodes=387,NElem=1127,n=3*NNodes
real*8, dimension(3) :: Head,Tail
real*8, dimension(2) :: A1,A2
real*8 theta,phi,angle,L1,L2
integer I,J,K

L1=0.0d0
L2=0.0d0
theta=0.0d0
phi=0.0d0
angle=0.0d0

A1(1)=Head(1)
A1(2)=Head(2)

A2(1)=Tail(1)
A2(2)=Tail(2)

do I=1,2
L1=A1(I)**2.0d0+L1
L2=A2(I)**2.0d0+L2
end do

L1=sqrt(L1)
L2=sqrt(L2)

do I=1,2
A1(I)=A1(I)/L1
A2(I)=A2(I)/L2
end do

theta=dasin((A1(1)*A2(2)-A2(1)*A1(2))/(A1(1)**2.0d0+A1(2)**2.0d0))
phi=dacos((A1(1)*A2(1)+A1(2)*A2(2))/(A1(1)**2.0d0+A1(2)**2.0d0))

if ((theta .lt. 0.0d0) .and. (phi .le. Pi/2.0d0)) then
theta=2.0d0*Pi+theta
phi=2.0d0*pi-phi
elseif ((theta .lt. 0.0d0) .and. (phi .gt. Pi/2.0d0)) then
theta=Pi-theta
phi=2.0d0*pi-phi
elseif ((theta .ge. 0.0d0) .and. (phi .le. Pi/2.0d0)) then
theta=theta
phi=phi
elseif ((theta .ge. 0.0d0) .and. (phi .gt. Pi/2.0d0)) then
theta=Pi-theta
phi=phi
endif

angle=(360.0d0/(2.0d0*pi))*(Theta+Phi)/2.0d0

end subroutine
```

Figure A.19: Fortran 90 improved coarse grainer code.

```
implicit none
integer, parameter :: NNodes=69649, NSurf=16960
real*8, dimension(NNodes,3) :: Nodes
integer, dimension(NSurf,3) :: Surf
integer, dimension(180,3) :: Elem
integer, dimension(100) :: List
real*8, dimension(3) :: New,NA,MP,Average
integer, dimension(2) :: Hold
real*8  length,l1,l2,l3,length2
integer I,J,K,A,B,RET1,RET2,SSTORE,CHECK,NCOARSE,NCount

open(10,file='sphere_fine.surf',status='old')

read(10,*)
read(10,*) A

if (A .ne. NNodes) then
write(6,*) 'Check NNodes'
stop
end if

do I=1,NNodes
read(10,*) (Nodes(I,J),J=1,3)
end do

read(10,*) B

if (B .ne. Nsurf) then
write(6,*) 'Check NSurf'
stop
end if

do I=1,NSurf
read(10,*) (Surf(I,J),J=1,3)
end do

close(10)

NCOARSE=8000
do I=1,NCOARSE
length=1.0d8
        do J=1,NSURF-(I-1)*2
                l1=0.0d0
                l2=0.0d0
                l3=0.0d0

                do K=1,3
                l1=(Nodes(Surf(J,1),K)-Nodes(Surf(J,2),K))**2.0d0+l1
                l2=(Nodes(Surf(J,1),K)-Nodes(Surf(J,3),K))**2.0d0+l2
                l3=(Nodes(Surf(J,2),K)-Nodes(Surf(J,3),K))**2.0d0+l3
                end do

                l1=sqrt(l1)
                l2=sqrt(l2)
                l3=sqrt(l3)

                if (l1 .lt. length) then
                length=l1
                RET1=Surf(J,1)
                RET2=Surf(J,2)
                SSTORE=J
                end if

                if (l2 .lt. length) then
                length=l2
                RET1=Surf(J,1)
                RET2=Surf(J,3)
                SSTORE=J
                end if

                if (l3 .lt. length) then
                length=l3
                RET1=Surf(J,2)
                RET2=Surf(J,3)
                SSTORE=J
                end if
        end do

        Check=0
        do J=1,NSurf
                if ((Surf(J,1) .eq. RET1) .and. ((Surf(J,2) .eq. RET2) .or. (Surf(J,3) .eq. RET2))) then
                Check=Check+1
                Hold(Check)=J
                end if

                if ((Surf(J,2) .eq. RET1) .and. ((Surf(J,1) .eq. RET2) .or. (Surf(J,3) .eq. RET2))) then
                Check=Check+1
                Hold(Check)=J
                end if

                if ((Surf(J,3) .eq. RET1) .and. ((Surf(J,1) .eq. RET2) .or. (Surf(J,2) .eq. RET2))) then
                Check=Check+1
                Hold(Check)=J
                end if
        end do

        !New code insertion here
        A=RET1
```

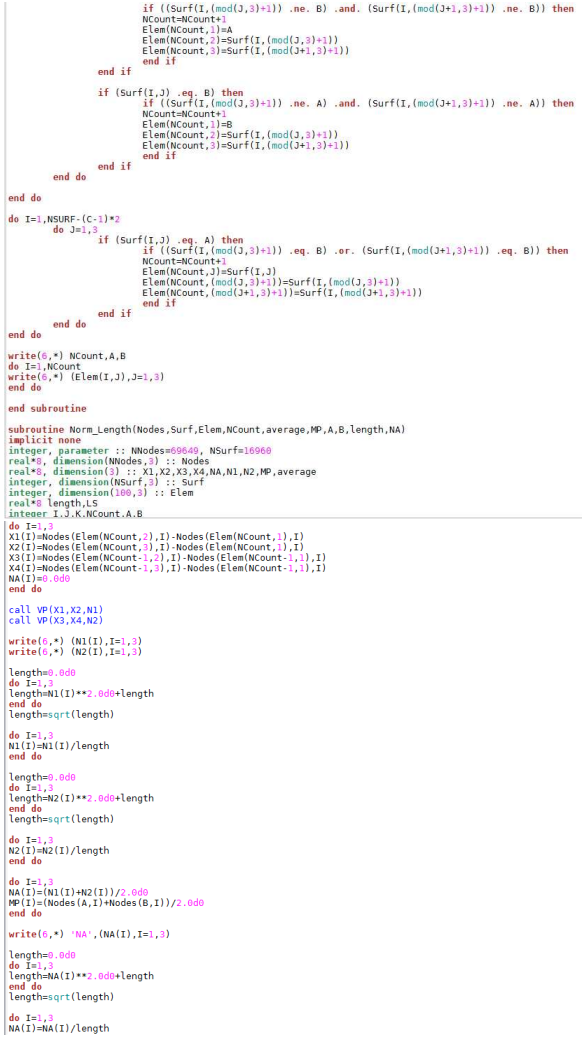
A.5 Coarse Graining Algorithm Fortran Code

The improved coarse graining algorithm Fortan code is provided here.

Figure A.20: Fortran 90 improved coarse grainer code.

```
        B=RET2

        call ELEM_BUILD(Nodes,Surf,Elem,A,B,NCount,I)
        call Norm_Length(Nodes,Surf,Elem,NCount,average,MP,A,B,length2,NA)
        call Converger(Nodes,Elem,NA,MP,Average,NCount)

        do K=1,3
        Surf(Hold(1),K)=0
        Surf(Hold(2),K)=0
        end do

        do K=1,3
        New(K)=MP(K)
        end do

        do K=1,3
        Nodes(RET1,K)=New(K)
        Nodes(RET2,K)=1.0d8
        end do

        !New code insertion here

        do J=RET2,NNodes-I
                do K=1,3
                Nodes(J,K)=Nodes(J+1,K)
                end do
        end do

        do J=1,NSurf-(I-1)*2
                do K=1,3
                        if (Surf(J,K) .eq. RET2) then
                        Surf(J,K)=RET1
                        end if
                end do
        end do

        do J=1,NSurf-(I-1)*2
                do K=1,3
                        if (Surf(J,K) .gt. RET2) then
                        Surf(J,K)=Surf(J,K)-1
                        end if
                end do
        end do

        do J=Hold(2),(NSurf-1)-(I-1)*2
                do K=1,3
                Surf(J,K)=Surf(J+1,K)
                end do
         end do

         do J=Hold(1),(NSurf-2)-(I-1)*2
                 do K=1,3
                 Surf(J,K)=Surf(J+1,K)
                 end do
         end do

write(6,*) I
write(6,*) length
write(6,*) RET1
write(6,*) RET2
write(6,*) SSTORE
write(6,*) Hold(1),Hold(2)

end do


open(20,file='sphere_test.surf',status='unknown')

write(20,*) 'surfacemesh'
write(20,*) NNodes-NCOARSE
do I=1,NNodes-NCOARSE
write(20,*) (Nodes(I,J),J=1,3)
end do
write(20,*) NSurf-2*NCOARSE
do I=1,NSurf-2*NCOARSE
write(20,*) (Surf(I,J),J=1,3)
end do

close(20)

write(6,*) 'finish'
end

subroutine ELEM_BUILD(Nodes,Surf,Elem,A,B,NCount,C)
implicit none
integer, parameter :: NNodes=69649, NSurf=16960
real*8, dimension(NNodes,3) :: Nodes
integer, dimension(NSurf,3) :: Surf
integer, dimension(100,3) :: Elem
integer I,J,K,A,B,C,NCount

NCount=0

do I=1,NSURF-(C-1)*2
        do J=1,3
                if (Surf(I,J) .eq. A) then
```

Figure A.21: Fortran 90 improved coarse grainer code.



```
                        if ((Surf(I,(mod(J,3)+1)) .ne. B) .and. (Surf(I,(mod(J+1,3)+1)) .ne. B)) then
                        NCount=NCount+1
                        Elem(NCount,1)=A
                        Elem(NCount,2)=Surf(I,(mod(J,3)+1))
                        Elem(Ncount,3)=Surf(I,(mod(J+1,3)+1))
                        end if
                end if

                if (Surf(I,J) .eq. B) then
                        if ((Surf(I,(mod(J,3)+1)) .ne. A) .and. (Surf(I,(mod(J+1,3)+1)) .ne. A)) then
                        NCount=NCount+1
                        Elem(NCount,1)=B
                        Elem(NCount,2)=Surf(I,(mod(J,3)+1))
                        Elem(Ncount,3)=Surf(I,(mod(J+1,3)+1))
                        end if
                end if
        end do

end do

do I=1,NSURF-(C-1)*2
        do J=1,3
                if (Surf(I,J) .eq. A) then
                        if ((Surf(I,(mod(J,3)+1)) .eq. B) .or. (Surf(I,(mod(J+1,3)+1)) .eq. B)) then
                        NCount=NCount+1
                        Elem(NCount,J)=Surf(I,J)
                        Elem(NCount,(mod(J,3)+1))=Surf(I,(mod(J,3)+1))
                        Elem(NCount,(mod(J+1,3)+1))=Surf(I,(mod(J+1,3)+1))
                        end if
                end if
        end do
end do

write(6,*) NCount,A,B
do I=1,NCount
write(6,*) (Elem(I,J),J=1,3)
end do

end subroutine

subroutine Norm_Length(Nodes,Surf,Elem,NCount,average,MP,A,B,length,NA)
implicit none
integer, parameter :: NNodes=69649, NSurf=16960
real*8, dimension(NNodes,3) :: Nodes
real*8, dimension(3) :: X1,X2,X3,X4,NA,N1,N2,MP,average
integer, dimension(NSurf,3) :: Surf
integer, dimension(100,3) :: Elem
real*8 length,LS
integer I,J,K,NCount,A,B
do I=1,3
X1(I)=Nodes(Elem(NCount,2),I)-Nodes(Elem(NCount,1),I)
X2(I)=Nodes(Elem(NCount,3),I)-Nodes(Elem(NCount,1),I)
X3(I)=Nodes(Elem(NCount-1,2),I)-Nodes(Elem(NCount-1,1),I)
X4(I)=Nodes(Elem(NCount-1,3),I)-Nodes(Elem(NCount-1,1),I)
NA(I)=0.0d0
end do

call VP(X1,X2,N1)
call VP(X3,X4,N2)

write(6,*) (N1(I),I=1,3)
write(6,*) (N2(I),I=1,3)

length=0.0d0
do I=1,3
length=N1(I)**2.0d0+length
end do
length=sqrt(length)

do I=1,3
N1(I)=N1(I)/length
end do

length=0.0d0
do I=1,3
length=N2(I)**2.0d0+length
end do
length=sqrt(length)

do I=1,3
N2(I)=N2(I)/length
end do

do I=1,3
NA(I)=(N1(I)+N2(I))/2.0d0
MP(I)=(Nodes(A,I)+Nodes(B,I))/2.0d0
end do

write(6,*) 'NA',(NA(I),I=1,3)

length=0.0d0
do I=1,3
length=NA(I)**2.0d0+length
end do
length=sqrt(length)

do I=1,3
NA(I)=NA(I)/length
```

Figure A.22: Fortran 90 improved coarse grainer code.

```
end do

LS=0.0d0

do I=1,NCount
        length=0.0d0
        do J=1,3
        Length=Length+(Nodes(Elem(I,1),J)-Nodes(Elem(I,2),J))**2.0d0
        end do
        Length=sqrt(Length)

        if (length .gt. LS) then
        LS=length
        end if

        length=0.0d0
        do J=1,3
        Length=Length+(Nodes(Elem(I,2),J)-Nodes(Elem(I,3),J))**2.0d0
        end do
        Length=sqrt(Length)

        if (length .gt. LS) then
        LS=length
        end if

        length=0.0d0
        do J=1,3
        Length=Length+(Nodes(Elem(I,1),J)-Nodes(Elem(I,3),J))**2.0d0
        end do
        Length=sqrt(Length)

        if (length .gt. LS) then
        LS=length
        end if
end do

do I=1,3
average(I)=MP(I)-NA(I)*length*5.0d0
end do

end subroutine

subroutine Converger(Nodes,Elem,NA,MP,Average,NCount)
implicit none
integer, parameter :: NNodes=69649, NSurf=16960
real*8, dimension(NNodes,3) :: Nodes
real*8, dimension(3) :: P1,P2,P3,P4,NA,MP
real*8, dimension(3) :: Average
integer, dimension(100,3) :: Elem
real*8 Vol,VolT2,V1,V2,V3,length,test
real*8, dimension(2) :: U,L,Mult
integer I,J,K,NCount

length=0.0d0

do I=1,3
length=(Nodes(Elem(NCount,1),I)-Nodes(Elem(NCount,2),I))**2.0d0+length
end do

length=sqrt(length)

VolT2=0.0d0
V1=0.0d0
V2=0.0d0
V3=0.0d0

do I=1,NCount
        do J=1,3
        P1(J)=Nodes(Elem(I,1),J)
        P2(J)=Nodes(Elem(I,2),J)
        P3(J)=Nodes(Elem(I,3),J)
        P4(J)=Average(J)
        end do
call VTET(P1,P2,P3,P4,Vol)
VolT2=Vol+VolT2
end do

U(1)=1.0d0
L(1)=-1.0d0

do K=1,1

        V2=0.0d0
        do I=1,NCount-2
                do J=1,3
                P1(J)=MP(J)+U(1)*length*NA(J)
                P2(J)=Nodes(Elem(I,2),J)
                P3(J)=Nodes(Elem(I,3),J)
                P4(J)=Average(J)
                end do

                call VTET(P1,P2,P3,P4,Vol)

                V2=Vol+V2
        end do
        V2=V2-VolT2
        U(2)=V2
```

Figure A.23: Fortran 90 improved coarse grainer code.

```
        write(6,*) 'average',(average(I),I=1,3)

        V3=0.0d0
        do I=1,NCount-2
                do J=1,3
                P1(J)=MP(J)+L(1)*length*NA(J)
                P2(J)=Nodes(Elem(I,2),J)
                P3(J)=Nodes(Elem(I,3),J)
                P4(J)=Average(J)
                end do

                call VTET(P1,P2,P3,P4,Vol)

                V3=Vol+V3
        end do
        V3=V3-Vol2
        L(2)=V3

        write(6,*) 'U,L',U(2),L(2)

        Mult(1)=L(1)-L(2)*(U(1)-L(1))/(U(2)-L(2))

        Mult(2)=0.0d0
        do I=1,NCount-2
                do J=1,3
                P1(J)=MP(J)+mult(1)*length*NA(J)
                P2(J)=Nodes(Elem(I,2),J)
                P3(J)=Nodes(Elem(I,3),J)
                P4(J)=Average(J)
                end do

                call VTET(P1,P2,P3,P4,Vol)

                Mult(2)=Vol+Mult(2)
        end do
        Mult(2)=Mult(2)-Vol2

        write(6,*) 'mult',Mult(1),Mult(2)

        if (Mult(2) .le. 0.0d0) then
        L(1)=Mult(1)
        elseif (Mult(2) .gt. 0.0d0) then
        U(1)=Mult(1)
        end if

end do

do I=1,3
MP(I)=MP(I)+mult(1)*length*NA(I)
end do

write(6,*) 'MP',(MP(I),I=1,3)

end subroutine

subroutine VTET(P1,P2,P3,P4,Vol)
implicit none
real*8, dimension(3) :: P1,P2,P3,P4,V1,V2,V3,V4
real*8 Vol
integer I,J,K

do I=1,3
V1(I)=P2(I)-P1(I)
V2(I)=P3(I)-P1(I)
V3(I)=P4(I)-P1(I)
V4(I)=0.0d0
end do

call VP(V1,V2,V4)

Vol=abs((dot_product(V3,V4))/6.0d0)


end subroutine

Subroutine VP(Bin1,Bin2,Bin3)
!Cross Product Subroutine
implicit none
real*8, dimension(3) :: Bin1,Bin2,Bin3
!Standard cross product
Bin3(1)= Bin1(2)*Bin2(3)-Bin1(3)*Bin2(2)
Bin3(2)= Bin1(3)*Bin2(1)-Bin1(1)*Bin2(3)
Bin3(3)= Bin1(1)*Bin2(2)-Bin1(2)*Bin2(1)
end subroutine
```

References

[1] R.C. Oliver et al. A stochastic finite element model for the dynamics of globular macromolecules. *Journal of Computational Physics*, 239(0):147 – 165, 2013.

[2] D.E. Shaw et al. Millisecond-scale molecular dynamics simulations on anton. In *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on*, pages 1–11. IEEE, 2009.

[3] H.J. Kim et al. On coupling a lumped parameter heart model and a three-dimensional finite element aorta model. *Annals of biomedical engineering*, 37(11):2153–2169, 2009.

[4] N. Ban et al. The complete atomic structure of the large ribosomal subunit at 2.4 Å resolution. *Science*, 289(5481):905–920, 2000.

[5] C. Mei et al. Enabling and scaling biomolecular simulations of 100 million atoms on petascale machines with a multicore-optimized message-driven runtime. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 1–11. IEEE, 2011.

[6] D. Abbott et al. *Quantum aspects of life*. Imperial College Press, 2008.

[7] H. Qian. Stochastic physics, complex systems and biology. *Qunatitative Biology*, 1(1):50–53, 2013.

[8] F. Alouges et al. Self-propulsion of slender micro-swimmers by curvature control: N-link swimmers. *International Journal of Non-Linear Mechanics*, 2013.

[9] Per E.M. Siegbahn and H. Fahmi. Recent developments of the quantum chemical cluster approach for modeling enzyme reactions. *Journal of Biological Inorganic Chemistry*, 14(5):643–651, 2009.

[10] L. Monticelli and D.P. Tieleman. Force fields for classical molecular dynamics. In *Biomolecular Simulations*, pages 197–213. Springer, 2013.

[11] G. Michal and D. Schomburg. *Biochemical pathways: an atlas of biochemistry and molecular biology*. Wiley, 2013.

[12] Y. Levy and J.N. Onuchic. Water mediation in protein folding and molecular recognition. *Annu. Rev. Biophys. Biomol. Struct.*, 35:389–415, 2006.

[13] D.E. Shaw et al. Atomic-level characterization of the structural dynamics of proteins. *Science*, 330(6002):341–346, 2010.

[14] H.A. Scheraga et al. Protein-folding dynamics: overview of molecular simulation techniques. *Annu. Rev. Phys. Chem.*, 58:57–83, 2007.

[15] M.J. Frisch et al. Gaussian 09, Revision B.01, 2009.

[16] V. Tozzini. Minimalist models for proteins: a comparative analysis. *Quart. rev. of biophys*, 43(3):333–371, 2010.

[17] W.M. Berhanu et al. Folding and association of a homotetrameric protein complex in an all-atom go model. *Phys. Rev. E*, 87:014701, Jan 2013.

[18] Y. Wang et al. Global ribosome motions revealed with elastic network model. *Journal of Structural Biology*, 147(3):302 – 314, 2004.

[19] G.R. Fleming and G.D. Scholes. Physical chemistry: Quantum mechanics for plants. *Nature*, 431(7006):256–257, 2004.

[20] J. Šponer et al. Nature of base stacking: Reference quantum-chemical stacking energies in ten unique b-dna base-pair steps. *Chemistry A European Journal*, 12(10):2854–2865, 2006.

[21] J. Šponer et al. How to understand quantum chemical computations on dna and rna systems? a practical guide for non-specialists. *Methods*, (0), 2013.

[22] T.A. Soares et al. An improved nucleic acid parameter set for the gromos force field. *Journal of Computational Chemistry*, 26(7):725–737, 2005.

[23] J. Wang et al. Development and testing of a general amber force field. *Journal of Computational Chemistry*, 25(9):1157–1174, 2004.

[24] M. Pekka and N. Lennart. Structure and dynamics of the tip3p, spc, and spc/e water models at 298 k. *The Journal of Physical Chemistry A*, 105(43):9954–9960, 2001.

[25] G. Emilio and R.M. Levy. Agbnp: An analytic implicit solvent model suitable for molecular dynamics simulations and high-resolution modeling. *Journal of Computational Chemistry*, 25(4):479–499, 2004.

[26] V.A. Voelz et al. Molecular simulation of ab initio protein folding for a millisecond folder ntl9(1 39). *Journal of the American Chemical Society*, 132(5):1526–1528, 2010.

[27] A.D. Bates et al. Small dna circles as probes of dna topology. *Biochemical Society Transactions*, 41(part 2), 2013.

[28] M.C. Small et al. Impact of ribosomal modification on the binding of the antibiotic telithromycin using a combined grand canonical monte carlo/molecular dynamics simulation approach. *PLOS Computational Biology*, 9(6):e1003113, 2013.

[29] S. Rouse and M.S.P. Sansom. From the micelle to the membrane: Molecular dynamics simulations of solution nmr structures of membrane proteins. *Biophysical Journal*, 104:408, 2013.

[30] S. Pronk et al. Gromacs 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, 2013.

[31] R. Salomon Ferrer et al. An overview of the amber biomolecular simulation package. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2012.

[32] J.C. Phillips et al. Scalable molecular dynamics with namd. *Journal of computational chemistry*, 26(16):1781–1802, 2005.

[33] S.J. Marrink et al. The martini force field coarse grained model for biomolecular simulations. *The Journal of Physical Chemistry B*, 111(27):7812–7824, 2007.

[34] L. Monticelli et al. The martini coarse grained force field: Extension to proteins. *Journal of Chemical Theory and Computation*, 4(5):819–834, 2008.

[35] J.N. Onuchic and P.G. Wolynes. Theory of protein folding. *Current opinion in structural biology*, 14(1):70–75, 2004.

[36] S.G. Estácio et al. Robustness of atomistic gō models in predicting native-like folding intermediates. *The Journal of Chemical Physics*, 137:085102, 2012.

[37] I. Bahar et al. Global dynamics of proteins: Bridging between structure and function. *Annual Review of Biophysics*, 39(1):23–42, 2010. PMID: 20192781.

[38] F.C. Bernstein et al. The protein data bank: a computer-based archival file for macromolecular structures. *Journal of molecular biology*, 112(3):535–542, 1977.

[39] S.M. Hollup et al. Webnm@: a web application for normal mode analyses of proteins. *BMC bioinformatics*, 6(1):52, 2005.

[40] W. Zheng et al. Toward the mechanism of dynamical couplings and translocation in hepatitis c virus ns3 helicase using elastic network model. *Proteins Structure, Function, and Bioinformatics*, 67(4):886–896, 2007.

[41] E.A. Fadlun et al. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, 161(1):35 – 60, 2000.

[42] S.G. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1):25 – 37, 1992.

[43] H.K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics the finite volume method*. Prentice Hall, 2007.

[44] B.M. Johnston et al. Non-newtonian blood flow in human right coronary arteries: steady state simulations. *Journal of Biomechanics*, 37(5):709 – 720, 2004.

[45] T. Johansson et al. A finite-element model for the mechanical analysis of skeletal muscles. *Journal of theoretical biology*, 206(1):131–149, 2000.

[46] E.M. Purcell. Life at low reynolds number. *Am. J. Phys*, 45(1):3–11, 1977.

[47] M. Doi and S.F. Edwards. *The theory of polymer dynamics*. Oxford Science Publications, 1988.

[48] E.R. Kay et al. Synthetic molecular motors and mechanical machines. *Angewandte Chemie International Edition*, 46(1-2):72–191, 2007.

[49] A. Yonath. Ribosome an ancient cellular nano-machine for genetic code translation. In *Biophysics and the Challenges of Emerging Threats*, pages 121–155. Springer, 2009.

[50] M. Weiss et al. Anomalous subdiffusion is a measure for cytoplasmic crowding in living cells. *Biophysical Journal*, 87(5):3518 – 3524, 2004.

[51] C.M. Pooley et al. Hydrodynamic interaction between two swimmers at low reynolds number. *Physical Review Letters*, 99:228103, Nov 2007.

[52] L.D. Landau and E.M. Lifshitz. *Fluid Mechanics*. Pergamon Press, 1959.

[53] N. Sharma and N.A. Patankar. Direct numerical simulation of the brownian motion of particles by using fluctuating hydrodynamic equations. *Journal of Computational Physics*, 201(2):466–486, 2004.

[54] P.J. Atzberger et al. A stochastic immersed boundary method for fluid-structure dynamics at microscopic length scales. *Journal of Computational Physics*, 224(2):1255 – 1292, 2007.

[55] S. Shah et al. Modeling particle shape-dependent dynamics in nanomedicine. *Journal of nanoscience and nanotechnology*, 11(2):919, 2011.

[56] C.K. Aidun and J.R. Clausen. Lattice boltzmann method for complex flows. *Annual Review of Fluid Mechanics*, 42:439–472, 2010.

[57] R.D. Groot and P.B. Warren. Dissipative particle dynamics bridging the gap between atomistic and mesoscopic simulation. *The Journal of Chemical Physics*, 107(11):4423–4435, 1997.

[58] D.L. Ermak and H. Buckholz. Numerical integration of the langevin equation: Monte carlo simulation. *Journal of Computational Physics*, 35(2):169 – 182, 1980.

[59] I.V. Pivkin and G.E. Karniadakis. Accurate coarse grained modeling of red blood cells. *Physical Review Letters*, 101:118105, 2008.

[60] K. Boryczko et al. Dynamical clustering of red blood cells in capillary vessels. *Journal of Molecular Modeling*, 9(1):16–33, 2003.

[61] T.W. Sirk et al. An enhanced entangled polymer model for dissipative particle dynamics. *The Journal of Chemical Physics*, 136(13):134903, 2012.

[62] E. Mayoral et al. Study of structural properties in complex fluids by addition of surfactants using dpd simulation. In *Fluid Dynamics in Physics, Engineering and Environmental Applications*, Environmental Science and Engineering, pages 233–238. Springer Berlin Heidelberg, 2013.

[63] X. Fan et al. Simulating flow of dna suspension using dissipative particle dynamics. *Physics of Fluids*, 18(6):063102, 2006.

[64] J.C. Chen and A.S. Kim. Brownian dynamics, molecular dynamics, and monte carlo modeling of colloidal systems. *Advances in Colloid and Interface Science*, 112(1 - 3):159–173, 2004.

[65] W. Schaertl and H. Sillescu. Brownian dynamics of polydisperse colloidal hard spheres: Equilibrium structures and random close packings. *Journal of Statistical Physics*, 77(5-6):1007–1025, 1994.

[66] S.R. McGuffee and A.H. Elcock. Diffusion, crowding and protein stability in a dynamic molecular model of the bacterial cytoplasm. *PLoS Comput Biol*, 6(3):e1000694, 03 2010.

[67] W.K. Liu et al. Mathematical foundations of the immersed finite element method. *Computational Mechanics*, 39(3):211–222, 2007.

[68] W.K. Liu et al. Immersed finite element method and its applications to biological systems. *Computer Methods in Applied Mechanics and Engineering*, 195(13 16):1722–1749, 2006.

[69] M. Mason and W. Weaver. The settling of small particles in a fluid. *Phys. Rev.*, 23:412–426, Mar 1924.

[70] R. Kubo. The fluctuation dissipation theorem. *Reports on Progress in Physics*, 29(1):255, 1966.

[71] A. Einstein. Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen. *Annalen der Physik*, 322(8):549–560, 1905.

[72] M. von Smoluchowski. Zur kinetischen theorie der brownschen molekularbewegung und der suspensionen. *Annalen der Physik*, 326(14):756–780, 1906.

[73] W. Sutherland. A dynamical theory of diffusion for non-electrolytes and the molecular mass of albumin. *Philosophical Magazine*, 9:781–785, 1905.

[74] D.S. Lemons and A. Gythiel. Paul langevin's 1908 paper "on the theory of brownian motion"["sur la théorie du mouvement brownien," cr acad. sci.(paris)[bold 146], 530–533 (1908)]. *American Journal of Physics*, 65:1079, 1997.

[75] O.G. Harlen. Finite element method. Course notes on the finite element method.

[76] K.J. Bathe. *Finite Element Method*. Wiley Online Library, 2008.

[77] J.N. Reddy. *An introduction to the finite element method*, volume 2. McGraw Hill New York, 1993.

[78] G. Dhatt et al. *Finite Element Method*. Wiley-ISTE, 2012.

[79] B. Cockburn. Discontinuous galerkin methods. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 83(11):731–754, 2003.

[80] C.M. Andersen. Evaluation of integrals for a ten-node isoparametric tetrahedral finite element. *Computers and Mathematics with Applications*, 5(4):297–320, 1979.

[81] C.L. et al Lawson. Emdatabank.org unified data resource for cryoem. *Nucleic Acids Research*, 39(suppl 1):D456–D464, 2011.

[82] G. Pigino et al. Cryoelectron tomography of radial spokes in cilia and flagella. *The Journal of cell biology*, 195(4):673–687, 2011.

[83] W.N. Findley et al. *Creep and relaxation of nonlinear viscoelastic materials*. North Holland Publishing Company, 1969.

[84] W.M. Lai et al. *Introduction to continuum mechanics*. Butterworth-Heinemann, 2009.

[85] A.F. Bower. *Applied mechanics of solids*. CRC press, 2011.

[86] T. Schlick. *Molecular modeling and simulation: an interdisciplinary guide*, volume 21. Springer, 2010.

[87] C. Hermann. *Statistical Physics including applications to condensed matter*. Springer, 2005.

[88] H.C. Öttinger. *Stochastic processes in polymeric fluids: tools and examples for developing simulation algorithms*. Springer Berlin, 1996.

[89] E. Süli and D.F. Mayers. *An introduction to numerical analysis*. Cambridge University Press, 2003.

[90] L. Verlet. Computer "experiments" on classical fluids. I. thermodynamical properties of lennard-jones molecules. *Physical review*, 159(1):98, 1967.

[91] G.W.E. Hairer. *Solving ordinary differential equations II*. Springer, 2010.

[92] D.V. Schroeder. *An introduction to thermal physics*. Pearson Education India, 2007.

[93] C. Geuzaine and J.-F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[94] J.M. Gere and S.P. Timoshenko. Mechanics of materials, 2004.

[95] R. Narasimha. Non-linear vibration of an elastic string. *Journal of Sound and Vibration*, 8(1):134–146, 1968.

[96] J. Cavanagh et al. *Protein NMR spectroscopy: principles and practice*. Academic Press, 1995.

[97] J. Drenth and J. Mesters. *Principles of protein X-ray crystallography*. Springer, 2007.

[98] J.L. Jimenez et al. Cryo-electron microscopy structure of an sh3 amyloid fibril and model of the molecular packing. *The EMBO journal*, 18(4):815–821, 1999.

[99] J. Lipfert and S. Doniach. Small-angle x-ray scattering from rna, proteins, and protein complexes. *Annu. Rev. Biophys. Biomol. Struct.*, 36:307–327, 2007.

[100] Alice Vrielink and Nicole Sampson. Sub-ångstrom resolution enzyme x-ray structures: is seeing believing? *Current Opinion in Structural Biology*, 13(6):709 – 715, 2003.

[101] Y. Hashem et al. High-resolution cryo-electron microscopy structure of the trypanosoma brucei ribosome. *Nature*, 494(7437):385–389, 2013.

[102] T.D. Grant. Small angle x-ray scattering as a complementary tool for high-throughput structural studies. *Biopolymers*, 95(8):517–530, 2011.

[103] T.D. Grant et al. Coa ligase enyzme saxs envelope. Structure obtained through collaboration with the authors.

[104] K. Palani. Crystal structure of a long-chain-fatty-acid coa ligase (fadd1) from archaeoglobus fulgidus. http://www.rcsb.org/pdb/explore/explore.do?structureId=3G7S, 2009.

[105] Hang S. Tetgen a quality tetrahedral mesh generator and three-dimensional delaunay triangulator. *Weierstrass Institute for Applied Analysis and Stochastic, Berlin, Germany*, 2006.

[106] T. Cellmer et al. Measuring internal friction of an ultrafast-folding protein. *Proceedings of the National Academy of Sciences*, 105(47):18320–18325, 2008.

[107] K.W. Plaxco and D. Baker. Limited internal friction in the rate-limiting step of a two-state protein folding reaction. *Proceedings of the National Academy of Sciences*, 95(23):13591–13596, 1998.

[108] Y. Wang and G. Zocchi. The folded protein as a viscoelastic solid. *EPL (Europhysics Letters)*, 96(1):18003, 2011.

[109] M. Radmacher et al. Imaging adhesion forces and elasticity of lysozyme adsorbed on mica with the atomic force microscope. *Langmuir*, 10(10):3809–3814, 1994.

[110] H. Suda et al. Direct measurement for elasticity of myosin head. *Biochemical and Biophysical Research Communications*, 211(1):219 – 225, 1995.

[111] N.E. Kurland et al. Measurement of nanomechanical properties of biomolecules using atomic force microscopy. *Micron*, 43(2-3):116 – 128, 2012.

[112] M. Tachibana et al. Effect of intracrystalline water on longitudinal sound velocity in tetragonal hen-egg-white lysozyme crystals. *Physical Review E*, 69(5):051921, 2004.

[113] H. Fischer et al. Average protein density is a molecular-weight-dependent function. *Protein Science*, 13(10):2825–2828, 2004.

[114] M.L. Quillin and B.W Matthews. Accurate calculation of the density of proteins. *Acta Crystallographica Section D*, 56(7):791–794, 2000.

[115] A. Ansari et al. The role of solvent viscosity in the dynamics of protein conformational changes. *Science*, 256(5065):1796–1798, 1992.

[116] C. Wu. Engineering fundamentals (efunda). http://www.efunda.com/materials, 2013.

[117] Goodfellow. All the materials you need for scientific and industrial research and manufacturing. http://www.goodfellow.com/E/Polyethylene-Low-Density.html, 2013.

[118] Goodfellow. All the materials you need for scientific and industrial research and manufacturing. http://www.goodfellow.com/E/Polyethylene-High-density.html, 2013.

[119] H. Kojima et al. Direct measurement of stiffness of single actin filaments with and without tropomyosin by in vitro nanomanipulation. *Proceedings of the National Academy of Sciences*, 91(26):12962–12966, 1994.

[120] R. Afrin et al. Pretransition and progressive softening of bovine carbonic anhydrase ii as probed by single molecule atomic force microscopy. *Protein science*, 14(6):1447–1457, 2005.

[121] A. Parra et al. Nanomechanical properties of globular proteins: lactate oxidase. *Langmuir*, 23(5):2747–2754, 2007.

[122] A. Ikai et al. Pulling and pushing protein molecules by afm. *Current Nanoscience*, 3(1):17–29, 2007.

[123] J. Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and visualization in science*, 1(1):41–52, 1997.

[124] M.J. Holmes et al. Temperature dependence of bulk viscosity in water using acoustic spectroscopy. In *Journal of Physics: Conference Series*, volume 269, page 012011. IOP Publishing, 2011.

[125] L. Margetts. Parafem - a general parallel finite element message passing library. http://www.parafem.org.uk/, 2013.

[126] S.J. Di Bartolo. Orientation change of a two-dimensional articulated figure of zero angular momentum. *American Journal of Physics*, 78:733, 2010.

[127] M. Rueda et al. A consensus view of protein dynamics. *Proceedings of the National Academy of Sciences*, 104(3):796–801, 2007.

[128] A.B. Guliaev et al. Structural insights by molecular dynamics simulations into differential repair efficiency for ethano-a versus etheno-a adducts by the human alkylpurine-dna n-glycosylase. *Nucleic acids research*, 30(17):3778–3787, 2002.

[129] C.D. Snow et al. The trp cage: folding kinetics and unfolded state topology via molecular dynamics simulations. *Journal of the American Chemical Society*, 124(49):14548–14549, 2002.

[130] T.W. Allen et al. On the importance of atomic fluctuations, protein flexibility, and solvent in ion permeation. *The Journal of general physiology*, 124(6):679–690, 2004.

[131] J.S. Higgins and H. Benoît. *Polymers and neutron scattering*. Clarendon Press Oxford, 1994.

[132] Haydyn D.T. Mertens and Dmitri I. Svergun. Structural characterization of proteins and complexes using small-angle x-ray solution scattering. *Journal of Structural Biology*, 172(1):128 – 141, 2010.

[133] D.J. Read. Scattering theory i. Course notes on elastic scattering.

[134] D.J. Read. Scattering theory ii. Course notes on form and structure factors.

[135] P. Bernadó et al. Structural characterization of flexible proteins using small-angle x-ray scattering. *Journal of the American Chemical Society*, 129(17):5656–5664, 2007.

[136] D.I. Svergun. Restoring low resolution structure of biological macromolecules from solution scattering using simulated annealing. *Biophysical Journal*, 76(6):2879 – 2886, 1999.

[137] M.V. Petoukhov et al. Atsas 2.1-towards automated and web-supported small-angle scattering data analysis. *Applied Crystallography*, 2007.

[138] T.D. Grant et al. Tom1l1 scattering envelope. Obtained on collaborative visit to Stanford and the Hauptman Woodward Institute.

[139] D. Franke and D.I. Svergun. Dammif, a program for rapid ab-initio shape determination in small-angle scattering. *Journal of Applied Crystallography*, 42(2):342–346, 2009.

[140] V.V. Volkov and D.I. Svergun. Uniqueness of ab initio shape determination in small-angle scattering. *Journal of Applied Crystallography*, 36(3):860–864, 2003.

[141] P.V. Konarev et al. Primus: a windows pc-based system for small-angle scattering data analysis. *Journal of Applied Crystallography*, 36(5):1277–1282, 2003.

[142] A. Guinier and F. Foumet. *Polymers and neutron scattering*. Wiley Interscience: New York, 1955.

[143] T. Kon et al. X-ray structure of a functional full-length dynein motor domain. *Nature structural & molecular biology*, 18(6):638–642, 2011.

[144] A.J. Roberts et al. Atp-driven remodeling of the linker domain in the dynein motor. *Structure*, 2012.

[145] R. Mallik and S.P. Gross. Molecular motors: strategies to get along. *Current Biology*, 14(22):R971–R982, 2004.

[146] Ingmar H.R-K. et al. How molecular motors shape the flagellar beat. *HFSP journal*, 1(3):192–208, 2007.

[147] I. Rayment et al. Structure of the actin-myosin complex and its implications for muscle contraction. *Science*, 261(5117):58–65, 1993.

[148] S.A. Burgess et al. Dynein structure and power stroke. *Nature*, 421(6924):715–718, 2003.

[149] K.W-H. Lo et al. Identification of a novel region of the cytoplasmic dynein intermediate chain important for dimerization in the absence of the light chains. *Journal of Biological Chemistry*, 281(14):9552–9559, 2006.

[150] K.A. Johnson. Pathway of the microtubule-dynein atpase and the structure of dynein: a comparison with actomyosin. *Annual review of biophysics and biophysical chemistry*, 14(1):161–188, 1985.

[151] J. Frank et al. Spider and web: processing and visualization of images in 3d electron microscopy and related fields. *Journal of structural biology*, 116(1):190–199, 1996.

[152] E.F. Pettersen et al. Ucsf chimera-a visualization system for exploratory research and analysis. *Journal of computational chemistry*, 25(13):1605–1612, 2004.

[153] E. Volterra et al. Dynamics of vibrations. *Journal of Applied Mechanics*, 33:956, 1966.

[154] L.P. Kadanoff. *Statistical physics: statics, dynamics and renormalization*. World Scientific Publishing Company, 2000.