A mesoscale model for coarse-grained protein dynamics

Robin Archibald Richardson

School of Physics and Astronomy

University of Leeds

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

September 2014

Declaration

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly authored publications has 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.

The work presented in Chapter 4 of this thesis is based on that published in: R. A. Richardson, K. Papachristos, D. J. Read, O. G. Harlen, M. Harrison, E. Paci, S. P. Muench, and S. A. Harris, "Understanding the apparent stator-rotor connections in the rotary atpase family using coarse-grained computer modelling," *Proteins: Structure, Function, and Bioinformatics*, 2014. FFEA simulations, analysis and figures (except figure 4.1) were produced by R. A. Richardson. The ENM simulations were run by K. Papachristos. Figure 4.1 was created by S. P. Muench. Contributions to discussions, ideas and manuscript preparation from all authors.

To my parents, Robert and Maryse, and my sister, Sophie

Acknowledgements

I am indebted first and foremost to Sarah Harris, Daniel Read and Oliver Harlen, under whose excellent supervision I carried out the work presented in this thesis. Their combined expertise in molecular modelling, polymer dynamics and Finite Element Analysis provided the ideal breadth of guidance. I am also grateful for the help and patience shown to me by Robin Oliver, creator of FFEA, and always a good, if somewhat evil, friend.

I would particularly like to acknowledge our eternally helpful collaborators in biology, Stan Burgess and Stephen Muench, always supportive and seemingly blessed with an impressive ability to tolerate the sorts of questions physicists ask about biology. Without them I would be lost. I also appreciate the helpful discussions I had with Peter Olmsted and Evy Kersale. Special thanks goes to Glenys Bowles who has been of enormous help throughout the years with admin and cake related issues.

I thank my friends, firstly those in the 'rival' Condensed Matter Group, a group that taught me that small doesn't mean gentle (Sophie, Susan and May) and which notoriously contains the sort of lunatic who organises his wedding for the last month of his PhD (yes, Rowan). I acknowledge also my friends in the polymer group (Winke, Steve, James, Outi), and particularly my office mate Dan Baker whose no nonsense approach to life has been an inspiration, and without whom I would have had no one to annoy but myself.

I must also thank my long suffering family, who have always supported me, and my oldest travelling companion, Anshul Sirur.

Finally, my apologies to Ben Hanson, who I hope can one day forgive me for the problems I have left behind.

Abstract

Proteins are the essential units of biological processes, but modelling their dynamics is a very computationally expensive task. A wide variety of simulation techniques exist, a popular example being Molecular Dynamics. However, such models typically involve detailed simulation of the protein's structure at or near the atomic level and as such are unsuitable for modelling biological systems composed of large or multiply interacting proteins.

This research takes a coarse-graining approach, called Fluctuating Finite Element Analysis, in which large, globular proteins are approximated by viscoelastic continua subject to thermal noise. Each protein is discretised into a tetrahedral mesh, parameterised locally by its bulk continuum properties. The forces are then calculated using Finite Element Analysis.

A parallel implementation of the FFEA algorithm has been developed for use on high performance computing facilities. The scalability of the algorithm with respect to number of cores and system size, and its stability with respect to integration time step has been investigated. A pipeline for fully automated FFEA system creation from atomistic (X-ray crystallography and NMR) or low resolution data (cryo-EM and SAXS) has also been developed.

In order to tackle multiprotein systems, the FFEA model has been extended to include van der Waals interactions and electrostatics. FFEA has been applied to a number of diverse biological systems. The van der Waals scheme was tested through simulation of myoglobins interacting with a polystyrene substrate. The major modes of motion of V- and A- type rotary ATPases were extracted using Principal Component Analysis, and compared with the normal modes obtained from the Elastic Network Model. Finally, the effect of axonemal dynein c's interaction with the microtubule track on its step length and exploration of binding sites was investigated. A mapping was developed to allow in-simulation conformational switching of the dynein motor.

Abbreviations

API	Application Programming Interface
ARC1	Advanced Research Computing Node 1
ATP	Adenosine triphosphate
ADP	Adenosine diphosphate
BC	Boundary Condition
BEM	Boundary Element Method
Bi-CGSTAB	Bi-Conjugate Gradient Stabilised
BTS	Bend-Twist-Stretch
CCP4	Collaborative Computational Project Number 4
Cryo-EM	Cryo Electron Microscopy
CG	Coarse-grained
Δt	Integrator time step
EMDB	Electron Microscopy Data Bank
ENM	Elastic Network Model
EOM	Equation of Motion
ϵ_{LJ}	Lennard-Jones well depth
η_B	Bulk viscosity
η_S	Shear viscosity
$\vec{F}$	Force
FAC	Fast Adaptive Composite
FD	Finite Difference
FE	Finite Element
FEM	Finite Element Method
FEA	Finite Element Analysis
FFEA	Fluctuating Finite Element Analysis
FMM	Fast Multipole Method
FPC	Face pair collapse
Γ	Surface
HPC	High Performance Computing
$\underline{J}^{\Gamma}$	Electrostatic surface flux

κ	Inverse Debye screening length
k_B	Boltzmann's constant
KE	Kinetic Energy
k_BT	Thermal energy
LHS	Left hand side
LJ	Lennard-Jones
LPBE	Linearised Poisson Boltzmann Equation
M	Mass matrix
MD	Molecular Dynamics
MLAT	Multilevel Adaptive Technique
MPI	Message Passing Interface
MRC	Medical Research Council
MT	Microtubule
Ω	Interior
$\bar{\Omega}$	Exterior
OpenMP	Open Multi-Processing
PBC	Periodic Boundary Conditions
PBE	Poisson-Boltzmann Equation
PCA	Principal Component Analysis
PCGS	Preconditioned Conjugate Gradient solver
PDB	Protein Data Bank
PDE	Partial Differential Equation
PE	Potential Energy
PEQ	Poisson's Equation
ϕ	Electrostatic potential
ψ	Shape function
r_{eq}	Lennard-Jones equilibrium separation
ρ	Mass density
ρ^Q	Charge density
RHS	Right hand side
RNG	Random Number Generator
SAXS	Small Angle X-ray Scattering
σ	Stress
SIMD	Single instruction, multiple data
$\vec{u}$	Node velocity
V	Volume
vdW	van der Waals
Y	Young's modulus

Contents

List of Figures

LIST OF FIGURES

Chapter 1

Introduction

Life is an intrinsically multiscale process. From electron transfer in photosynthesis[1] to the beating of a human heart there are eight orders of magnitude in length (10^{-9}m to 10^{-1}m) and twelve orders of magnitude in time (ps to seconds). A full spectrum understanding of the physics of life is therefore reliant on connecting and integrating the various physical descriptions appropriate to each scale. While experimental techniques provide an invaluable source of knowledge on the workings of biological systems, the fragility and complexity of living organisms can make it challenging or impossible to observe many processes *in vivo*. In such cases, computer simulations may provide useful insights into the system of interest and act as a comparatively inexpensive means of testing hypotheses and the effects of experimental artifacts. Crucially, simulations also provide a means of visualising these complex processes and drawing together often diverse experimental data into a coherent physical model. Atomistic computer simulations, for example, have been described as providing a "computational microscope for biomolecular systems"[2].

It is unsurprising, therefore, that there exist a vast number of different simulation methods, each suitable to processes occurring on particular length and time scales (see section 1.1 below). The computational cost of simulations typically increases as a function of length and time, often with a prohibitive power law of system size, limiting the applicability of accurate, high resolution modelling techniques, and necessitating substantial approximations to access larger and more complex systems. This process of coarse-graining lies behind the transition from the quantum physical treatment of photosynthesis to the classical continuum treatment of a heart.

1.1 Current simulation methods

For the purposes of this project, we shall crudely divide the length-time-scale spectrum into three scale regimes: the nanoscale, the mesoscale and the macroscale. Figure 1.1 illustrates this spectrum of length and time scales with the relevant biological processes. The nanoscale concerns processes with length scales of Angstroms or nanometres, or time scales on the order of nanoseconds. Molecular recognition of genes[4], protein folding[7], enzyme catalysis[3] and photosynthesis[1] are all processes that can be said to occur on the nanoscale. On the other hand, the macroscale scale is concerned with processes on lengths exceeding a hundred μm, or times greater than a millisecond. The biology of organs and tissue occurs at this scale. The physics appropriate to the macroscale averages over the atomistic interactions of the nanoscale, treating matter as a continuum defined by its bulk properties. Furthermore, thermal fluctuations are negligible (due to the large length and time scale). The mesoscale (indicated in orange in the figure) refers to

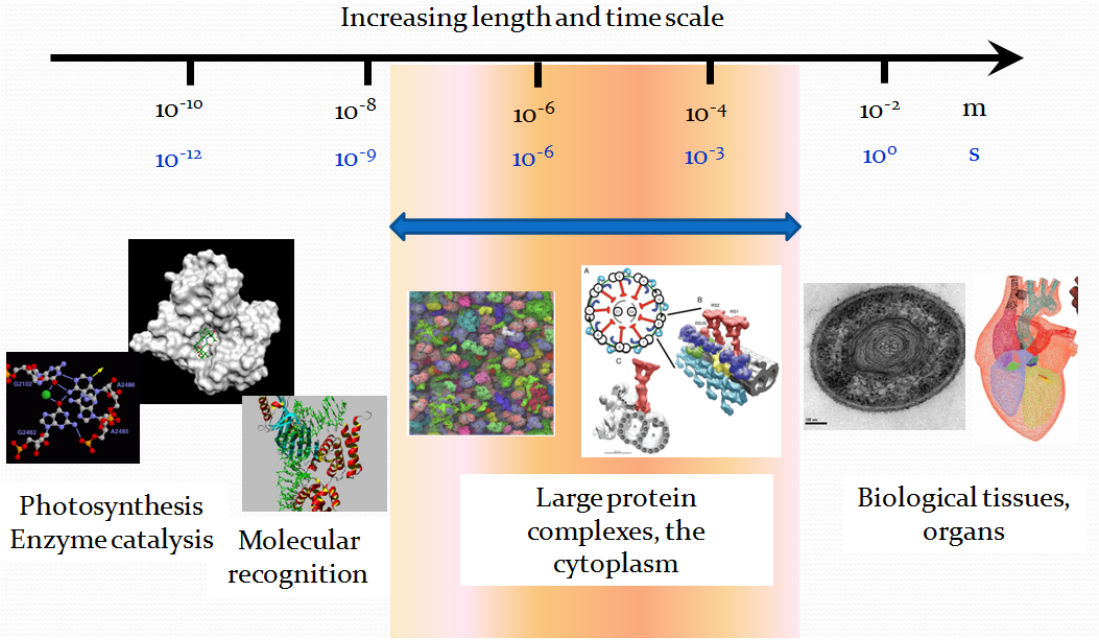


Figure 1.1: Illustration of the length-time scale of existing methods of biological simulation. Photosynthesis[1], enzyme catalysis[3] and molecular recognition[4] are processes said to occur at the nanoscale (lengths of around 10 nm or less, and times of around 10 ns or less). The physics of biological tissues and organs[5]. belongs to the macroscale (lengths greater than around 100 μm, and times of ms or greater). Between these two extremes we have the mesoscale (shown in orange), in which interesting biophysical systems exist, such as cytoplasmic crowding (left figure, reference [6]) or the molecular motors found in flagella (right figure, see chapter 5).

the regime lying between the two extremes of the nanoscale and the macroscale, in which the quantum physical (or even atomistic) detail is unimportant but the systems are small enough such that thermal fluctuations are significant. The available simulation techniques applicable to each of these three scales are now discussed.

1.1.1 Nanoscale

At the nanoscale the individual atoms, and frequently even the electron clouds, are important. Processes that depend on electrons (such as enzyme catalysis and photosynthesis) must be treated using quantum physical models, whereas processes such as molecular recognition and protein folding may be treated using classical methods.

At the lower end of the length-time scale we have *ab initio* quantum chemistry approaches such as density functional theory (DFT). DFT is a method for solving the Schrödinger equation in many-body systems, allowing study of the electronic structure of materials[8]. Popular DFT codes include Gaussian[9] and CASTEP[10].

However, these quantum physics based simulations are generally too computationally expensive to treat molecules of much more than 100 atoms[11]. For this reason, it is necessary to resort to classical approximations to simulate more complex systems. Molecular Dynamics (MD) calculates the trajectory of molecules using classical potentials and Newtonian physics. The force-fields within these models can be parameterised empirically, or using DFT[12]. Widely used MD packages include CHARMM[13], Amber[14], Gromacs[15] and NAMD[16]. It is

possible to combine quantum mechanical methods such as DFT with MD, by using DFT to take into account the electronic structure during the breaking or formation of covalent bonds, a technique known as Ab Initio MD[17]. Alternatively, the simulation may treat only a specific region of interest using quantum mechanical (QM) methods, embedding it in a larger region treated with molecular mechanics (MM), a technique known as hybrid QM/MM [18]. QM/MM can be used to gain insight into biological processes such as enzyme reactions[19] but is still limited to systems of tens or hundreds of atoms[20].

Even with fully classical MD simulations running on specialised resources it is typically only possible to simulate dynamics over a millisecond time scale for objects of around 1 nm[7]. Substantial improvements to techniques and hardware in recent years allow probing of microsecond time scales for systems comprising tens of thousands of atoms[21]. This is clearly not sufficient for handling the much larger proteins and protein complexes (such as the hundreds of thousands of atoms making up the ribosome[22]) that are essential to life processes.

1.1.2 Macroscale

At the macroscale, the molecular detail is replaced by a continuum approximation, in which the locally spatially averaged dynamics are described by differentiable fields satisfying partial differential equations (PDEs). There are a number of ways of solving the PDEs which arise from this continuum treatment. Perhaps the simplest is the *Finite Difference Method* which spatially divides the system into a regular grid, reducing the PDE to a simple set of discretised equations acting on this grid. However, this method requires the material boundaries to

be mapped onto the grid, making it computationally expensive for geometrically complex systems[23]. The *Finite Element Method*, which discretises the domain into a mesh of elements (simple shapes, such as tetrahedra) with the solution defined at the nodes, allows solution on unstructured meshes and so is better suited to irregular geometries. This method is described in detail in section 1.2.1 below. A related technique is *Finite Volume*, in which we define a volume around each grid point, and convert the continuum equation into surface integrals across the boundaries between the volumes[24]. This technique is good for modelling fluid flow and is used in the study of processes such as blood flow[5; 25].

1.1.3 Mesoscale

This scale is loosely defined as comprising systems between 10 nm and less than a μm, with time scales from ns to hundreds of μs. Interesting biology occurs on this scale, for example molecular motors in flagella [26] and cytoplasmic crowding[6]. Towards the upper end of the mesoscale we find individual cells and micro swimmers like sperm (lengths of μm), with little effect from thermal fluctuations. In this thesis we are primarily concerned with the lower end of the mesoscale, in which thermal noise is not only significant, but is also one of the main drivers of dynamics in the system.

To tackle these larger scale problems, a number of coarse-grained (CG) models based on Molecular Dynamics have been developed. The advantages and disadvantages of each model are determined mainly by the class of problems each is designed to address. In processes such as protein folding, for example, it is necessary to coarse grain the protein representation no further than to the scale

at which it can still reliably and accurately describe the secondary and tertiary structure[1] of the protein. Commonly used representations for this are the "one-bead" models, in which each amino acid building block is reduced to a single "bead" with an interaction potential encapsulating the behavioural parameters of that part of the chain[27]. A number of levels of coarse-graining are possible, such as increasing the number of beads to include explicitly the side-chains of the amino acids, or reducing the number of beads so that each bead represents several amino acids[27]. The most common force-field in use for this type of coarse-grained MD simulation is the MARTINI force-field [28].

An alternative means of reducing simulation time is to bias the potential such that the energy landscape resembles a funnel towards the protein's native state[29], by defining favourable interactions for native contacts. This type of technique is referred to as a Gō-model and is applied to protein folding problems.

For even larger proteins, a more extreme coarse-graining technique may be used, in the form of Elastic Network Models (ENMs). ENMs are a widely used type of structure-based coarse-grained model that represent the protein's native structure by a set of beads connected by springs[30]. The coarse-graining step consists of mapping the nodes onto protein structural elements (typically one node per alpha carbon atom). Cut-off distances are usually employed to set interacting nodes. ENMs have been used to investigate biomolecular dynamics of both high-resolution X-ray structures and low-resolution cryo-electron microscopy data[31; 32].

The coarse-grained models described above are "particle-based", meaning that

[1]The primary structure is the linear sequence of amino acids the protein is built from. Secondary and tertiary structure describe how these blocks position themselves in the protein's local and global structure.

they represent molecules as collections of particles or other structural units (e.g. α-helices) that represent groups of atoms and model the forces between them resulting from hydrodynamics, electrostatics and thermal fluctuations. One key differentiation between these different algorithms is in their treatment of the solvent environment. In Brownian Dynamics (BD) the solvent is treated implicitly through a hydrodynamic force on each particle derived on the assumption of Stokes flow in the solvent. The simplest treatment is to use Stokes drag on an isolated particle. This widely adopted approximation is simply implemented and minimises the computational cost, but neglects hydrodynamic interactions between particles (other than through a mean-field approximation) and so does not capture hydrodynamic coupling. The lack of explicit solvent and long range hydrodynamics allows far longer time scales to be simulated, and is particularly apt for crowded solutions for which experimentally observable trends in diffusion can be reproduced[6]. In 2010, McGuffee and Elcock used BD in order to simulate diffusion of 1008 macromolecules (made up from 50 common types) within the crowded bacterial cytoplasm for up to 20 μs[33] (to probe such time scales, however, the molecules were modelled as completely rigid). The inclusion of hydrodynamics interactions results in a formulation of the fluctuation-dissipation theorem that requires the factorisation of a full matrix of the size of the number of particles at each time-step, greatly increasing the computational costs as the system increases. However, recently schemes that truncate long-range interactions have been developed to provide $\mathcal{O}(N)$ dependence on system size [34; 35].

The alternative is to include the solvent explicitly either as particles or as a continuum. Dissipative Particle Dynamics (DPD) [36] uses dissipative interactions between soft-spheres to reproduce fluid motion with significantly fewer

solvent particles and longer time steps than that required by classical MD. It has been used to model a wide range of problems, including drug delivery vehicles and their transport across lipid membranes[37] or in modelling suspension flows and the assembly of objects such as micelles[36]. An additional coarse-grained particulate solvent technique is Multiparticle Collision Dynamics (MPC), in which the solvent behaviour is modelled through alternating 'streaming' and 'collision' steps[38]. Particle interaction is modelled in the collision step by grouping particles into a number of 'collision cells', for which the velocities of the member particles are rotated by some angle around a random rotation axis, such that momentum and energy are conserved in each cell.

All of the above methods in essence approach the mesoscale via the coarse-graining of nanoscale simulations. The alternative is to incorporate mesoscale physics into macroscale simulation techniques. Lattice Boltzmann (LB) methods use a discrete version of the Boltzmann equation to determine fluid flow from the particle velocity distribution function on a lattice. Although originally devised for macroscopic flows, LB methods can be applied to mesoscale systems by incorporating thermal fluctuations within the LB framework [39]. These methods require only local collisions and so are linear in system size and straightforward to parallelise. However, since they require calculation of the velocity distribution to determine the fluid motion they are typically several orders of magnitude slower than methods such as Finite Element Analysis (FEA) and finite volume methods that solve the continuum equations directly. These continuum mechanics methods have also recently been used to model systems over smaller length scales by including thermal fluctuations, for example to model the Brownian motion of nanoparticles in a Newtonian fluid due to thermally induced hydrodynamic fluc-

tuations [40], and through a stochastic version of the immersed boundary method [41].

1.2 FFEA

1.2.1 Finite Element Analysis

As discussed above, the Finite Element Method (FEM) provides a flexible approach for solving partial differential equations (PDEs) in problems with complex geometries. This method involves subdividing the domain of integration into a set of *elements* (generally simple shapes such as triangles in 2-d, or tetrahedra in 3-d) and then finding an approximate solution to the PDE from a piecewise polynomial solution constructed for each of these elements. Rather than solve the equation exactly, a so-called *weak form* of the equation is introduced[42].

The basis of the Finite Element Method is as follows. Consider the solution of the PDE:

$$\mathcal{D}u = 0, \tag{1.1}$$

where u satisfies the appropriate boundary conditions (BCs). For any function u, we can define the residual:

$$R(u) = \mathcal{D}u. \tag{1.2}$$

Now we seek the best approximation for u of the form:

$$u = \sum_{i=1}^{N} c_i \psi_i \tag{1.3}$$

where the ψ_i are a prescribed set of basis functions and the c_i are unknown coefficients. In general there will be no solution for the c_i that exactly solves the PDE $\mathcal{D}u = 0$, for which R is identically zero. Instead we seek to find solutions for c_i for which some weighted integral of R is zero:

$$\int_\Omega w_i R dV = 0 \tag{1.4}$$

where w_i are weighting functions and Ω is the domain of integration. This is referred to as the *weak formulation*. If $\mathcal{D}$ is a linear differential operator then a set of weighting functions equal in number to the set of basis functions will give a unique solution for the c_i. A simple choice is therefore to take the ψ_i as the weighting functions in which case we have:

$$\int_\Omega \psi_i R dV = 0 \tag{1.5}$$

which is referred to as the *Galerkin formulation* of the problem. It then remains to choose appropriate basis functions (known as *shape functions*). It is usual to choose c_i to be the value of the function at a particular node x_i and so the ψ_i are chosen such that:

$$\psi_i(x) = \begin{cases} 1 & x = x_i \\ 0 & x = x_j, i \neq j \end{cases} \tag{1.6}$$

For example, for tetrahedral elements we can choose the nodes to be the vertices of the tetrahedron. The shape functions can then be chosen to be linear functions that take the value of unity or zero at the vertices, yielding a piecewise linear approximate solution to the PDE within that element.

1.2.2 Fluctuating Finite Element Analysis

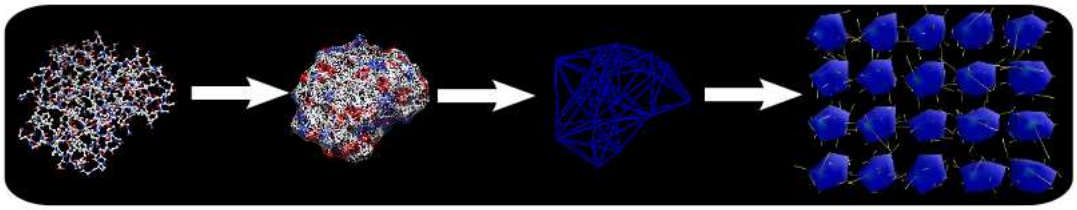


Figure 1.2: An illustration of the transformation of a protein (myoglobin) to its continuum representation. An FFEA simulation of myoglobin proteins is presented in chapter 3.

Fluctuating Finite Element Analysis (FFEA) treats molecules as continuum viscoelastic solids subject to thermal noise chosen so as to satisfy the fluctuation-dissipation theorem, and was introduced as a method for simulation of biological macromolecules by Oliver *et al.* in 2013 [43; 44]. The equation of motion for a viscoelastic continuum subject to a thermal stress is given by equation (1.7):

$$\rho\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right) = \frac{\partial \sigma_{ij}}{\partial x_j} \tag{1.7}$$

where x_i and u_i denote the position and velocity, respectively, of a point in the material. i and j are indices denoting the three spatial dimensions. $\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right)$ is the total time derivative of the velocity vector field in the material's Lagrangian frame. The stress σ_{ij} has three contributions:

$$\sigma_{ij} = \sigma^v_{ij} + \sigma^e_{ij} + \sigma^t_{ij} \tag{1.8}$$

where σ^v, σ^e and σ^t, are the viscous, elastic and thermal stress contributions respectively. Conceptually, the viscosity term is the combined effect of non-

conservative, short range molecular friction within the protein while the elastic term represents the effects of conservative forces due to recoverable changes in local molecular configuration. In the present model, the viscous and elastic stresses are derived from the current state of deformation and deformation rate using a Kelvin-Voigt model[1]. Experiments have suggested that a folded protein behaves as a viscoelastic solid with a relaxation time of order 10^{-2}s[45]. This relaxation is negligible when compared with the time scales intended for treatment with FFEA (μs), indicating that a Kelvin-Voigt model of viscoelasticity is indeed appropriate.

In FFEA the material is given an isotropic linear viscous stress σ^v_{ij}:

$$\sigma^v_{ij} = \eta_S \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + (\eta_B - \tfrac{2}{3}\eta_S) \frac{\partial u_m}{\partial x_m} \delta_{ij} \tag{1.9}$$

where $\vec{u}$ is the velocity field, and η_S and η_B are the shear and bulk viscosities respectively. The elastic stress is determined from the local deformation gradient from equilibrium, $F_{ij} = \frac{\partial x_i}{\partial X_j}$ (where $\vec{x}$ represents the current location of the material and $\vec{X}$ the initial, equilibrium location), as follows:

$$\sigma^e_{ij} = \frac{G}{det(F_{ij})} F_{ik} F^T_{kj} + B(det(F_{ij}) - \alpha)\delta_{ij} \tag{1.10}$$

where G is the shear modulus, $K = B - \frac{G}{3}$ is the bulk modulus and $\alpha = 1 + \frac{G}{B}$.

Oliver *et al.* demonstrated that equations (1.7) and (1.8) could be discretised using the finite element method, yielding equation (1.11), the equation of motion

[1]The material properties of a Kelvin-Voigt solid can be expressed as an elastic spring and a viscous damper connected in parallel.

for a single protein[43]:

$$M_{pq}\frac{\partial u_q}{\partial t} = -K_{pq}u_q + E_p + N_p \tag{1.11}$$

where M is the mass matrix, $\vec{u}$ is the velocity, t is time, K is the viscosity matrix, E is the elasticity vector and $\vec{N}$ is the thermal stress vector. The thermal stress arises from the fluctuation dissipation. In the Kelvin-Voigt model this is dependent purely on the viscosities. The matrices in equation (1.11) are constructed as follows:

$$M_{pq} = \delta_{ij}\left(\int_V \rho\psi_\alpha\psi_\beta dV\right) \tag{1.12}$$

$$K_{pq} = \int_V \eta_S\frac{\partial\psi_\beta}{\partial x_c}\frac{\partial\psi_\alpha}{\partial x_c}\delta_{ij} + \eta_S\frac{\partial\psi_\beta}{\partial x_j}\frac{\partial\psi_\alpha}{\partial x_i} + (\eta_B - \tfrac{2}{3}\eta_S)\frac{\partial\psi_\beta}{\partial x_i}\frac{\partial\psi_\alpha}{\partial x_j}dV \tag{1.13}$$

$$E_p = -\int_V \frac{\partial\psi_\beta}{\partial x_j}\sigma^e_{ij}dV \tag{1.14}$$

where α and β count over the nodes, i and j refer to the spatial dimension, and p and q are counting indices for the full dimensions of the FE system. p and q can therefore be written as $p(i,\beta)$ and $q(j,\alpha)$[1]. The return values of the p and q functions depend on the connectivity of the mesh. The vector N_p in equation (1.11) represents stochastic forces and has the following statistics:

$$\langle N_p N_q\rangle = \left(\frac{k_B T}{\Delta t}\right)(K_{pq} + K_{qp}), \langle N_p\rangle = 0 \tag{1.15}$$

where $k_B T$ is the thermal energy. Oliver showed further that for linear elements

[1]For example, for a 3-d linear mesh we have $i, j = 1..3$ and $\alpha, \beta = 1..4$, resulting in a 12×12 (mass or viscosity) matrix per element, consisting of nine 4×4 blocks.

N could be constructed locally. In this representation, each element represents a fixed set of atoms within the protein, so that fluctuations in molecular configuration are represented by the deformation of the element.

Oliver *et al.* introduced external fluid dynamics to the FFEA model through a simple, isolated Stokes drag on a sphere positioned on each node[44]. The size of each sphere is determined by the length scale of the mesh. The drag force F_p on node p is therefore given by:

$$F_p = -6\pi R_p \mu^s v_p \tag{1.16}$$

where $R(p)$ is the radius of the sphere around node p, μ^s is the external solvent viscosity and v_p is the velocity of node p. This dissipation leads to a corresponding noise vector $\underline{N}^e$ with the following statistics[44]:

$$\langle N_p^e N_q^e \rangle = \frac{12 k_B T \pi R_p \mu^s}{\Delta t} \delta_{pq} \tag{1.17}$$

Equations (1.16) and (1.17) describe the external fluid dynamics implementation used in the present work. A full hydrodynamics treatment of the external solvent through solving the Navier Stokes equation is non-trivial and beyond the scope of this thesis.

Equation (1.11) is integrated numerically to update the positions of the nodes through a given time interval Δt. As there is no mass transport between elements of a protein, the mass matrix M (equation (1.12)) is constant throughout any simulation, and is calculated only once, during the initialisation of the simulation. Any implementation of code solving this equation therefore concerns itself with

building the vector on the right hand side in the most efficient manner possible. In its naivest form, the viscosity matrices would be fully re-assembled at each step, but such an approach does not lend itself well to parallelisation, as discussed in chapter 2.

1.2.3 Experimental data and parameterisation

In order to apply FFEA to a biological system, it is first necessary to obtain a 3-d mesh of the relevant proteins. The structure of molecules can be determined from experimental data in several different ways, depending primarily on the level of structural resolution available. The higher resolution experimental techniques, such as X-Ray crystallography[46] and Nuclear Magnetic Resonance[47] can resolve detail at the atomistic level. Data from these techniques is therefore necessary to set up Molecular Dynamics simulations. However, there also exist low resolution methods of obtaining information about the structure of biological entities, notably Cryo Electron Microscopy (Cryo-EM)[48] and Small Angle X-Ray Scattering (SAXS)[49]. These low resolution techniques are unable to give information regarding the atomic level structure of the proteins, but they do provide a 3-d envelope describing its shape. Low resolution techniques are important particularly in biological contexts since not all proteins are amenable to crystalisation, and the process of drying out biological systems can partially denature or completely destroy the structure[50]. Cryo-EM involves rapid freezing of a specimen in solution (by plunging into liquid ethane), resulting in vitreous ice with no crystals[50]. This allows the specimen to keep its natural form, such that its different functional states can be observed.

Unlike existing methods of simulation, FFEA can use data as input from all of these experimental sources and, indeed, is particularly well suited to the low resolution case since the continuum representation has no concept of atomic positions. Consequently, FFEA is uniquely able to make use of the structural information provided by cryo-EM.

Aside from shape data, a continuum description of protein matter also requires estimates of density, viscosity and elasticity. The average density of biomolecules is of order 1500 kg m^3[51]. The viscosity of protein matter (on the time scales of interest to FFEA) can be measured through solvent based techniques and is found to be of order 10^{-3} Pa s[52]. The elasticity can be parameterised from experimental data (as in chapter 5), or, when such data is not available, a sensible guess may be made based on Atomic Force Microscopy (AFM) experiments on proteins such as lysozyme which place the Young's modulus in the range of 300 MPa to 700 MPa[53]. The Poisson ratio is generally assumed to be between 0.3 and 0.4[54].

1.3 Summary and aims

The general aim of this work is to continue the development of Fluctuating Finite Element Analysis[43], a technique which brings continuum finite element approaches down into the mesoscale region. More specifically, the aims of the work presented in this thesis are as follows:

- To produce an efficient, parallel implementation of the FFEA model, and to develop a pipeline for fully automated FFEA system creation from atomistic or low resolution data (see chapter 2).

- To extend the model to include new physics (see chapters 3 and 6).
- To compare the FFEA method against existing models, specifically the Elastic Network Model (see chapter 4).
- To apply the model to real biological systems (chapters 4 and 5).

A description of ongoing work and the future direction of the project is given in chapter 7.

Chapter 2

Simulation package and tools

2.1 Introduction

In order to facilitate the future adoption and continued usage of the FFEA model, it is necessary that there exist a user-ready simulation package and tools satisfying criteria described in the following categories:

1. Performance: In order to be able to exploit HPC resources, the chosen implementation must scale well with both system size and number of cores.

2. Usability: The pipeline from structural data to FFEA-ready input files should be automated as much as is feasible, handling all technical details that could be a barrier to less experienced users.

3. Analysis: A set of tools for manipulating, analysing and visualising the output simulation data must be provided.

4. Interoperability: The FFEA package exists amongst a great many other codes and visualisation programs, so convertability between FFEA files and

comparable formats is necessary to ease entry to this community.

This chapter will describe the development of a parallel, multiprotein implementation of `FFEA` and the corresponding `FFEA_tools` package. It will explain the procedure for typical usage, and detail the results of performance testing and validation of the code. An existing serial, single protein implementation will serve as additional validation of the new implementation.

2.2 Code and Parallelisation Strategy

2.2.1 Shared memory, Distributed memory and Amdahl's law

There are two major API[1]s for parallelisation of code in `C` and `Fortran` (among others), and the choice of which one to use is generally made based on what type of system architecture the program is to be executed on. For *distributed memory* systems, in which each individual processor has its own distinct memory space (only accessible by itself), we use `MPI` (Message Passing Interface)[55; 56]. For *shared memory* systems, in which multiple processors read and write from the same memory space, we might use the `OpenMP` (Open Multi-Processing) API[57]. In some cases (such as in certain supercomputers) there may be an architecture consisting of multiple shared memory units networked together. In this case a hybrid form of parallelisation may be used, involving both `OpenMP` and `MPI`. In recent years there has also been great interest in using the highly vectorised capabilities of graphics cards, particularly as these are typically optimised for

[1] Application Programming Interface

the matrix and vector operations common in graphics rendering (especially 3-d scenes). `CUDA`[58] is a popular parallel programming architecture developed by NVIDIA for such a use, and has seen usage in recent Molecular Dynamics applications[59]. In this project, `OpenMP` has been used for parallelisation. In future some level of hybrid parallelisation may be introduced, depending on the system architecture, but this is beyond the scope of this thesis.

An issue with parallelisation of any algorithm is that of communication costs between processors (such as in MPI) or the size of the *serial fraction* of the code (in OpenMP). These costs effectively define an upper limit for the number of processors after which no further increase in speed can be obtained. This is the basis of *Amdahl's law*:

$$\text{Max. Speedup} = \frac{1}{1 - f_p} \tag{2.1}$$

the theoretical maximum speedup (wall clock time on 1 processor divided by wall clock time on N processors) when f_p is the fraction of the code that is parallelised. This maximum is for an infinite number of processors, but in practice the limit can be reached quickly, even for large f_p (f_p close to 1)[60]. Note that the serial fraction $(1 - f_p)$ represents all interprocessor communication and synchronisation costs, and therefore Amdahl's law applies to all parallel programs (not just shared memory).

2.2.2 Implementation

Language

The `C++` programming language[61] was chosen for developing the simulation code for the reason that it is a well established, object oriented language with compilers

for more or less every known platform. It can produce fast and efficient programs using well tested compilers (e.g. `gcc`). Although some older highly optimised scientific programming libraries exist only in `Fortran`, `C++` can interface relatively easily with `Fortran` functions, and many modern libraries are built with `C` or `C++`, so this is not a significant issue.

The object oriented nature of `C++` is also an asset when producing large and complex programs. The ability for data encapsulation[62] was useful in implementing the many possible cases in multiple protein, multiple solver simulations. Although the `Fortran 90` standard onwards implements a number of the features of object oriented languages, and as such would be suitable to tackle such issues, its (intentionally) limited memory pointer capabilities and smaller set of standard list types and containing classes render it a less attractive option than `C++` for the complex data types required in this project.

Algorithm

In FFEA the conformation of the protein is evolved in time according to the equation of motion (1.11) from section 1.2.2:

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

Here $\vec{u}$ is the vector of node velocities, M is the mass matrix and K is the viscosity matrix. The vector $\vec{E}$ contains all the continuum forces arising from elasticity, van der Waals (see chapter 3) and electrostatics (see chapter 6), while $\vec{N}$ is the stochastic forcing from thermal fluctuations. This equation is solved using a first order (forward Euler) time-stepping scheme. As noted by Oliver *et al.*, the

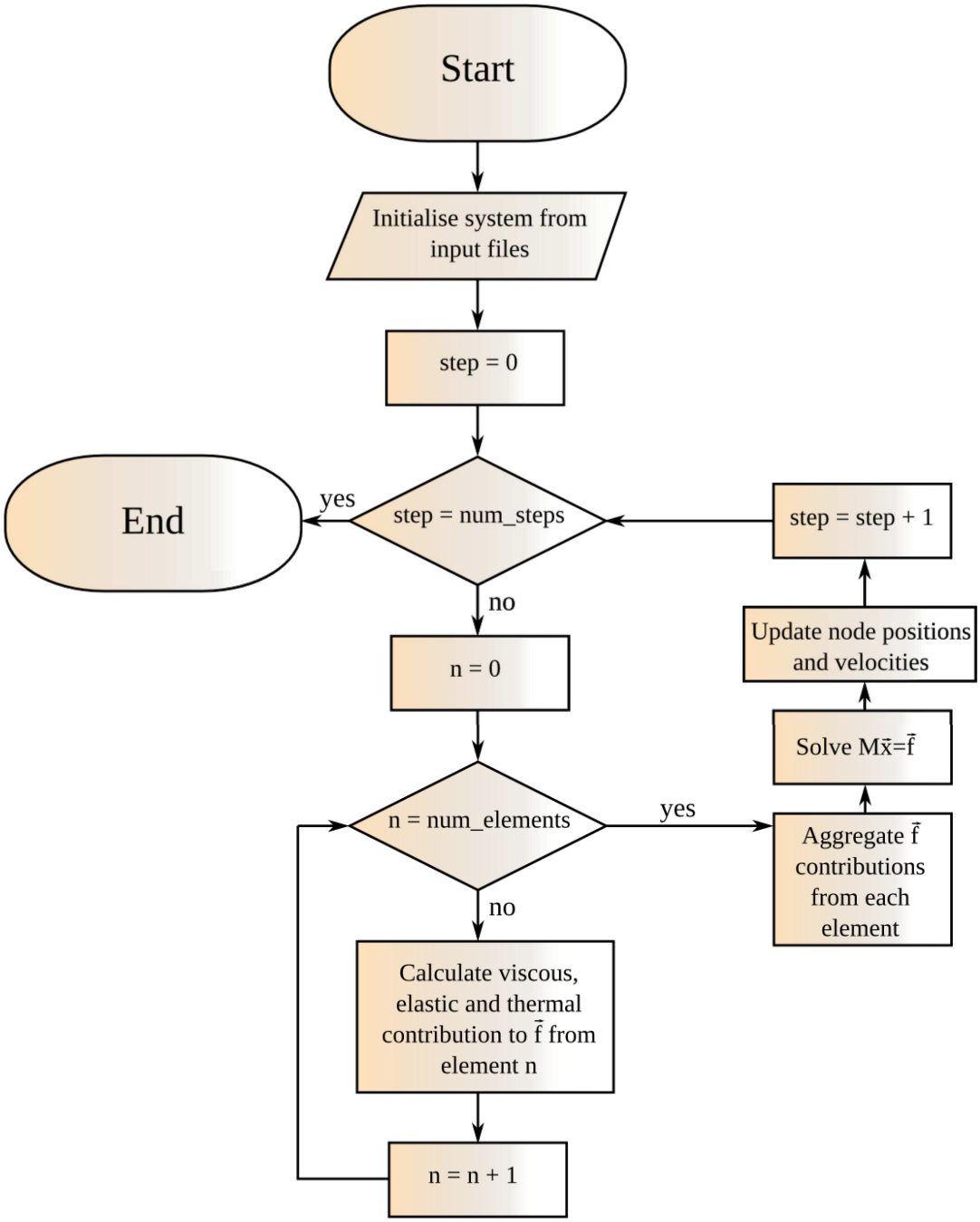


Figure 2.1: A flowchart of the basic FFEA algorithm used in this project. For simplicity, the algorithm is given for a single protein with no external forces (more detail is given in Appendix A). There are synchronisation barriers after the element loop, the force aggregation, the solve step and the numerical integration, due to each of these steps requiring the results of the calculation in the previous step. These barriers are the main contributors to the 'serial fraction' $(1 - f_p)$ from Amdahl's law, whose effects can be seen for small systems in figure 2.4.

presence of the stochastic term means that the size of the time step is dictated by stability rather than accuracy so that a higher order is not beneficial. Once the velocities of the nodes are known, the positions of the nodes can be updated. A simplified flowchart of the algorithm for a single protein with no external forces is given in figure 2.1. A more detailed pseudo-code description is given in Appendix A.

Matrix structure

The mass matrix M is sparse and positive definite[1]. Furthermore, the three coordinate directions (x, y, z) are independent from one another with identical components, so that it has a block diagonal structure in which the block for each spatial direction is identical. Consequently only one block is created and used for all three spatial components.

The viscosity matrix K has similar sparsity to M, but contains coupling of the different spatial directions.

Both matrices can be assembled from the local contributions to each element. Indeed, the code never generates the full sparse matrix for the entire system, but instead stores the contributions from each element individually, summing these contributions after all the elements have been processed. This independency allows work sharing of the element loop, in which all the elements are divided up between the available processors, and processed in parallel.

The main problem in parallelising this code lies in the "solve" step, in which $M\vec{a} = \vec{f}$ must be solved for $\vec{a}$. A typical direct method for solving this equation is to perform a Cholesky decomposition[63] of matrix M. We then perform a

[1] A matrix A is positive-definite if $\vec{x}^T A\vec{x} > 0 \ \forall \vec{x}$.

forward and backward substitution with the resulting upper and lower triangular matrices to find $\vec{a}$. However, the parallelisation of the forward and backward substitution still requires the starting and stopping of parallel regions $2N$ times, where N is the number of nodes. The overhead for starting parallel regions is sufficiently high that this only provides a speed increase in the case of large N and for matrices much denser than we typically find in this model. In fact, the size of M required before the parallel version shows any significant speed-up is so large that iterative methods will have long outstripped it by that point.

The alternative is to use an indirect method to solve matrix equations. Since M is a symmetric positive-definite matrix, we can use the *Conjugate Gradient Method* [1][64]. As the Conjugate Gradient algorithm is composed of repeated matrix-vector multiplications and dot product calculations, it lends itself to parallelisation. Iterative methods such as this can be more efficient than 'direct' solvers for large sparse matrices (see figure 2.5), particularly if they have a large bandwidth[2] (since a Cholesky decomposition will fill-in all elements between the initial and final non-zero elements on each row, potentially destroying the sparseness of the matrix). To speed up the rate of convergence of the Conjugate Gradient method it is normal to use a preconditioner[65]. Fortunately the matrix M is sufficiently diagonally dominant that a simple Jacobi-preconditioner[3] provides sufficiently rapid convergence. The parallelised code for the Preconditioned

[1]To solve for $\vec{x}$ in the matrix equation $A\vec{x} = \vec{b}$ using the iterative CG method, we define a residual, $\vec{r}_i = \vec{b} - A\vec{x}_i$, where r_i is the residual after the i^{th} step. The residual is exactly zero if $\vec{x}$ is the true solution. Starting from an initial guess for $\vec{x}$, $\vec{x}_0$, the CG algorithm seeks to minimise $\vec{r}$ by moving $\vec{x}$ along vectors conjugate with A.

[2]The length, in number of entries, between the first non-zero entry and the last non-zero entry in a matrix row.

[3]The inverse diagonal of the matrix.

Conjugate Gradient solver (PCGS) involves an initial *barrier*[1] followed by four barriers per iteration. In the typical systems considered for this project, the number of iterations was typically around 12, leading to (very) roughly 50 barriers per solve. An unfortunate result of this is that the solve step is not as scalable as the assembly loop. The number of iterations, however, remains fairly constant even up to very large systems[2], so the PCGS is generally a better choice than the Direct solver for systems comprised of a large number of elements (see figure 2.5).

With regards to parallel efficiency, the manner in which data is divided up and sent to the various cores is also important. When parallelising a for loop, one has three choices: divide the entire loop into approximately equal 'chunks' and distribute to all threads immediately; break loop into small, uniform 'chunks' and distribute to idle threads on a first come first served basis; or progressively reduce the 'chunk' size, starting large. These three types of 'scheduling' are known as *static*, *dynamic* and *guided*, respectively[66]. The latter two schedules are designed to smooth the load imbalance caused by the different wall clock time taken by different threads to process the given data. In the shared memory implementation of this code, it was found that a *guided* schedule for processor intensive loops (such as the element calculation loop, or force aggregation) was more efficient, and a *static* schedule worked best for naturally load balanced, short calculations (such as the Euler integration over all nodes). This is simply due to the fact that the overhead in the *guided* schedule outweighs the benefit

[1] A synchronisation 'barrier' refers to any point in an algorithm at which parallel threads must wait until all threads have reached this point. This is necessary when the next step in the algorithm requires the combined result of a previous step (e.g. a dot product operation).

[2] Typically 6 to 10 iterations were sufficient for convergence for small systems (1000 elements or less). This rises to 12 or 13 for large systems (50000 elements).

it brings in short calculation loops. The most effective schedule to use can in principle be architecture dependent, however these choices were found to work well on both the ARC1[67] and Polaris[68] supercomputers[1], as well as various x86 AMD workstations.

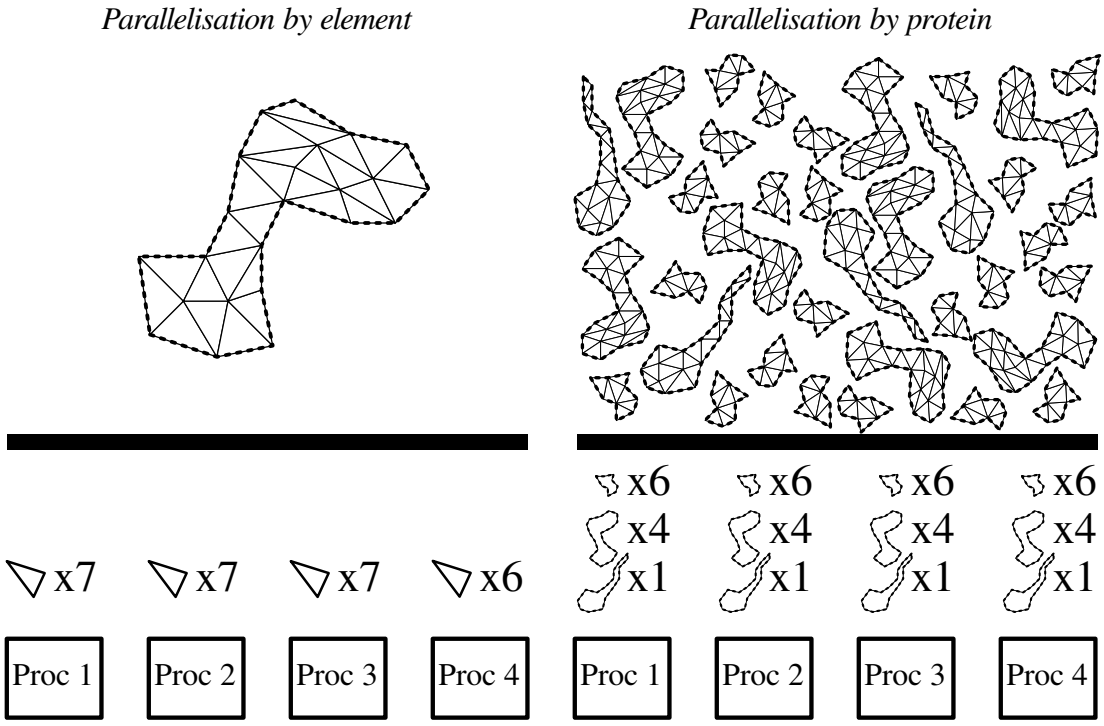


Figure 2.2: An illustration of the parallel subdivision of two simulation systems between four cores in the case of a single large protein (for which the *by element* scheme is favoured) and for a large ensemble of interacting proteins (using the *by protein* scheme). Both the matrix construction and solution steps are affected by the choice of parallelisation scheme.

The above describes the parallelisation strategy used for simulations of one or a few very large element number proteins. It is, however, not the optimal strategy for simulating large numbers of interacting proteins. In such a case, it is more

[1]The Advanced Research Computing Node 1 (ARC1) and the N8 HPC Polaris are supercomputing facilities based at the University of Leeds[67; 68].

efficient to process entire groups of proteins on each core, rather than dividing up their elements as this reduces the number of synchronisation barriers drastically. Figure 2.2 gives an illustration of the two schemes implemented in `FFEA`, termed *by element* and *by protein*.

Random Number Generation and Parallelisation

The stochastic vector $\vec{N}$ requires the code to generate $7N_e$ random numbers per time step, where N_e is the number of elements. The time required can therefore become significant as N_e increases. Equally, as the model is fundamentally stochastic, the *quality* of the random numbers produced is also very important (uncorrelated output, uniform distribution, large period etc). An example of a rapid Random Number Generator (RNG) (and indeed the choice for this code) which performs well on tests of statistical randomness is the Mersenne-Twister (MT) algorithm developed by Matsumoto *et al.*[69]. The MT RNG is notable for its large period of $2^{19937-1}$ and speed[69] (particularly the SIMD-oriented Fast Mersenne Twister, developed in 2006[70]).

There are some troublesome issues surrounding the parallelisation of the RNG. If only one RNG is shared by all threads then there will concurrent modification of the state of the RNG, resulting in unpredictable behaviour that is not guaranteed to fulfill the criteria of statistical randomness. Threads accessing the RNG at almost the same time may end up producing the same random numbers, for example. One solution is to 'lock' the state of the RNG so that only one thread may access it at a time. Clearly, however, this will have an enormous impact on the performance of the code and make its scalability very poor. The method employed in this project is for each thread to have its own, individual RNG. This

solves the concurrent modification and scalability problems but introduces a new, more subtle issue - each RNG requires a different seed, but not all seed combinations are guaranteed to produce uncorrelated parallel RNGs (though methods exist to produce independent parallel RNGs for different process identifiers[71]). Similarly, depending on the availability of a particular processor in a given time step, it may process more or fewer elements. This means that running the same simulation again with the same seeds will not necessarily produce the same results, depending on which processor (and therefore which RNG) an element is allocated to. However, this non-deterministic behaviour does not affect the validity of the physics itself; the resultant dynamics will still be physically sound.

Lookup tables

The van der Waals (chapter 3) and electrostatics (chapter 6) calculations both involve cut-off distances beyond which two faces can be said to be not interacting. In order to exploit this for improving performance, it is necessary to build and maintain a 3-d lookup table, partitioning the simulation box into a number of cells within one of which each face can be said to reside. The electrostatics solver is required to build two types of matrix; one for which the sparsity pattern is known at the start of the run, and one for which it is unknown and can change every step. Building these matrices and lookup tables consumes a lot of processor time and therefore a scalable method of building the relevant data structures was necessary. This is discussed in more detail in chapters 3 and 6.

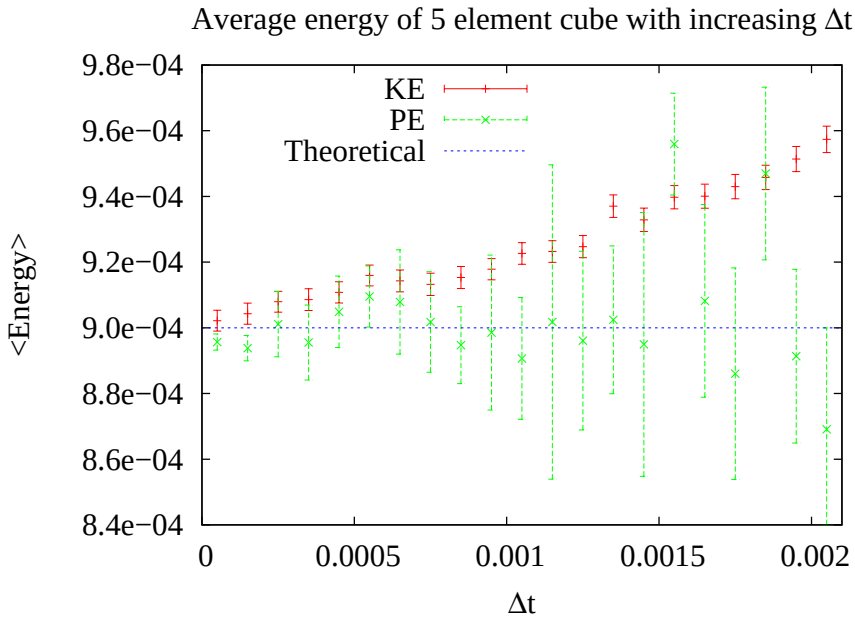


Figure 2.3: The kinetic and potential energy of a five element cube under an Euler integration scheme, averaged over 1×10^9 time steps, for various values of Δt. Blue line shows the theoretical average upon convergence (with $k_B T = 0.0001$).

Validation

The initial testing of the code was performed on a five element cube. The 'validation' of the code here comes in the form of checking that the following two conditions are met for an FFEA protein *in vacuo* (no external forces):

1. After a suitable equilibration period the average kinetic and potential energy of the simulated system at thermal equilibrium converges to $\langle KE \rangle = \langle PE \rangle = \frac{3N-6}{2} k_B T$where N is the number of nodes.

2. The angular momentum at every time step is 'zero' (in practice 'negligible', due to machine precision errors).

Satisfying the above by no means ensures that the code is entirely sound, but it gives confidence since, in practice, this should uncover many (but not all) coding errors. Figure 2.3 shows the average kinetic energy and potential energy of a 5 element cube (of unit size) against time step size under an Euler integration scheme. The figure shows the expected convergence behaviour for the $\langle KE \rangle$, and the $\langle PE \rangle$ (which converges very quickly, although it has a much higher variance, as can be seen in a couple of anomalous points). There are 3 degrees of freedom per node, from which we subtract 3 translational and 3 rotational degrees of freedom as all forces are internal (in this test case). Applying equipartion theorem, we have $\frac{k_B T}{2}$ per degree of freedom, resulting in the analytically determined value plotted in 2.3 (for $N = 8$). As both the KE and PE converge to the theoretical value with reducing Δt, we conclude that the first validation criterion is met.

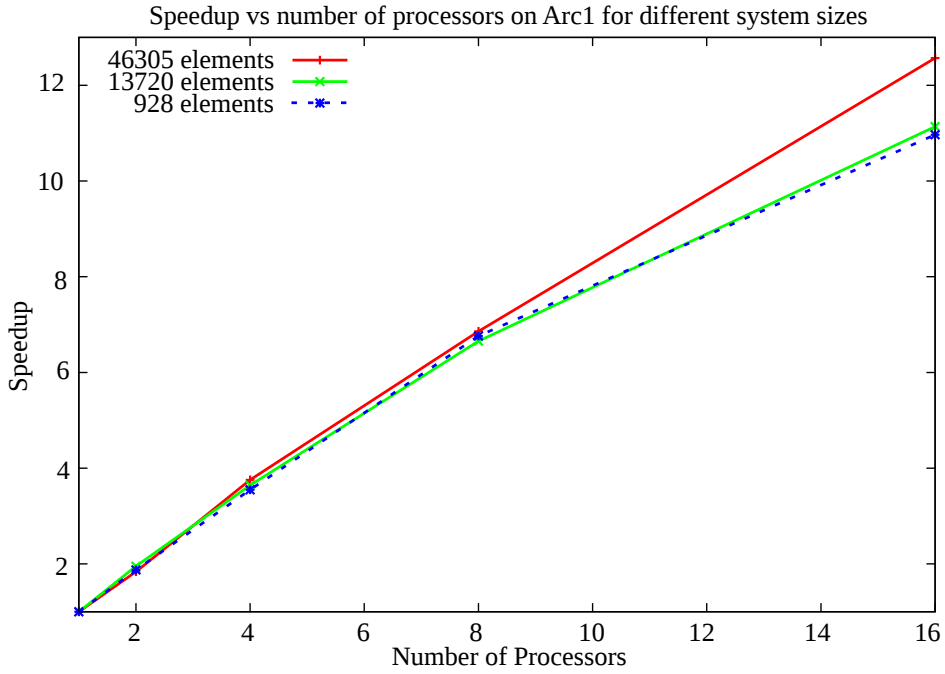


Figure 2.4: Simulation speedup vs number of processors for various system sizes.

Scalability

Figure 2.4 shows the results of benchmarking the code using the ARC1 high performance supercomputing facility, on a series of cubes with different numbers of elements. The speedup is defined as the average wall clock time to complete a task with 1 processor divided by the average wall clock time to complete a task with n processors. The wall clock run times used in the calculation of speedup were obtained from the average of 20 separate runs on ARC1's AMD processor shared memory nodes. No more than 16 processors were used due to this being the maximum number of processors on a shared memory node. A larger number of processors would result in extra communication costs and therefore would mask the true scalability of the code in a pure shared memory environment.

The code shows acceptable speedup with number of processors for large prob-

lem sizes. For the largest problem (a cube of 46305 elements) we find a speedup of 12.6 with 16 processors. Note that the same sort of parallel efficiency can be obtained with fewer elements if more intensive work is carried out per element. For example, replacing the first order (linear) elements, used here, with 2^{nd} order (quadratic) elements would result in high parallel efficiency for systems with far fewer elements.

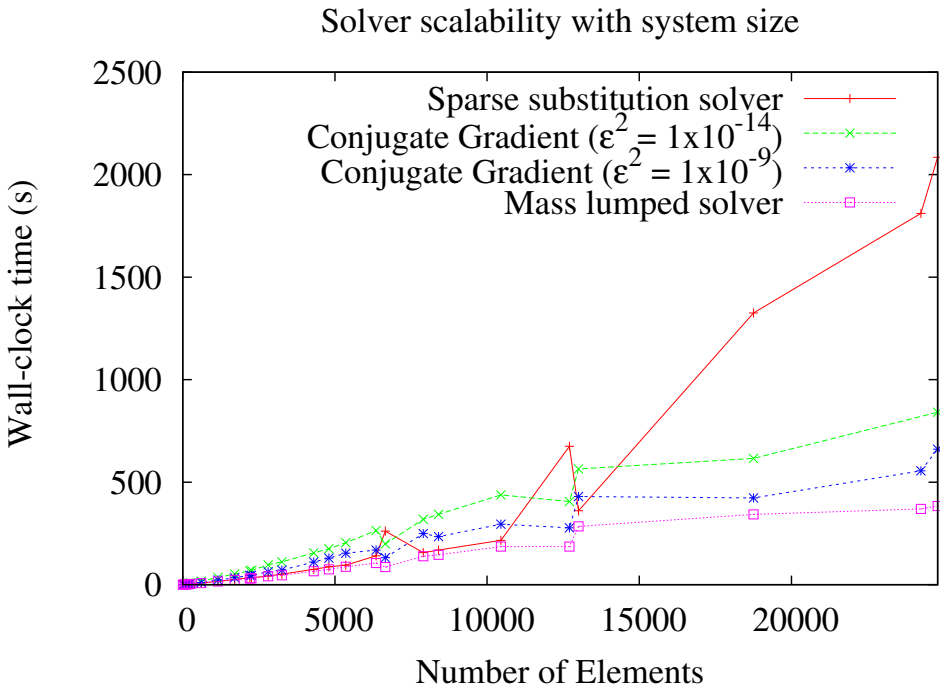


Figure 2.5: Effect of solver choice on time taken for FFEA code to complete ten thousand time steps for increasing number of mesh elements. The meshes are cuboid, with varying dimensions and connectivities. The time step, material parameters and spatial size of the system do not affect the time taken.

Three different linear solvers for the $M\vec{a} = \vec{f}$ step (discussed in 2.2.2) are compared in figure 2.5. The first is the direct method using a Cholesky factorisation, referred to here as the "Sparse Substitution Solver". The second is the preconditioned conjugate gradient method with Jacobi preconditioner. The third

is not an actual solution of the linear system, but an approximation derived from lumping all the mass terms on each row of M onto the diagonal.

Figure 2.5 shows the time taken to complete 10^4 time steps for cuboid meshes of varying dimensions and element number, and the scalability of each solver. The fluctuations (most pronounced in the direct "sparse substitution solver") are the result of connectivity differences in the mesh. While the current FFEA algorithm has its bottleneck in the computation of internal element forces, the mass matrix has the dimensionality of the number of nodes in the system. This means that for large systems, the number of nodes in the mesh can become significant. It is unsurprising, therefore, that the "mass lumped" case (in which the mass matrix is diagonal) shows an almost constant scaling, and the best speed of all the solvers. In the case of distributed mass, the direct sparse substitution solver is fastest for smaller system sizes, but quickly becomes uncompetitive for systems larger than around ten thousand elements. This is due to the fact that Cholesky decomposition of a sparse matrix "fills in" all matrix elements between the first and last nonzero element of each row. While bandwidth minimisation algorithms exist to minimise this fill-in, large or high connectivity meshes will always compromise the original sparsity of the mass matrix. The iterative Conjugate Gradient solver requires only matrix multiplication, and therefore wins out over the sparse substitution method for larger system sizes. Figure 2.5 shows the effect of altering the PCGS's error tolerance, ϵ. The shape of the scalability curve is essentially the same, as the code merely takes more or less iterations to complete. While mesh shapes other than cuboid (spherical, cylindrical, etc.) will produce their own particular scaling, the figure shows the general trend, and suggests that the choice of solver to use for each protein in the simulation should be based on the

elements to node ratio of its mesh. It should also be noted that the analysis provided in figure 2.5 strongly favours the Sparse Substitution Solver due to the use of a growing cuboid as test mesh. It should be noted that this shape produces meshes with low bandwidth, minimising fill-in from the Cholesky decomposition of the mass matrix and thus allowing this solver to remain competitive up to a larger system size than typically seen with meshes for real applications.

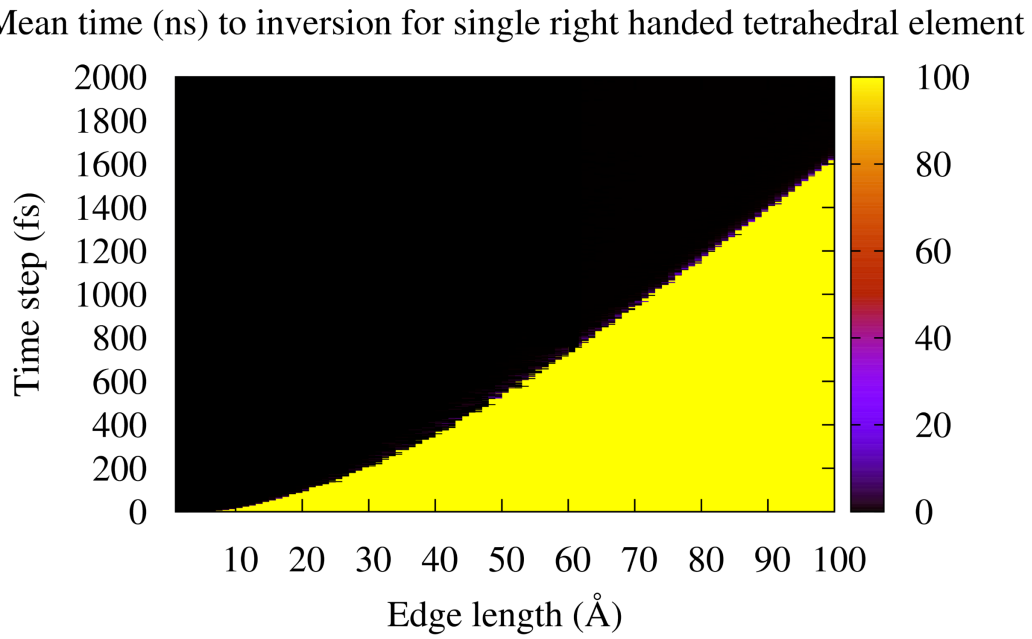


Figure 2.6: Stability analysis of a single, right handed tetrahedral mesh element for different values of time step and edge length. Stability is measured as mean time to inversion. Simulations completing ten runs of (arbitrarily) 100 ns without inversion are considered stable, and are shown in yellow. Simulations with a mixtures of inversions and stable runs are considered metastable and shown in red/orange. Note the acutely sharp boundary between stable and unstable; very few simulations resulted in times between the two extremes (0 ns and 100 ns) despite each data point being the average of ten separate runs.

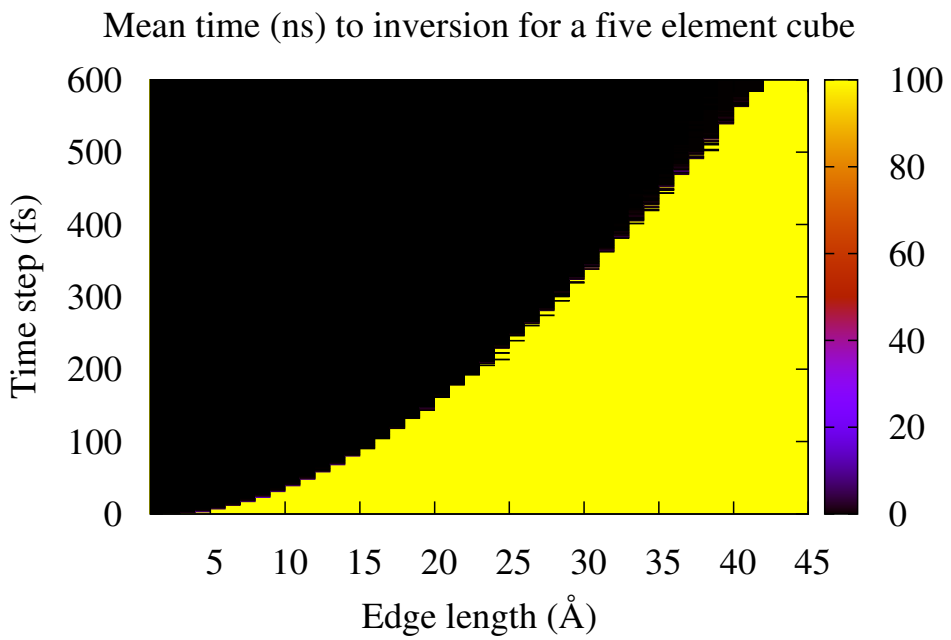


Figure 2.7: Stability analysis for a 5 element cube for different choices of time step and edge length. Stability is measured as mean time to inversion. Simulations completing ten runs of (arbitrarily) 100 ns without inversion are considered stable, and are shown in yellow. Simulations with a mixtures of inversions and stable runs are considered metastable and shown in red/orange. The stable/unstable boundary shows the same sharpness as in figure 2.6.

Finally, the ability of the code to simulate long time scales is also highly dependent on the size of the time step, Δt. Due to the random nature of the thermal noise, there is always a finite probability of an element inverting or becoming unstable, due to random forces stacking up in one particular direction. Modifying Δt is, in a sense, simply changing the probability of an element inverting during an FFEA simulation. Figure 2.6 shows the result of a stability analysis for a mesh comprised of one right handed tetrahedral element with the material properties given in table 2.1 (using some typical values for the density[51], viscosity[52] and Young's modulus[53] of protein matter as discussed in section 1.2.3). The mean

Density	1500 kg m^{-3}
Young's modulus	338.8 MPa
Poissons Ratio	0.41
Shear Viscosity	1 mPa.s
Bulk Viscosity	1 mPa.s
Time step	10 fs
Temperature	290 K

Table 2.1: Material parameters used in the stability analyses shown in figures 2.6 and 2.7.

time to inversion is calculated for varying edge lengths and time steps, with the average of 10 runs at each parameter pair. Cases where no element inverts during an arbitrarily chosen time of 100 ns are considered "stable". Changing this time will decrease the size of the stable (yellow) region, and in the limit of infinite simulation time, all elements would eventually explode. Nevertheless, the analysis shows that there is a very sharp and well defined phase boundary for each element size, across which the mean time to inversion increases dramatically. In practice, the meshes being simulated are comprised of many thousands of elements, and the surrounding mesh has several stabilising effects. These can be seen in figure

2.7 which shows the results of a similar stability analysis for a 5 element cube (but for a smaller range of parameters). For example, we can see that for an edge length of 40Å, the mean time to instability increases from around 330 fs in the single element case, to around 530 fs in the 5 element case, illustrating the sizable effect the neighbouring elements have on mesh stability. The stability diagram is therefore dependent on the specific mesh (and material properties) being used. As a general rule, however, the mean time to inversion is determined by the volume of the smallest element in that mesh.

2.3 Preparation of an FFEA system

The process of transforming experimentally derived structural and dynamical data into a form appropriate for the FFEA model takes the following steps:

1. Creation of a protein surface mesh from the structural data (section 2.3.1).
2. Coarse-graining of the protein surface mesh (section 2.3.2).
3. Meshing of the protein interior and assignment of material parameters (section 2.3.3).
4. Positioning within protein ensemble and assignment of exterior interactions (section 2.3.5).

2.3.1 Creation of a surface mesh

The input files used by `FFEA` can be generated from any cryo-EM density map, PDB structure or SAXS envelope. The method used throughout this thesis relies

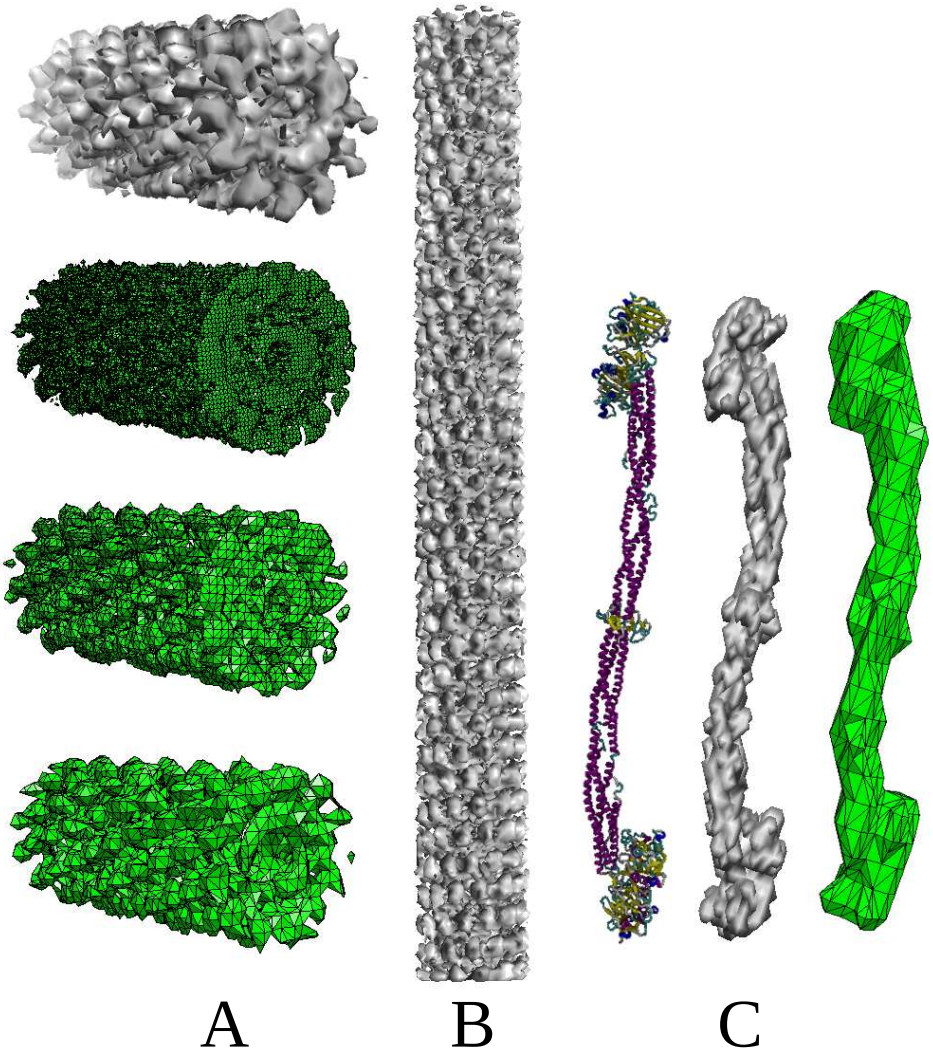


Figure 2.8: **A**: A cryo-EM density map of a section of a measles virus (top). Surface mesh output from `FFEA_tools meshmap` at various levels of coarsening shown below. **B**: Use of `addmap` function to produce a longer measles system. **C**: An example of a 3GHG PDB structure[72] (Left) being converted to an MRC CCP4 electron density map using `FFEA_tools pdbtomap` (Centre), then meshed using `netgen` (Right).

on first producing a density map[1] When starting from a PDB structure, each point in the PDB file is assigned a particular sphere radius (generally 20 to 30 Å). The density contribution is chosen to vary linearly from 1 at the centre of the sphere, to 0 at the edge. The density contributions of these spheres are then mapped onto a cartesian 3-d grid. Figure 2.8C illustrates this conversion from a PDB structure to a density map. The same method could be applied in the conversion of SAXS data to a density map (although no SAXS data is used for the work presented in this thesis), using the SAXS 'particle beads' as the points. In some cases, such as the measles virus shown in figure 2.8B, the data only represents a short section and so in this case density maps are combined into a larger system formed from smaller repeating units.

A surface mesh can then be produced from the resultant density map using the marching cubes algorithm[73]. This requires a contour level (isovalue) determining the surface. For maps obtained from the EMDB there is typically a 'recommended' level at which the relevant biological features are deemed to be present. The density map may contain 'dust' (extraneous specs of density related to noise in the experimental data) which must be removed. There may also be small cavities present (at the chosen contour level) which are problematic as they introduce small length scales to the system and create poorer quality meshes during the 3-d interior meshing step. Using a modified version of the flood fill algorithm (implemented in three dimensions rather than the usual two), all bodies and cavities can be identified. Any bodies below a certain volume (measured in number of voxels) can be designated 'dust' and culled. Similarly, any cavities

[1]Clearly this is unnecessary in the case of cryo-EM, for which the data is already in this form.

below a certain volume may be filled in. Filling cavities is particularly necessary in the case of maps produced from atomistic structures, which are frequently riddled with holes rendering any output mesh virtually unusable. Figure 2.8C shows the full conversion from atomistic structure to density map to surface mesh.

2.3.2 Surface coarse-graining methods

Having obtained a surface mesh, it is now necessary to coarsen it. This is to avoid the small length scales which make the FFEA model unstable and necessitate small Δt (as discussed in section 2.2.2). Three different methods are employed, each being appropriate in certain cases.

The *Face pair collapse* (FPC) method iteratively finds the smallest edge length in the mesh and culls it, collapsing the two parent faces. Each cull removes one node, two faces and three edges. This scheme is continued until a threshold minimum edge length is reached that depends on the smallest biologically significant feature of the mesh. This method works well as a 'polishing' stage, but can become stuck when handling meshes with complicated connectivities. The FPC method is also poor at keeping the overall shape (since it will always cull the smallest edge, regardless of whether this is an important feature).

For meshes with a mixture of large smooth regions (which can be heavily coarse grained with no loss of important detail) and small finer regions (which would be culled by the FPC method) it is possible to use the *Decimation* approach from the `blender` 3-d modelling package[74]. This method takes into account the overall shape of the mesh and attempts to remove a proportion of faces such that the shape is preserved. However, like the FPC method this approach does not

work well with meshes with highly complex connectivities (such as that generated from the measles map in figure 2.8A), and is prone to producing intersecting surface elements at high coarse-graining levels.

A final method of coarse-graining surface meshes which can be applied when FPC and decimation fail, is *voxel averaging*. This method is applied before the creation of a surface mesh, and coarsens the density map itself by grouping together cubes of voxels and averaging the densities. For example, coarse grain level 3 would correspond to averaging each $3 \times 3 \times 3$ block of voxels into one average voxel. This is the coarse-graining being represented in 2.8A. The drawbacks of this method are that it is very crude and can easily average over important detail in the density map.

In practice a mixture of the above three methods may be used if necessary (for example, decimation or voxel averaging first, and FPC to finish).

2.3.3 FFEA model creation

The coarse-grained surface mesh can now be meshed using any standard 3-d meshing software such as `NETGEN`[1][75] or `TETGEN`[76]). The output mesh is then converted into the FFEA input files described in table 2.2. These FFEA files must satisfy a number of requirements (such as all surface face normals pointing outwards) and encode extra information (such as which nodes lie at the surface) that is not generally provided by `NETGEN` or `TETGEN`. At this stage it is also possible to cull small (below a given volume threshold) elements, although this should be used with caution. Such culling is appropriate in the case of meshes with a large

[1]For the purposes of the FFEA project, `NETGEN` version 4.4 should be used, as later versions produce meshes with small elements by default.

FFEA input file	Contains
.node	Node positions of mesh
.top	Topology (element connectivity) of mesh
.surf	Surface faces and parent element
.vdw	Face 'type' for van der Waals interaction
.mat	Material properties (density, bulk and shear modulus, bulk and shear viscosity) of each element
.stokes	Stokes radius of each node
.pin	Indices of nodes to be 'pinned' (immobilised) during simulation.

Table 2.2: The 7 input files describing an FFEA protein.

number of very small surface elements whose removal changes little about the shape or connectivity of the mesh. The whole FFEA system is described in a .ffea file, an example of which is given in Appendix B.

2.3.4 Simulation states

Each FFEA model in the simulation can be assigned one of three states: 'DYNAMIC', 'FROZEN' and 'STATIC'. 'DYNAMIC' is the default state, in which all the protein's dynamics are simulated in full. A 'FROZEN' protein has its dynamics temporarily disabled, but can be set to 'DYNAMIC' at any time and resume its function. This is useful in cases where the protein has more than one conformation with different mesh topologies (see chapter 5). 'STATIC' proteins are fixed, unmoving surface meshes only. This state is useful for large, rigid (on the scale of the simulation) structures, such as the microtubule surface in chapter 5. It avoids the computationally expensive dynamics calculation, while allowing van der Waals interactions (described in chapter 3).

2.3.5 Visualisation

A custom visualisation program, `FFEA_viewer` was produced for the project, designed to read a given `.ffea` input script file and render the trajectory with various visual settings. While `FFEA_tools` is able to convert `FFEA` trajectories to, for example, PDB files, for viewing with existing visualisation software (e.g. `VMD` [77] or `chimera` [78]) the viewer is designed to allow `FFEA` specific editing. For example, specifying the van der Waals type of individual surface faces of a protein can be done visually through point-and-click interaction using this viewer. The viewer is written in `python`, using the standard `tkinter` for the control panel and with `OpenGL` bindings through `pythongl`. Performance increases are brought in via `cython`.

2.3.6 `FFEA_tools`

As detailed above, FFEA input files can be generated from any cryo-EM density map, PDB structure or SAXS envelope, but this process requires a lot of "cleaning" and sanitisation of the resultant meshes to be carried out. As such, the process of manually preparing input files for FFEA can be laborious and opaque. The `FFEA_tools` program seeks to remedy this by providing a number of fully automated conversion and clean up tools. An example of using the `FFEA_tools` suite to automate the process of setting up an FFEA system is given in Appendix C.

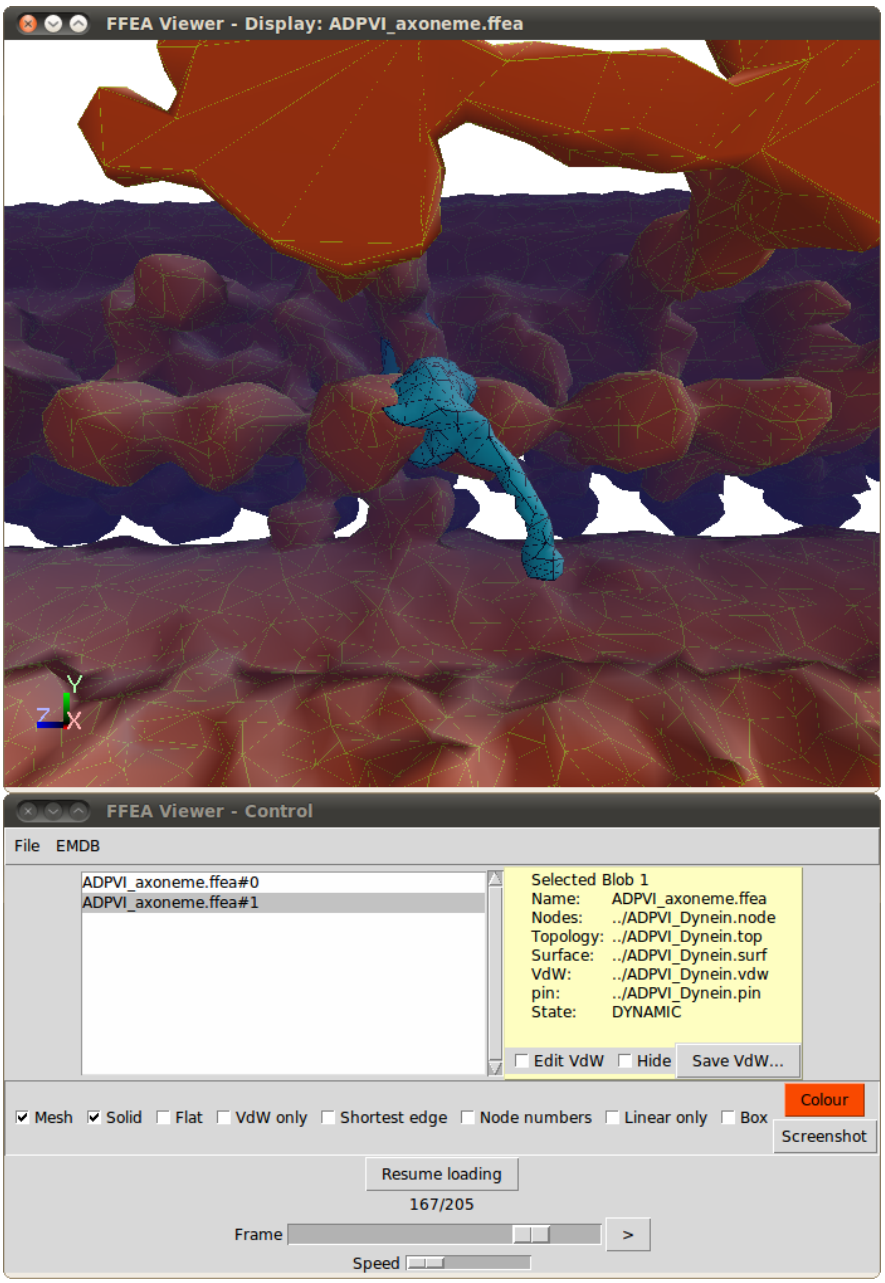


Figure 2.9: A sample screenshot of the `FFEA_viewer` visualisation program, displaying an FFEA trajectory of molecular motor dynein interacting with the axoneme.

2.4 Summary

A shared memory implementation of the FFEA model has been produced which scales reasonably with the number of processors. A pipeline has been created to automate the production of FFEA files given atomistic or cryo-EM input files, and several coarsening methods have been explored and discussed. The scaling with system size was investigated for different solvers, and the effect of mesh connectivity on simulation speed. The stability of a single element and a group of elements under the explicit, inertial scheme was analysed. A viewer was created for use in editing and visualising FFEA simulations.

Chapter 3

van der Waals interaction

3.1 Introduction

The mesoscale model described in chapter 1 includes internal forces within the protein only and neglects surface forces preventing interpenetration of surfaces. This is sufficient for modelling experimental data from isolated proteins such as that produced by SAXS[44], as this technique focuses on the low resolution shape of individual proteins, seeking to eliminate aggregation or crowding effects[79]. However, the FFEA model is by design very versatile and scales (to a good approximation) linearly with system size. It is therefore desirable to consider the case of very many interacting proteins, with complex and varied geometries, such as is the case in the cell cytoplasm[80]. This clearly requires an accurate and efficient implementation of interactions between the proteins. One such important interaction is the electrostatic interaction (implemented in Chapter 6), but aside from this there are a myriad of other important forces present in a protein solution that must be taken into account. Perhaps most obviously, there is steric repul-

sion. That is to say, the protein cannot pass through itself or another protein. There also exist complex protein-solvent interactions due to the effect different functional groups have on water's ability to form a hydrogen bond network, and the associated energy penalty. This can be greatly simplified by treating different parts of the protein surface as either "hydrophobic" or "hydrophilic".

3.2 'van der Waals' forces in simulations

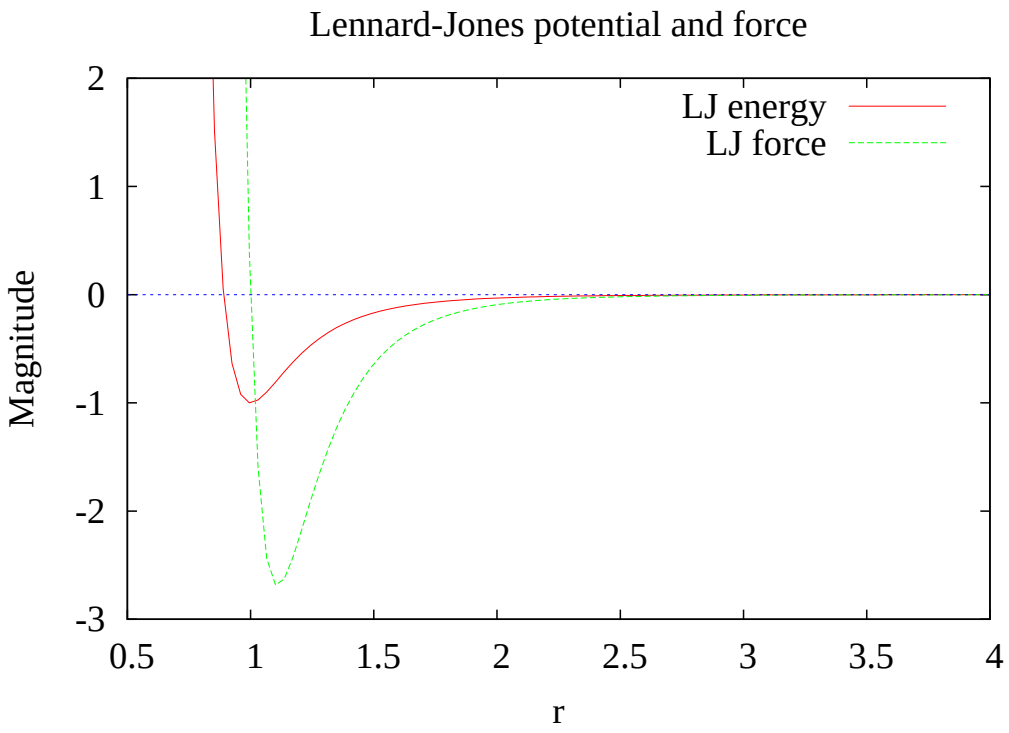


Figure 3.1: Magnitude of the Lennard-Jones potential (equation (3.1)) and the corresponding force (equation (3.2)) for an equilibrium separation $r_{eq} = 1.0$ and well depth $\varepsilon = 1.0$.

In simulation techniques such as Molecular Dynamics, the combined steric

repulsion, hydrophobicity and dispersion forces between two atoms (or 'beads') are often modelled using the simple Lennard-Jones potential[81]. The functional form of this potential, E_{LJ}, is given by

$$E_{LJ}(r) = \varepsilon_{LJ}\left(\left(\frac{r_{eq}}{r}\right)^{12} - 2\left(\frac{r_{eq}}{r}\right)^{6}\right) \tag{3.1}$$

where r_{eq} is the equilibrium separation distance and ϵ_{LJ} is the depth of the energy well at $r = r_{eq}$. This results in the following force

$$F_{LJ}(r) = -\frac{-dE_{LJ}(r)}{dr} = 12r_{eq}^{6}\varepsilon_{LJ}\left(\frac{r_{eq}^{6}}{r^{13}} - \frac{1}{r^{7}}\right). \tag{3.2}$$

The force and energy are plotted in figure 3.1. For separations $r < r_{eq}$, the potential is strongly repulsive, modelling the hard, steric repulsion between atoms. For separations with $r > r_{eq}$, the potential is weakly attractive, modelling the dispersion forces between atoms. At high r (typically $r > 3r_{eq}$) the force is sufficiently close to zero to be neglected, reducing the amount of computation required to model systems with large numbers of atoms.

There are other potentials in use, notably the Morse potential[82] and the Buckingham potential[83], which approximate the basic shape of the LJ potential. While the following implementation can accept these other potential forms, the LJ potential was sufficiently inexpensive for the needs of the project, and is the potential used for simulations in chapters 4 and 5.

3.3 Considerations for FFEA implementation

The approach considered here treats vdW forces in the FFEA model as the continuum limit of vdW forces in an MD simulation. That is to say, the finite sum of pairwise force contributions in MD becomes a volume integral for every infinitesimal volume in the system. This is represented below

$$\sum_{j \neq i}^{N} \vec{F}(\vec{r}_i - \vec{r}_j) \longrightarrow \int_{\Omega_i} \int_{\Omega_j} \psi_i(\vec{r}_i) \vec{F}(\vec{r}_i, \vec{r}_j) dV_i dV_j \tag{3.3}$$

where Ω is the entire protein domain, $\psi_i(\vec{r}_i)$ is the shape function corresponding to node i (in the finite element formulation), and N is the number of atoms.

Thus, for each element equation (3.3) requires the sum of interactions with all parts of all proteins including itself. Given the continuum nature of the model, this could clearly be very costly. These problems can be avoided, however, with a number of physical justifications:

- *The protein interior can be excluded*: The effects of vdW interactions between the atoms that compose the interior of the protein have already been taken into account through the moduli, viscosity and noise terms in the current equation of motion. Adding a vdW term for this would create an additional, unwanted coupling, making parameterisation of the protein's mechanical properties very difficult.

- *Solvent interactions are 'short range'*: The disruption of the water network that introduces the energy penalty driving the attraction occurs on a very short range relative to the length scales normally explored with this model. This means "distant" (in terms of mesh connectivity) parts of a protein will

only feel an interaction when brought close by.

- *The vdW force only takes place through the exterior*: As per points 1 and 2, the interior short range interactions have already been taken into account, and long range interactions are excluded. Therefore, the vdW force is limited to the surface of the proteins only, and takes place through the exterior medium. This allows us to exclude any pair of faces which are not both in front of and facing each other.

The above points allow most face pair interactions to be neglected, and thus preserve the computational efficiency of the algorithm.

3.4 Technical issues regarding Surface-Surface vs Element-Element

Given that the van der Waals force is a volume-volume interaction, it may at first seem counter intuitive to treat it as a surface interaction. However, a volume-volume interaction does not provide a convenient way of introducing localised hydrophobicity/hydrophilicity to the surface, since parameterisation would be on an element by element basis rather than face by face. To allow proteins to "stick" to each other would require additional surface-surface calculations. Furthermore, although a volume-volume interaction would allow for a softer interaction (as surfaces could overlap without consequence), the electrostatics solver described in chapter 6 would not converge in the case of overlapping proteins (since that would require multiple solutions for the potential at the same point in space).

More importantly, from an implementation point of view, the look-up table

and connectivity problem for an element which can have surface nodes on several sides of the same protein is extremely complicated and inefficient. To avoid the mechanical properties 'coupling' problem (discussed in the previous section) would involve excluding all pairs of elements sharing at least one node. Using face-face interactions allows us to use a simple look-up table. A cubic array of `Linked Lists` acts as a nearest neighbour look-up table to exclude distant pairs (see figure 3.4). Interactions are only included if the faces are sufficiently close and satisfy the following two criteria:

$$\vec{n}_1.\vec{n}_2 < 0 \tag{3.4}$$

and

$$(\vec{x} - \vec{c}).\vec{n}_1 > 0 \tag{3.5}$$

where $\vec{n}_1, \vec{n}_2$ are the normals of the two faces, $\vec{x}$ represents the three nodes of face 2, and $\vec{c}$ is a point on face 1. These two checks ensure that both faces are facing each other, and that no force is being transmitted through a protein interior.

It is physically more consistent to convert this into a surface interaction for the exterior, and leave the interior physics to be determined by the elasticity parameters (which are, effectively, already taking internal vdW into account).

The Surface-Surface approach avoids the problem of singularities by excluding any face pairs which share one or more mesh nodes. This is a relatively rapid search which can easily be precalculated in a lookup table, but most importantly it is not dependent on the mesh, providing the mesh is of good quality (broadly equliateral faces, and no sharp surface angles). The Element-Element case, on the other hand, must exclude elements linked via the interior of the mesh too,

which results in highly inconsistent pairings, in locational terms. FFEA meshes are mainly generated from surface data alone - the meshing algorithm gives no physical consideration to the internal mesh structure, but merely optimises it according to a user specified volume and element angle. Small changes to the surface mesh can produce profound changes in the internal mesh structure, and therefore completely change the internal dynamics of the protein when simulated. This inconsistency is even more problematic than the coupling problem between vdW and material parameters, discussed in the previous section.

3.5 Finite Element Formulation

Consider a protein whose surface is Γ_p. Now consider a point $\vec{p}$ on the surface Γ_p interacting with a point $\vec{q}$ on a full-system surface Γ_q with force density $\vec{f}(\vec{p},\vec{q})$, noting that $\Gamma_p \in \Gamma_q$ (see figure 3.2). $\vec{f}(\vec{p},\vec{q})$ is pairwise and acts along the line of separation (we can use a Lennard-Jones potential, for example). So the total surface force at point $\vec{p}$ is given by

$$\vec{F}(\vec{p}) = \int_{\Gamma_q} \vec{f}(\vec{p},\vec{q}) dA_q. \tag{3.6}$$

In order to include this force in the finite element calculation, we must put this in the appropriate weak form. The weak form of the FFEA momentum equation (equation (1.7)) is given by:

$$\int_\Omega w \left(\rho \frac{Du_i}{Dt} - \frac{\partial \sigma_{ij}}{\partial x_j} \right) dV = 0. \tag{3.7}$$

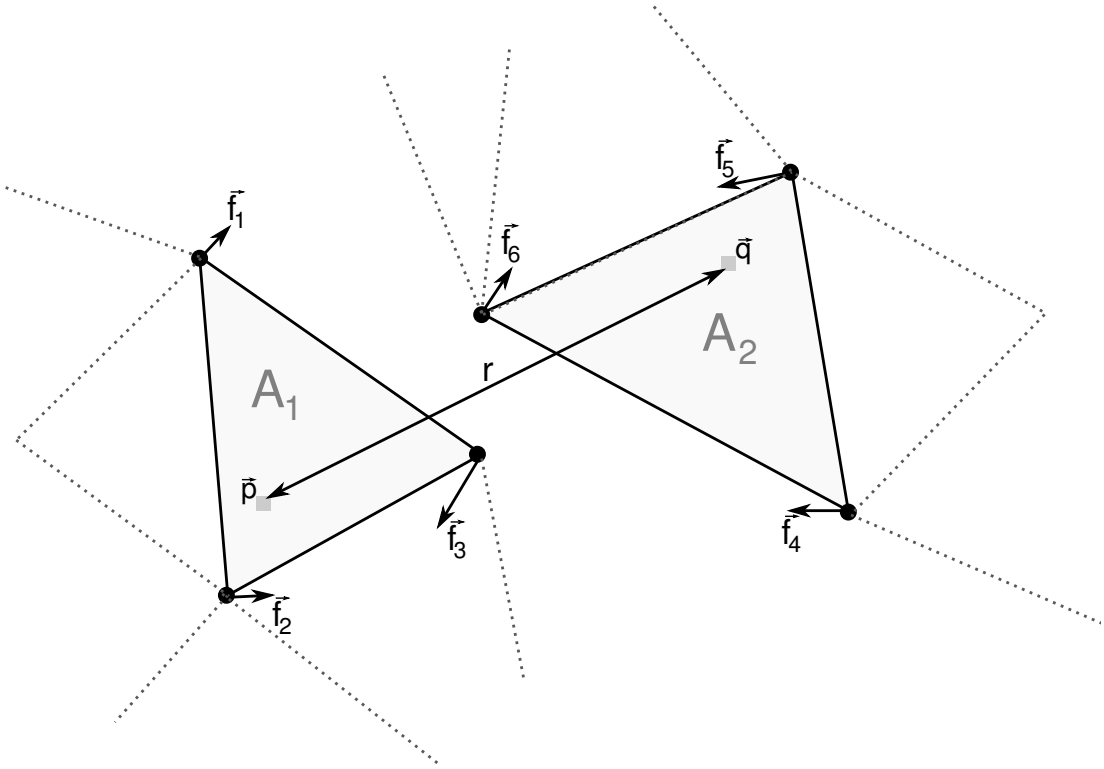


Figure 3.2: Steric repulsion and hydrophobic/hydrophilic solvent interactions modelled through a pairwise force per unit area. Shown here are two triangular faces interacting through such a force. The grey, dotted lines are intended to suggest the rest of the protein surface. By integrating the pairwise force $f(\vec{p}-\vec{q})$ over both areas, the force on each of the nodes can be calculated. Note that both faces may in fact belong to the same protein surface.

Integrating by parts this becomes:

$$\int_{\Omega} \rho \frac{Du_i}{Dt} w dV + \int_{\Omega} \sigma_{ij} \frac{\partial w}{\partial x_j} dV = \int_{\Gamma} w F_i dA \tag{3.8}$$

where $\vec{F}$ is the surface force density arising from vdW and σ_{ij} is the internal stress. Hence using equation (3.6) we have:

$$\int_{\Gamma_p} \omega(\vec{p}) F(\vec{p}) dA_p = \int_{\Gamma_p} \omega(\vec{p}) \int_{\Gamma_q} f(\vec{p}, \vec{q}) dA_q dA_p. \tag{3.9}$$

By decomposing $F(\vec{p})$ in terms of the surface restriction of the shape functions of the system

$$F(\vec{p}) = \sum_{i}^{N} \psi_i F_i \tag{3.10}$$

and choosing the weight function to be the shape function ψ_j, we arrive at the Galerkin form for the contribution of vdW forcing:

$$\int_{\Gamma_p} \psi_j \psi_i F_i dA_p = \int_{\Gamma_p} \psi_j \int_{\Gamma_q} f(\vec{p}, \vec{q}) dA_q dA_p. \tag{3.11}$$

The LHS is now essentially a surface mass matrix multiplying a vector of unknowns, so we can write this term as

$$M_{\Gamma}^{p} \vec{F}^{p} = \int_{\Gamma_p} \psi_j \int_{\Gamma_q} f(\vec{p}, \vec{q}) dA_q dA_p. \tag{3.12}$$

The integral on the RHS of equation (3.12) cannot be solved analytically (in general) so we use a Gaussian Quadrature scheme for triangles:

$$\int_{\triangle} g(\vec{x}) dA_{\triangle} \approx A_{\triangle} \sum_{k}^{N_{gp}} W_k g(\eta_1^k, \eta_2^k, \eta_3^k) \tag{3.13}$$

where the integral is approximated by a weighted sum of the integrand's values at the N_{gp} different Gauss Points. We then use this scheme with equation (3.12) to obtain:

$$\int_{\Gamma_p} \psi_j \int_{\Gamma_q} f(\vec{p}, \vec{q}) dA_q dA_p \approx \sum^{\triangle_p} \sum^{\triangle_q} A_p A_q \sum_{l}^{N_{gp}} \sum_{k}^{N_{gp}} W_l W_k \eta_j^k f(\vec{p}^k, \vec{q}^l) \tag{3.14}$$

where $\vec{p}^k = \eta_1^k \vec{n}_1 + \eta_2^k \vec{n}_2 + \eta_3^k \vec{n}_3$ and we have used the fact that, in this case, $\psi_j = \eta_j$. Calculating this quantity leads to the force per unit volume on the protein due to van der Waals interactions with all surfaces in the system. Some further considerations:

- The net interaction force of an element with itself is zero and can be ignored.
- Adjacent face pairs can in principle be safely ignored, as the angle between the faces necessary for the interaction to be significant is so small that it would indicate a badly formed mesh. Ignoring such faces avoids the need to deal with the singularities in the potential.

The implementation described above can also handle different types of surface interaction. As illustrated in figure 3.3, the different surface elements of the protein surface may be assigned a particular index representing their 'type'. A (symmetric) interaction matrix defines the LJ parameters for interaction between

each pair of face types. This allows the FFEA model to recreate, for example, the attraction between hydrophobic regions of a surface. Amino acids such as Valine, Cysteine and Leucine are strongly hydrophobic, and will therefore interact with different strengths than with surfaces composed of more weakly hydrophobic groups such as Glycine or Alanine[84]. Note that if the atomistic structure is known (and hence the amino acid side groups lying at the surface), then in principle a map of hydrophobic/hydrophilic regions (strong and weak) could be produced automatically.

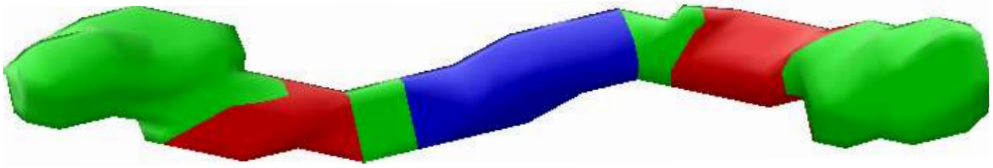


Figure 3.3: Example van der Waals patterning of the 3GHG human fibrinogen from figure 2.8 using the `FFEA_viewer` developed in section 2.3.5. The colour of the faces indicates their type (red, green, blue indicate types 1, 2 and 3 respectively). Any number of different face types is possible in general. Black faces (not shown on this figure) indicate non-interacting faces. See figures 4.4, 5.6 and 5.13 for examples of vdW patterning of FFEA systems.

3.6 Practical implementation

A naive implementation of the vdW scheme in section 3.5 would compute interactions between all face pairs in the system, resulting in a scaling of $\mathcal{O}(N^2)$ which rapidly becomes prohibitively expensive, particularly for higher order Gaussian quadrature schemes. Given that the long range part of the vdW interaction goes as r^{-6} and tails off after distances of a few r_{eq} (see figure 3.1), we introduce a cutoff distance ($4r_{eq}$, for the simulations presented in this thesis). Any faces

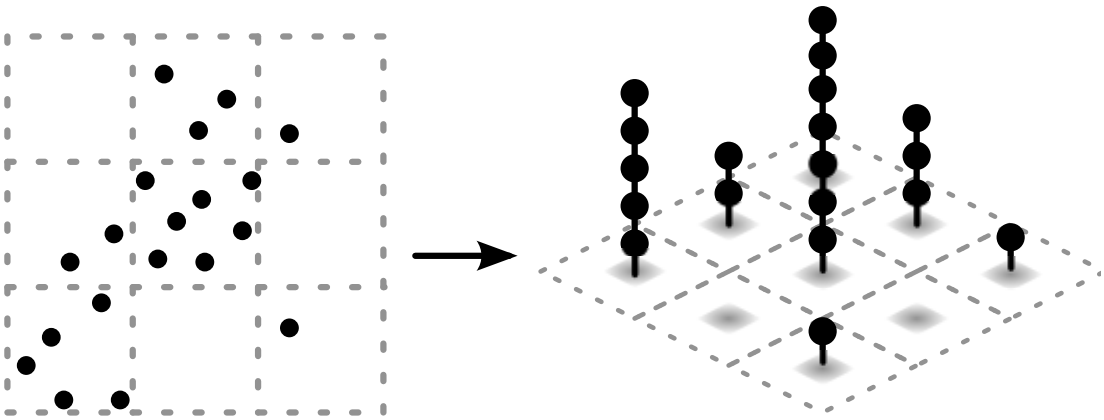


Figure 3.4: An illustration of a nearest neighbour look-up grid, shown here in 2-d for simplicity. The black circles represent BEM nodes (one at the centre of each surface element). The dotted grid shows the conceptual discretisation of the system. On the RHS of the figure a linked list stack has been constructed for each cell, containing the nodes within that area. In reality this grid is three dimensional and can be of any number of cells in length.

further apart than this distance will be assumed to have negligible interactions. This gives large improvements to the efficiency of the algorithm, but for large systems even the calculation of separation distance between face pairs can be expensive. To minimise this, a 3-d nearest neighbour lookup table is generated every few steps. The parameters defining the functioning of the lookup table (cell side length and update frequency) need to be carefully set based on the maximum size of surface faces and the maximum drift speed of proteins through the system (see section 3.7.3).

It is also necessary to choose an appropriate number of Gauss points for the face size. Interacting faces in the simulations presented in this thesis were generally no larger than 100Å^2, for which a 3-point Gaussian quadrature scheme was adequate. A spurious unphysical torque can sometimes be observed[1] in systems

[1]For systems with no external damping.

for which the number of Gauss points is too low for the face size, particularly when there is a large difference in size between the interacting faces.

The efficacy of the scheme was tested by running simulations of two interacting elements, two cubes, multiple spheres and, finally, a large number of irregularly shaped meshes. The latter were constructed from the atomistic structure of myoglobin (shown in figure 3.5).

3.7 Test system: Myoglobins interacting with an attractive substrate

3.7.1 Introduction

As a test system for the stability and performance of the vdW implementation on the arc1 supercomputer, FFEA was used to model a crowded system of myoglobin molecules interacting with a hydrophobic surface (non-oxidized polystyrene). This was inspired by the work of Muntean *et al.* in which they use MD to investigate the adsorption of a single myoglobin molecule to a polystyrene surface[85]. The adsorption of proteins onto surfaces is important for the creation of biosensors. Due to the performance constraints of all-atom MD (discussed in chapter 1) it is not currently feasible to simulate the entire adsorption process, even for a single molecule, and therefore separate simulations were carried out for the molecule placed at different distances from the substrate. Most of the runs were of 1 ns in length, although some longer runs at 15 ns were performed to check agreement between the time scales[85].

The coarse-grained nature of FFEA allows it to simulate not only the full

adsorption trajectory of a single molecule, but equally the effects of crowding from a large number of such molecules interacting. Myoglobin is around 4.7 × 3.4 × 3.9 nm in size, and therefore towards the lower end of the scale applicable for FFEA. Nevertheless, experimental evidence that folded proteins can behave as viscoelastic continua[45] suggests this simple vdW implementation test case could yield results of biophysical interest. However, the aim here is to assess the code as applied to a 'real' system, and as this is one of the types of mesoscale system FFEA was designed to address, it was decided it would be an appropriate test system for the vdW interactions.

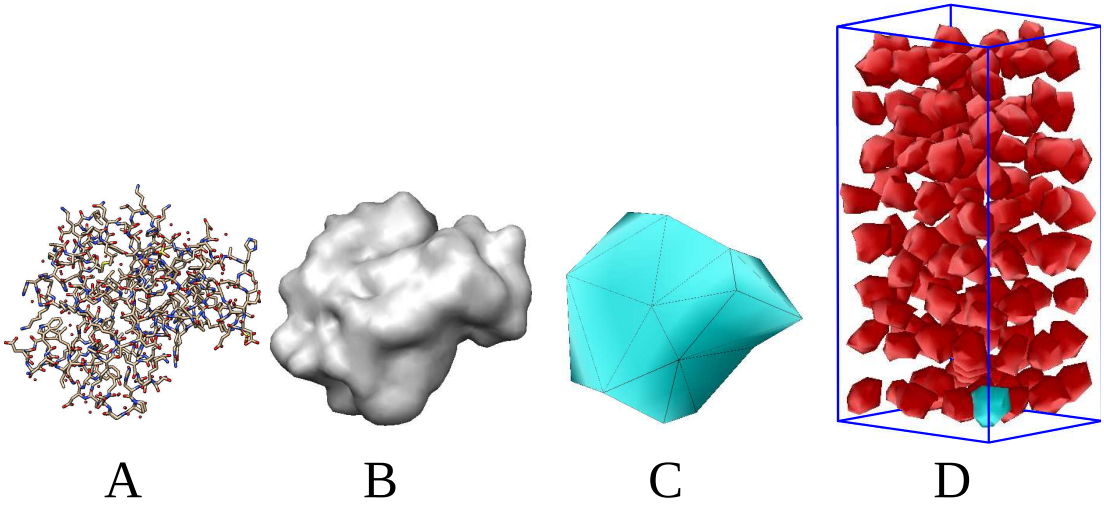


Figure 3.5: Transformation from the atomistic structure of myoglobin (**A**) to a density map (**B**) to a (very crude) FFEA representation (**C**). **D** shows the simulation box setup as described in section 3.7.2. The simulation box is shown in dark blue. Myoglobins are shown in red apart from the protein being tracked (light blue, interacting with surface).

3.7.2 Method

128 interacting myoglobins were simulated interacting with attractive surfaces on the top and bottom x-z planes, and periodic boundary conditions on the other four faces. The proteins were initially tiled in a cubic lattice $4\times8\times4$ resulting in a simulation box of dimensions $21\times42\times18$ nm with a lookup cell side length of 3 nm (dimensions $7\times14\times6$). The equilibrium separation distance in the LJ potential, r_{eq} = 1nm for proteins interacting with the surface. The interaction energy between myoglobins was chosen to be 10^{12}Jm^{-4} with an equilibrium separation of 0.5 nm. The rest state volume of the myoglobin mesh is around 22.7nm^3, giving a volume fraction of around 20% (note that the concentration will effectively be higher due to the LJ equilibrium separation distance around each myoglobin). A protein was determined to be 'on' the surface when the centroid of its surface face nearest either of the attractive surfaces was less than 3 nm from the surface. The time evolution of the on-off ratio of a single molecule was calculated for 10 values of Young's modulus (ranging from 40 to 130 MPa), and 7 values of Poisson ratio (0.3 to 0.42), thus 70 simulations were run per surface interaction energy. The simulations were run until convergence of the on-off ratio. This typically required simulating a few μs of dynamics. Depending on the stiffness of the elastic properties, the contact area the myoglobins made with the polystyrene surface was typically 10 nm^2 (but almost 14 nm^2 for low Young's modulus). The energy of interaction between a myoglobin and the surface was chosen as 10^{13}Jm^{-4}, resulting in interaction energies of order k_BT or more (broadly similar to that in reference [85]).

3.7.3 Observations

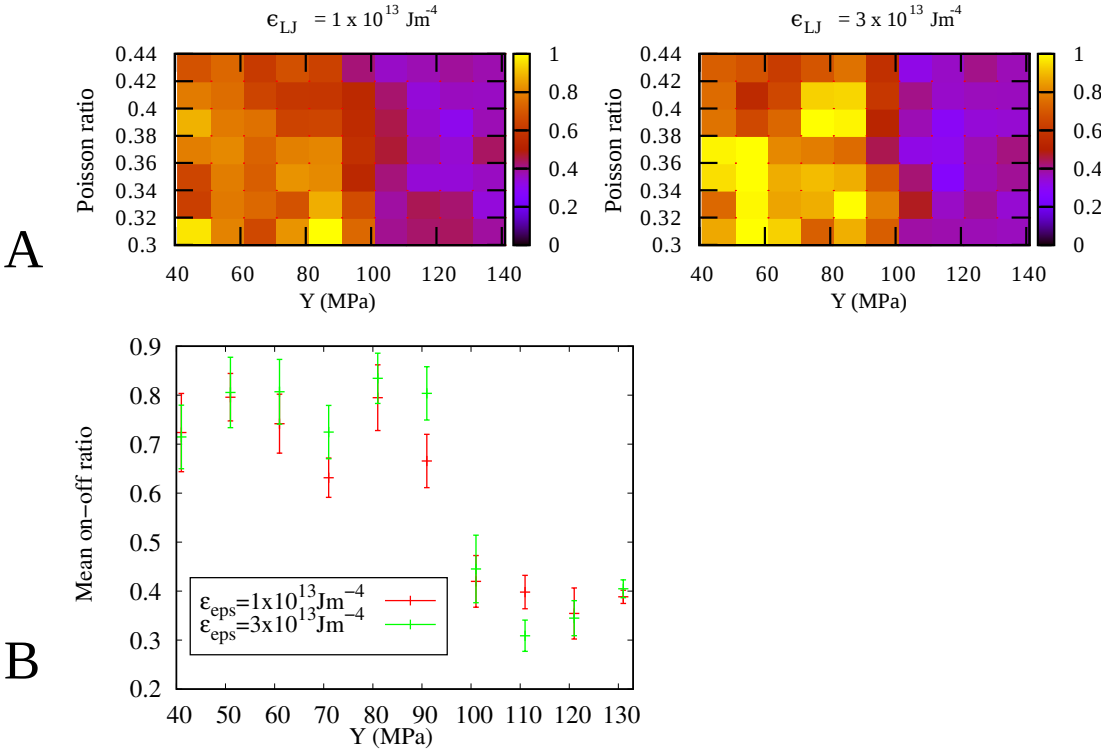


Figure 3.6: **A**: Graphs of on-off sticking ratios for myoglobins on an LJ surface for different energy well depth, ϵ_{LJ}. There are 70 simulations for each interaction energy (10 values of Young's modulus and 7 values of Poisson ratio.) Poisson ratio appears to have little effect on the sticking ratios. **B**: A plot of the on-off ratio at each value of Young's modulus (averaged over the 7 Poisson ratio simulations) with standard error. This makes the sharp transition between high and low sticking rates around 100 Mpa clearer.

The results of the on-off ratios calculated in the simulations described in section 3.7.2 are given in figure 3.6. In physical terms, we observe several interesting features. First that there appear to be two distinct regimes (high ratio and low ratio) with a relatively abrupt transition at $Y = 100$ MPa. This change is apparent in figure 3.6 despite the statistical noise (due to only one run per set of parameters). The softer proteins can deform to provide a greater area of contact,

allowing them to stick to the surface strongly. Second, the Poisson ratio appears to have a negligible effect on the sticking ratio. The weak effect of the Poisson ratio on dynamics is a recurring observation, also seen for the systems studied in chapters 4 and 5 of this thesis. Finally, despite a comparatively low energy of sticking (around $1k_BT$) some proteins achieved very high on-off ratios, particularly for the $3 \times 10^{13}\mathrm{Jm}^{-4}$ simulations. Qualitative observations suggest that the crowding near the surface may obstruct a protein's attempt to diffuse away, thus increasing the probability that it will be re-adsorbed to the surface.

Despite these general observations, the aim of the simulations was also to assess the functioning of the vdW implementation in a more realistic system. The scheme was found to produce stable simulations for the simulated μs for all chosen elastic parameters and interaction energies. It was found adequate to rebuild the lookup table only every 10 simulation steps as the diffusion rate of the myoglobins was sufficiently small that faces were not traversing the lookup grid too quickly.

3.8 Conclusions

The 'van der Waals' interaction is widely used in molecular simulations to simulate the effects of steric repulsion and attractive dispersion forces. In this chapter a method of incorporating van der Waals forces into the FFEA model was presented which allows for different types of surface interaction (e.g. hydrophobicity). A surface-surface method (rather than volume-volume) was chosen to prevent undesirable coupling of material properties in the protein interior (in which vdW forces are already effectively accounted for by the viscous and elastic parame-

ters) and to resolve inconsistencies in self-interaction of elements arising from complicated topologies. The interaction was implemented as a Lennard-Jones force integrated over the surface. Performance improvements were introduced via a nearest neighbour lookup grid. The efficacy of the scheme was investigated through application to a 'real' system of 128 myoglobin molecules interacting with an oxidised polystyrene substrate via a hydrophobic attraction.

Chapter 4

ATPase

The work presented in this chapter emerged from a collaboration with K. Papachristos, D. J. Read, O. G. Harlen, M. Harrison, E. Paci, S. P. Muench, and S. A. Harris, and is published in reference [86] (see page i for attribution details).

4.1 Introduction

4.1.1 The rotary ATPase family

The rotary ATPase family is a set of proteins which exist within biological membranes. They are highly efficient energy-conversion machines: they can either use a gradient of protons across the membrane to fuel the production of ATP or, conversely, use ATP to generate a proton gradient[87]. This is achieved through the coupling of two distinct motors; an ATP binding domain, and a membrane bound proton pump. The rotary ATPase family is composed of three sub families: F-ATPases, V-ATPases and A-ATPases. The $F_1F_\circ$-ATPases (F-ATPase) are mainly responsible for synthesising ATP using a proton gradient across the membrane.

The vacuolar H+-ATPases (V-ATPase), on the other hand, use ATP hydrolysis to drive proton transport across the membrane (see figure 4.1A). Contrastingly, the third family, archaeal A-ATPase (figure 4.1B), is capable of working in either direction, producing ATP synthesis or proton transport according to the potential across the membrane. These three cases all involve rotation of the rotor and *c*-ring as part of the mechanism.

In the V- and A- type ATPases, the soluble domain (V_1 and A_1 respectively) consumes ATP, driving the mechanism that pumps protons across the membrane. The three AB dimers that make up the soluble domain form a three step motor, cycling in sequence through three states: open (no nucleotide bound), loose (ADP + Pi bound) and tight (ATP hydrolysing)[88]. This cycle induces conformational changes in the AB dimers, producing a torque on the central rotor axle and consequently driving rotation of the *c*-ring.

4.1.2 The apparent stator-rotor connection

The *a*-subunit and AB domains (responsible for ATP hydrolysis) are held relative to the central rotor and *c*-ring (responsible for proton pumping) via coiled-coil structures known as *stators* (figure 4.1). There are three stators connecting the motors in the V-ATPase, two stators connecting the motors in the A-ATPase, and one stator connecting the F_1 and $F_\circ$ domains in the F-ATPase. There is also an apparent second rotor/stator connection observable in the complete V- and A-ATPase structures, linking the EG stator to either the *d* subunit of the central axle, or the *c*-ring. A connection at this point prevents the *c*-ring from rotating freely against subunit *a* during hydrolysis of ATP. Figures 4.1C-E show

the effects of varying the contour level to the point at which this connection disappears, showing that other (known) features of the motor disappear first. This provides evidence that the connection is real, rather than an experimental artifact.

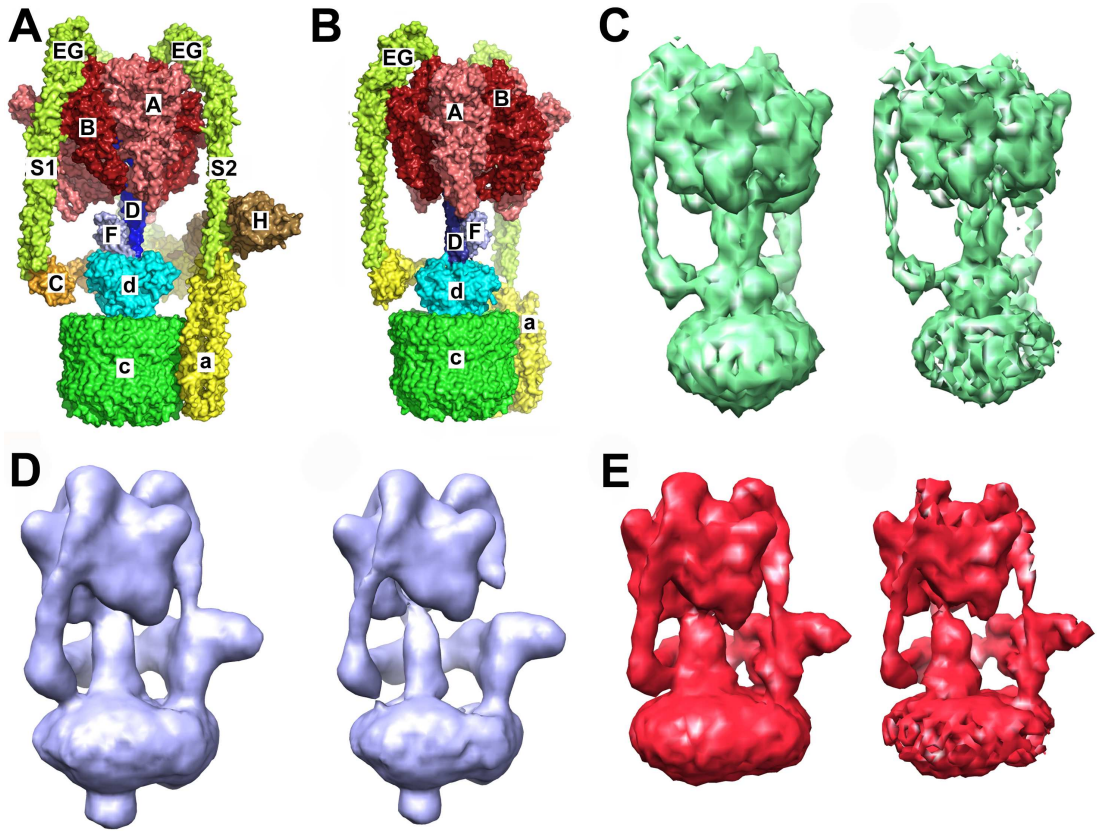


Figure 4.1: Subunit fitting for the V-ATPase (**A**) and A-ATPase (**B**) complex with those subunits involved in the ATPase motor domain (A/B), stator (E/G/C/H), rotor (D/F/*d*/*a*) and *c*-ring labelled. Single particle cryo-EM reconstruction of the *T. thermophilus* A-ATPase (**C**) *M. sexta* (**D**) and yeast (**E**) V-ATPase contoured at the recommended level in the EMDB (left) and at a level where the apparent link between stator and rotor axle is removed (right).

4.1.3 Elasticity studies in the V- and A-ATPases

There is substantial structural conservation between corresponding subunits in the V- and A-ATPase. However, some important differences exist, notably in the size of the *c*-ring. As the *c*-ring is not commonly formed from a multiple of 3 subunits, there is a symmetry mismatch with the 3 step ATP hydrolysis/synthesis domain[89; 90; 91]. As ATP hydrolysis is the rate limiting step, this symmetry mismatch cannot be overcome by applying constant torque to the central rotor to generate continuous rotation of the *c*-ring. It has therefore been suggested that there may be an elastic connection between the ATP hydrolysing domain and the *c*-ring acting as an energy buffering device[92; 93; 94; 95; 96]. This elasticity is proposed to increase motor efficiency by minimising the free energy cost of the ATP induced conformational changes[97]. Possible candidates for this elastic energy buffer are the stators or the central rotor axle (or both). Indeed, such flexibility has been observed in crystal structures and electron microscopy data[98; 99; 100].

Elastic Network Models (described in section 1.1.3) have also been used to study questions of elasticity in the rotary ATPases. Normal Mode Analysis on the coiled-coil dimeric protein that forms the A-ATPase stator stalk revealed flexing in a radial direction [101]. More recently, Song *et al.* have shown that both the *Saccharomyces cerevisiae* and *Manduca sexta* V-ATPases are capable of bending along their long axis such that the V_1 domain is displaced by up to 10° relative to the V_o domain[102].

4.1.4 Aims

The aim of this chapter is to apply the FFEA model to a real biological system, in this case the V- and A-ATPase rotary proteins (F-ATPases are too small for treatment with FFEA), and to show where FFEA can exceed the scope of the Elastic Network Model, with which it shares several conceptual similarities. As we seek to compare the results of a typical FFEA approach with those of a typical ENM approach, we initially limit ourselves to a system in which both ENM and FFEA are applicable. Having compared the two models, FFEA is then used to investigate differences in the dynamics of the rotary ATPase family in the presence and absence of the apparent stator-rotor connection, as well as the effects of stator number (2 or 3) on the principal modes of motion. Finally, the results of an FFEA simulation of the dissociation of the $V_\circ$ and V_1 domains is presented as an example of a simulation that could not be carried out with standard ENM implementations.

4.2 Methods

4.2.1 Meshing

The *Manduca sexta* (*M. sexta*) V-ATPase (EMD-1590 at 17 Å resolution), *Saccharomyces cerevisiae* (*S. cerevisiae*) V-ATPase (EMD-5476 at 11 Å resolution) and the *Thermus thermophilus* (*T. thermophilus*) A-ATPase (EMD-5335 at 9.7 Å resolution) maps were obtained from the EMDB database[103]. Surface meshes were calculated from the maps using the marching cubes algorithm at the recommended contour levels, and 'cleaned' and coarse grained using the *decimate*

and FPC procedures described in section 2.3.2 of this thesis. The finite element meshes were then created from these surface meshes using NETGEN [104] as illustrated in figure 4.2.

4.2.2 FFEA simulations of base structures

4 μs FFEA simulations of these base structures were run using the simulation parameters listed in table 4.1. The typical material parameters for a protein were chosen (as described in section 1.2.3): The Young's modulus was taken from the range for lysozyme[53] and the viscous parameters were taken to be the same as for water. As for all the FFEA simulations in this thesis, the density parameter was chosen to be the average density of biomolecules.

As the aim of this chapter is to compare modes of motion, the precise choice of material parameters is not crucial. The density and viscous parameters affect the time scale of the simulation, and the manner in which the model explores conformational space, but they do not affect the range of configurations available to the protein (which corresponds to the modes). Similarly, varying the Young's modulus will change the mode amplitudes, but not the resultant eigenvectors themselves (except where the deformation is large and non-linear). To test this, simulations with a 20% difference in Young's modulus were analysed for the *M. sexta* and *S. saccharomyces* V-ATPases. The results, given later in figure 4.15, show the eigenvectors to be largely unchanged. The Poisson ratio also has only a weak effect on the modes. As with ENM, it is the shape and topology of the protein that determines the modes of motion in FFEA.

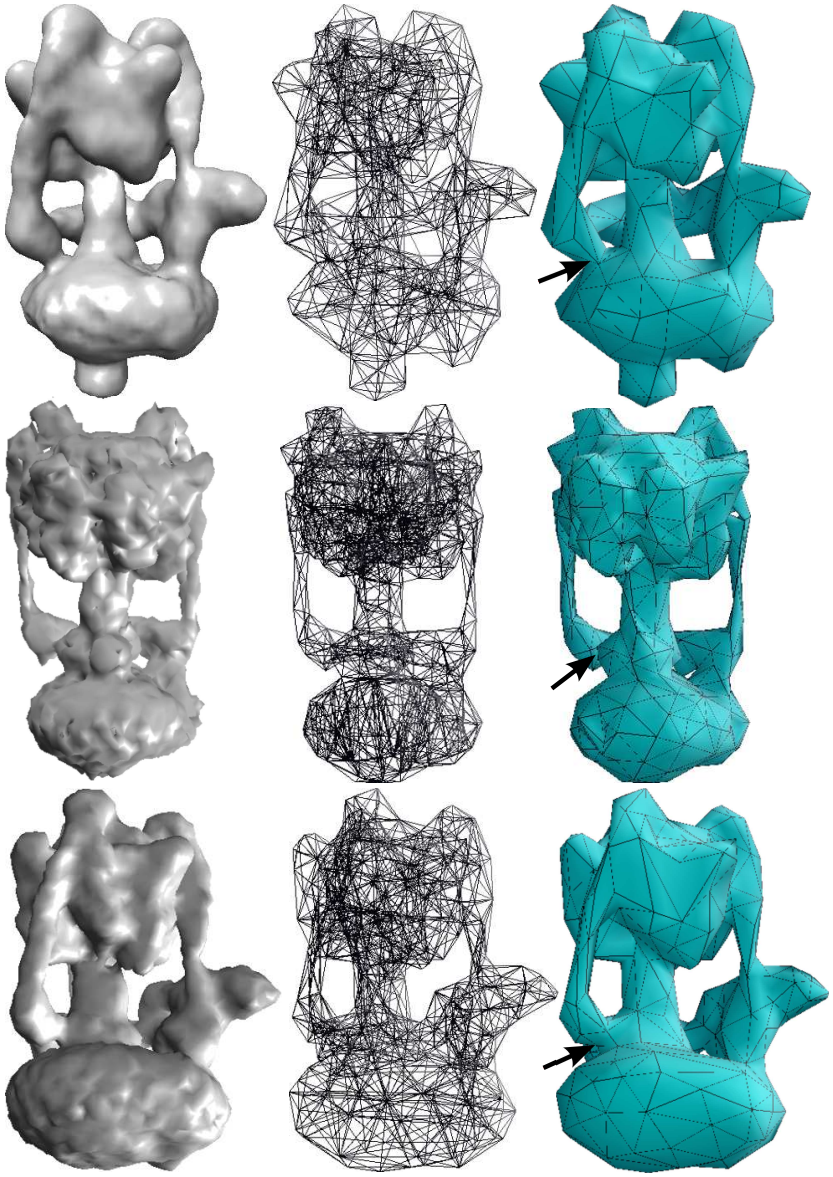


Figure 4.2: The single particle cryo-EM map (left) vs FFEA models (centre and right) for the M. sexta V-ATPase (**top**), the T. thermophilus A-ATPase (**middle**) and the *S. cerevisiae* V-ATPase (**bottom**). The location of the apparent connection between the stator network and rotor axle is indicated on each FFEA representation via an arrow.

Density	1500 kg m^{-3}
Young's Modulus	338.8 MPa
Poisson's Ratio	0.41
Shear Viscosity	1 mPa.s
Bulk Viscosity	1 mPa.s
Time step	5 fs
Temperature	290 K

Table 4.1: Simulation parameters for FFEA calculations.

4.2.3 FFEA simulation of partially severed structures

The apparent connections between the stator and rotor (shown in figure 4.2) in the above base structures were severed by removing a small amount of matter from the connecting area. The simulations were then repeated using the same material parameters with these topologically different meshes.

4.2.4 Comparison with elastic network models

The ENM normal-mode analysis calculations were carried out by K. Papachristos. The *M. sexta* V-ATPase and *T. thermophilus* A-ATPase were coarse-grained into a total of 250 beads, and the *S. cerevisiae* V-ATPase into 256 beads, using a topology-preserving algorithm[105] to reproduce the overall shape and topology of the complexes. The beads were connected by springs with spring constants derived by Stember and Wriggers[106]. The eigenvectors were calculated using MODEHUNTER[106].

Principal Component Analysis [107] was used to extract the quasi-harmonic normal modes from the FFEA trajectories. The first six trivial modes, which describe the overall translation and rotation of the system, are removed prior to the PCA. In order to compare the ENM modes with the FFEA modes, it

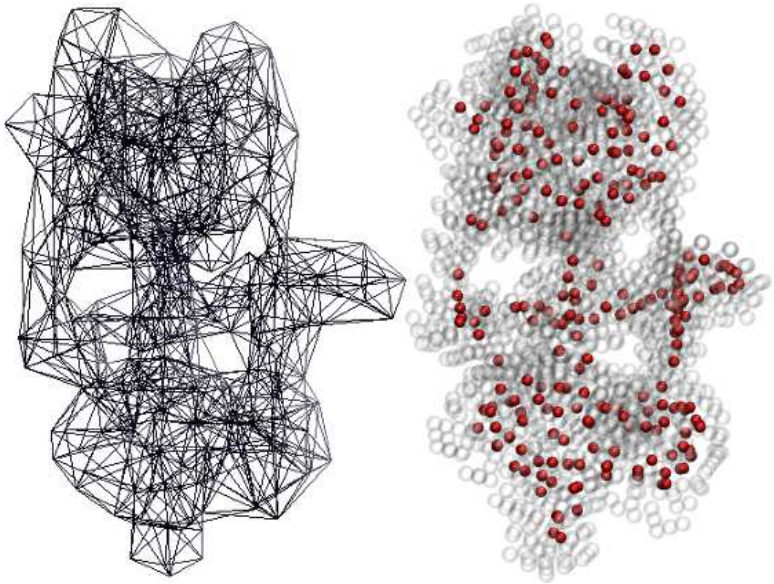


Figure 4.3: Continuum model showing the FE mesh (left) and embedding the ENM pseudo-particle structure (red) inside the FFEA continuum mesh (grey) represented here by the nodes of the finite elements (right).

was necessary to obtain eigenvectors of the same dimension. To achieve this, the ENM structure was aligned within the finite element mesh by minimising the square of the total pair-pair separation distance between all ENM pseudo-particles and FFEA nodes. A Monte Carlo approach to minimisation was applied, sampling many random translations and rotations of the ENM structure. The probability of accepting a new configuration is given by the Boltzmann probability $P = \exp(\frac{-dE}{kT})$ where dE represents the change in the "energy penalty" of the configuration (the total pair-pair distance squared) resulting from that step, and T is a scaling energy analogous to thermal energy in a real system. The purpose of this artificial "temperature" is to allow the structure to sample the whole configurational space and find the global minimum. T was slowly reduced, allowing the ENM structure to settle with optimal alignment in the FFEA structure.

Each bead of the ENM structure was then assigned to its local (containing) element in the continuum mesh, as shown in figure 4.3. The barycentric coordinates (equivalent to the linear shape functions) of the particle within that local tetrahedron were calculated, and could then be used to map the new position of the particle in all subsequent frames. This allows the motions of the FFEA derived modes to be mapped onto the ENM structure, allowing a direct comparison to be made between the two models (as both eigenvectors now have the dimensions of the ENM structure).

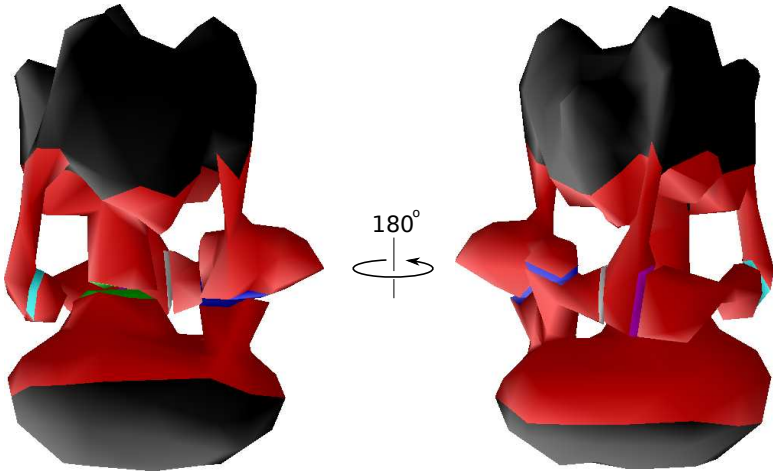


Figure 4.4: The vdW surface patterning of the *S. cerevisiae*, shown from the 'front' (left) and 'back' (right). All non-black faces are interacting. Red faces interact with all non-black faces through a hard repulsion (negligible attraction $\epsilon_{LJ} = 10^{12}\text{Jm}^{-4}$). All other non-black faces interact strongly ($\epsilon_{LJ} = 10^{15}\text{Jm}^{-4}$) with faces of matching colour, and through hard repulsion otherwise. This is to prevent, say, the severed axle attempting to bind strongly with the C subunit.

4.2.5 FFEA simulation of fully severed (dissociating) structures

For the purposes of comparison with ENM, the above simulations intentionally limited FFEA and its analysis to systems treatable with ENM. We now apply FFEA to the simulation of the dissociation of the $V_\circ$ and V_1 domains, a dynamical process for which ENM is not applicable. The *S. cerevisiae* was severed along the interface of its $V_\circ$ (*c*, *a* and *d*) and V_1 (motor domain, D, H and stators) domains, and the C subunit. This disconnection was performed as described in section 4.2.3, however the interface at each cut was now modelled as two interacting vdW surfaces. These three now distinct parts were then allowed to interact via a vdW surface interaction (as described in chapter 3), until dissociation. As the $V_\circ$ domain is embedded in the membrane, its lower half was pinned in place by pinning (immobilising) the corresponding nodes. The exterior solvent viscosity was chosen to be 10^{-6} Pa.s to increase the rate of exploration of conformational space. The LJ equilibrium separation distance was taken to be 5Å (as around 5 Å of matter was removed), and the surface interaction energy to be 10^{15} Jm^{-4} [1]. The separation distance between the centres of mass of the $V_\circ$ and V_1 domains with time was then calculated.

[1]For comparison, consider that an interaction energy of 10^{15} Jm^{-4} with an interacting surface area of 2 nm^2 corresponds to a total interaction energy of $1k_BT$.

4.3 Results

4.3.1 Comparison of FFEA and ENM dynamics

The quasi-harmonic modes extracted from the 4μs FFEA simulations, and the normal modes for the same complexes calculated using an ENM approach are represented in figures 4.5 and 4.6 respectively. In both models the first two modes corresponded to twisting and bending.

A quantitative comparison of the ENM and FFEA model was obtained by calculating the dot products of each eigenvector obtained from the ENM with each eigenvector extracted from the FFEA trajectories using PCA, as shown in figure 4.7. The differences in shape of the ATPase models in figure 4.5 and figure 4.6 are due to the fact that the FFEA mesh contained more nodes than the number of pseudo-particles in the ENM, and this is accounted for by the alignment procedure in section 4.2.4. Nevertheless, figure 4.7 shows that there is a clustering of high correlation between modes close to the diagonal, indicating that similar modes of flexibility of the ATPases are predicted by both the ENM and FFEA calculations. Although the agreement between the two modelling methods is reduced for the higher order modes in each of the three ATPases, these modes have far smaller amplitudes (figure 4.11) suggesting that the major modes of flexibility captured by the two methods are comparable.

Differences in the ordering of these modes are due to the contrasting treatment of local elasticity by ENM and FFEA. The characteristic "springs" of the BTS ENM are one-dimensional objects and, in isolation, exhibit little torsional resistance, whereas the volume elements of FFEA strongly resist torsion as well as extension. This is particularly evident in the central stalk region of the simu-

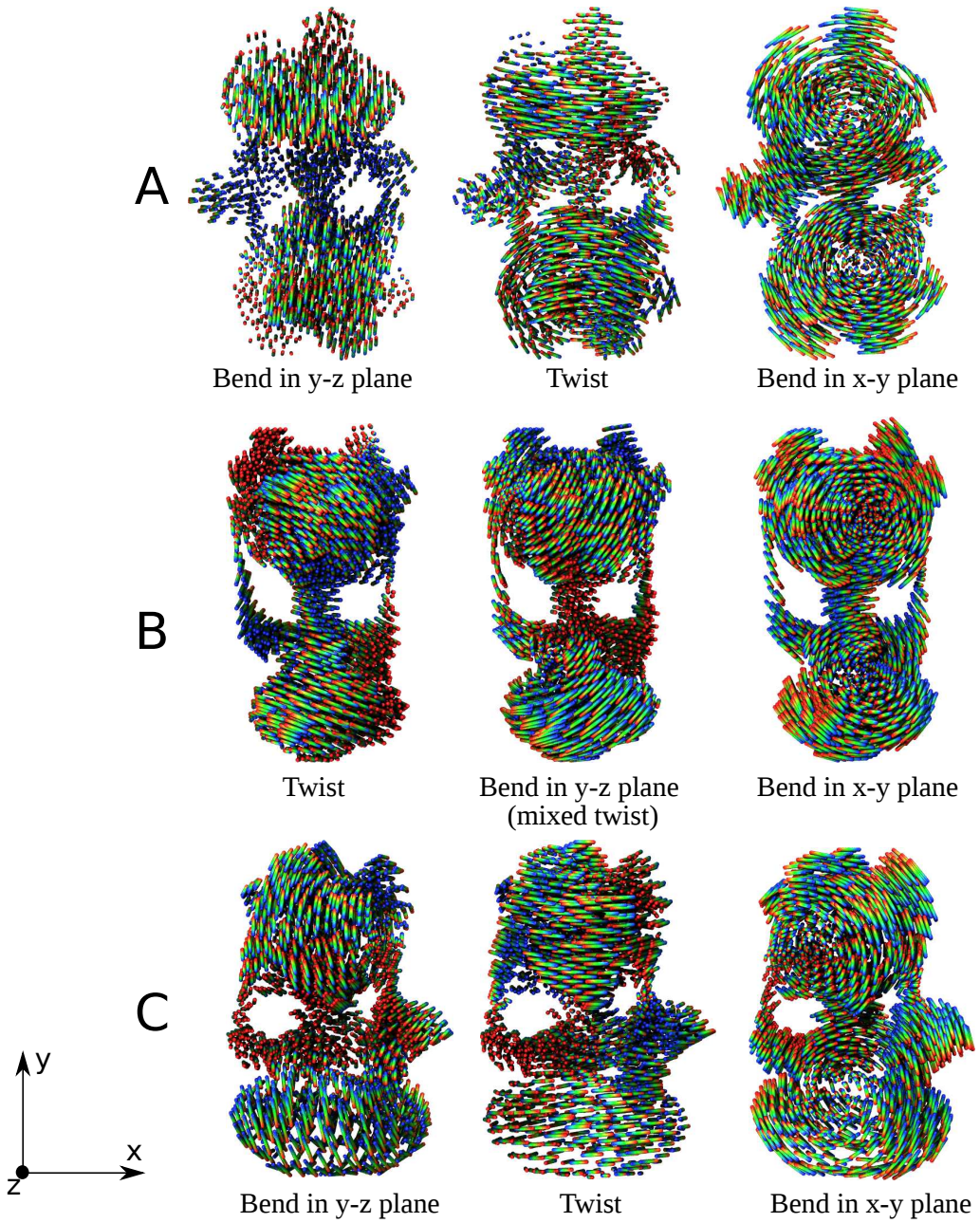


Figure 4.5: First three modes of the FFEA model for the M. sexta V-ATPase (row **A**), the *T. thermophilus* A-ATPase (row **B**) and the Saccharomyces V-ATPase (row **C**). Colours represent time range of motion, with red indicating the start of the motion and blue indicating the end.

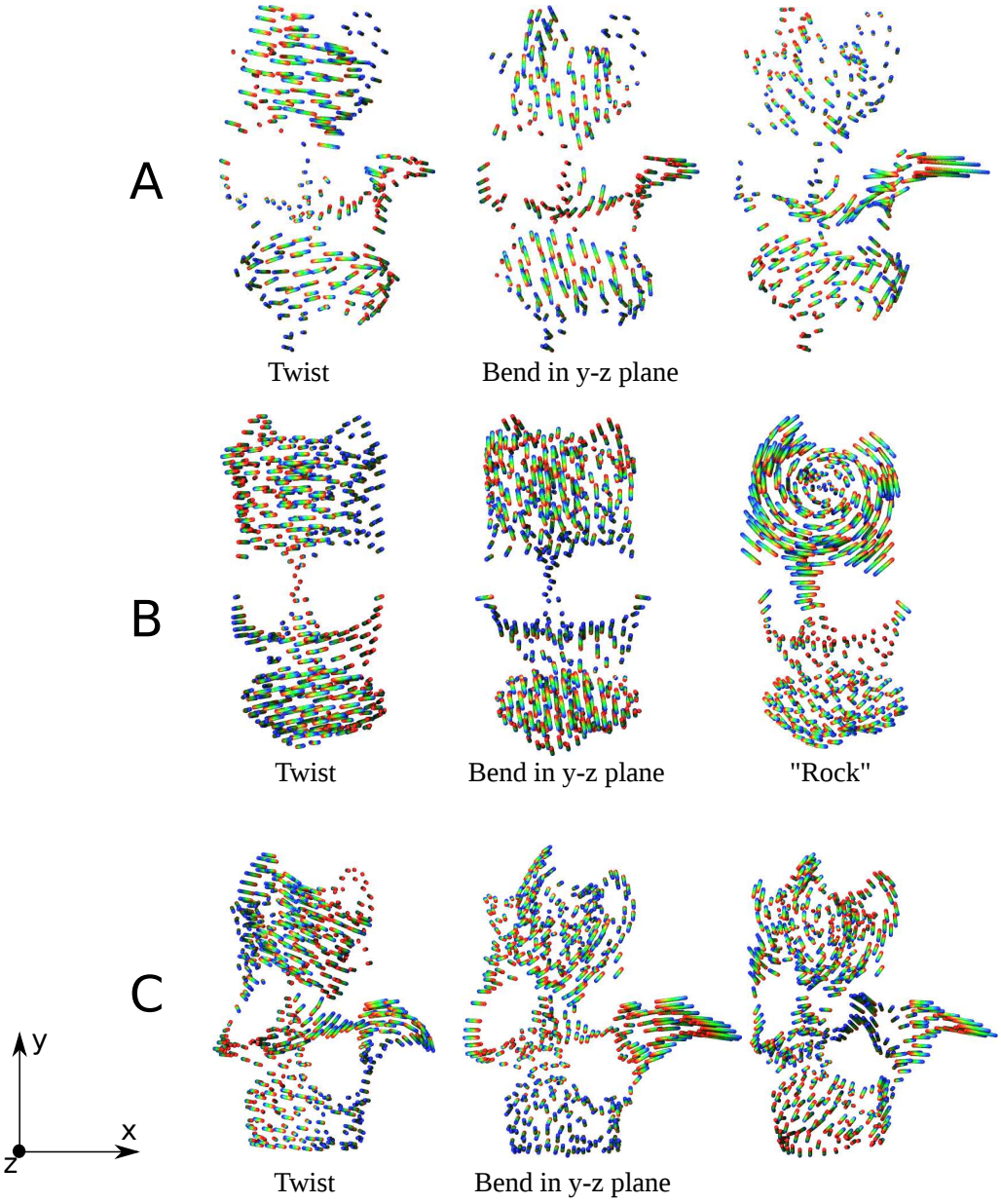


Figure 4.6: First three modes of the ENM model for the M. sexta V-ATPase (row **A**), the *T. thermophilus* A-ATPase (row **B**) and the *S. cerevisiae* V-ATPase (row **C**). Colours represent time range of motion, with red indicating the start of the motion and blue indicating the end.

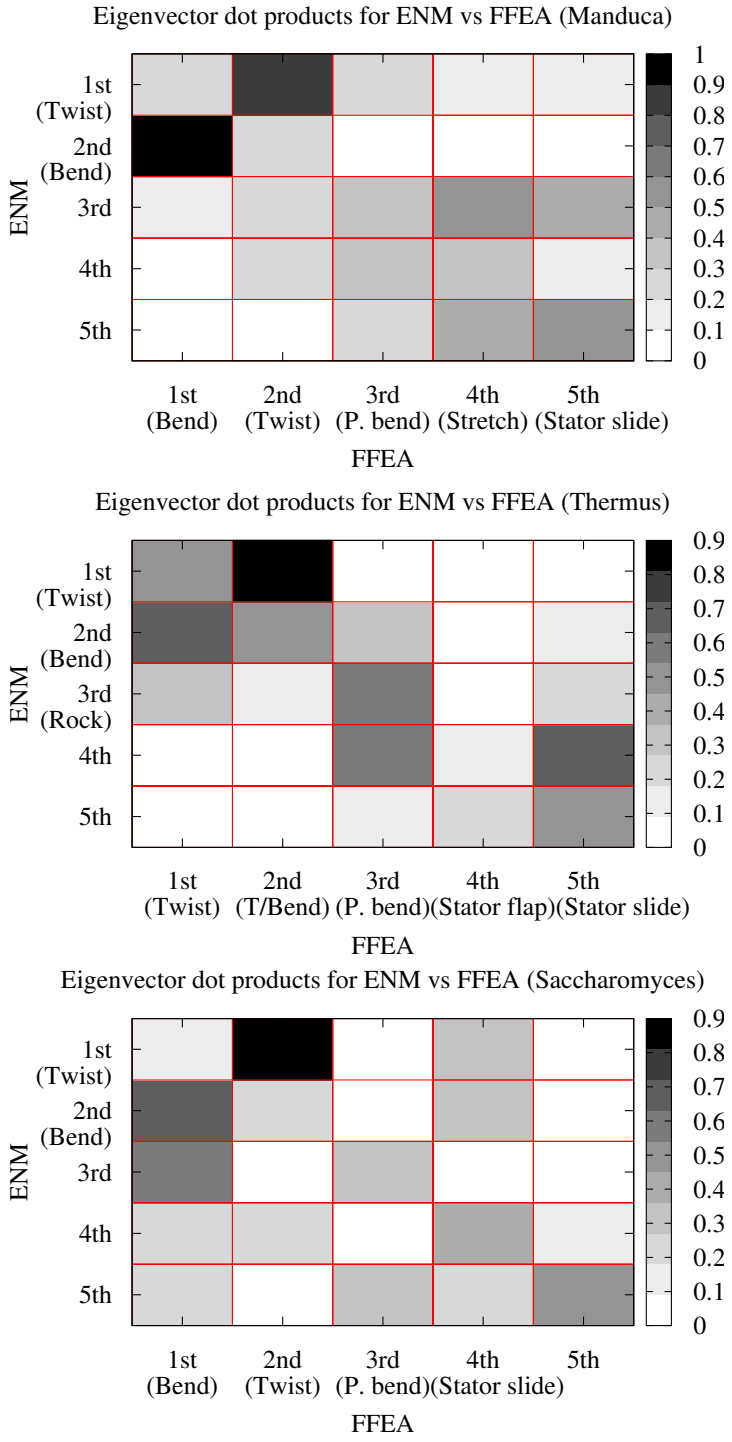


Figure 4.7: Comparison of the ENM modes with the FFEA modes for the M. sexta V-ATPase, *T. thermophilus* A-ATPase and *S. cerevisiae* V-ATPase. The shading scale runs from white (dot product of eigenvectors yields 0) signifying total disagreement, to black (dot product yields 1 or 0.9) signifying perfect agreement.

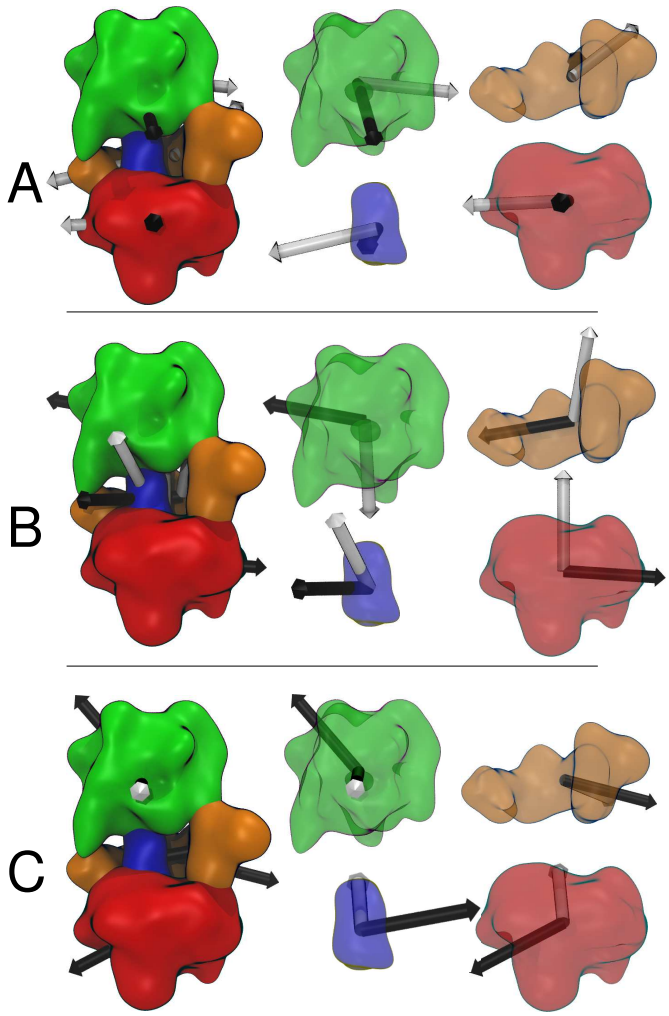


Figure 4.8: The mobility of four sections of the FFEA representation of M. sexta V-ATPase, divided by motor domain (green), rotor (blue), *c*-ring (red) and the remainder of the C, H and *a* subunits (orange), for the first three modes (**A**, **B** and **C**, respectively). The white arrow shows the normalised rotational velocity vector of that section, and the black arrow shows the motion of the centre of mass during the motion.

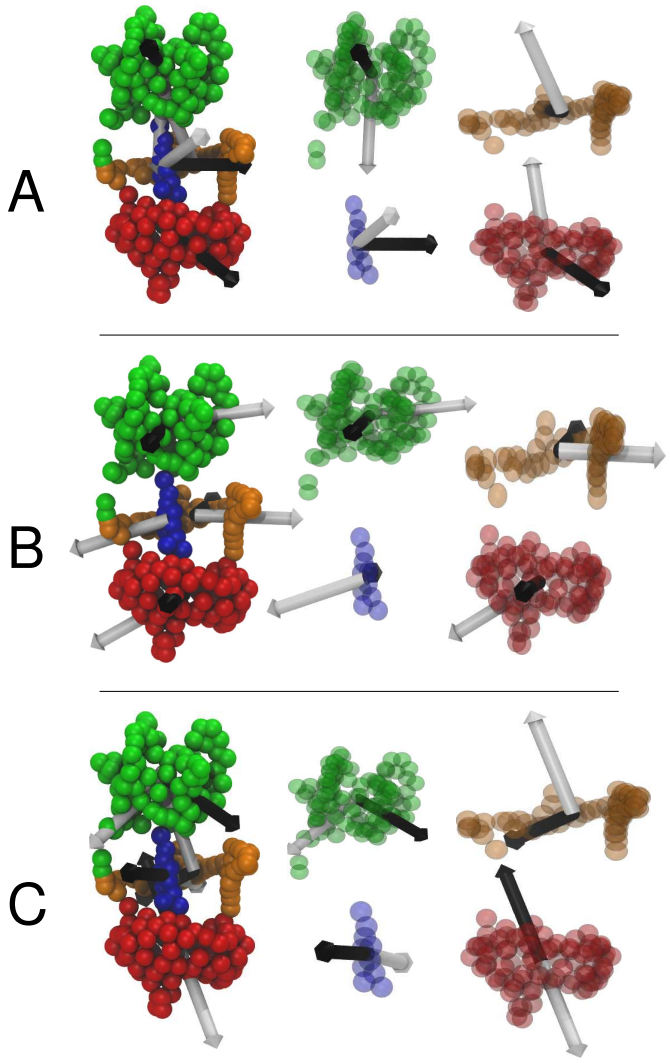


Figure 4.9: The mobility of four sections of the ENM representation of M. sexta V-ATPase, divided by motor domain (green), rotor (blue), *c*-ring (red) and the remainder of the C, H and *a* subunits (orange), for the first three modes (**A**, **B** and **C**, respectively). The white arrow shows the normalised rotational velocity vector of that section, and the black arrow shows the motion of the centre of mass during the motion.

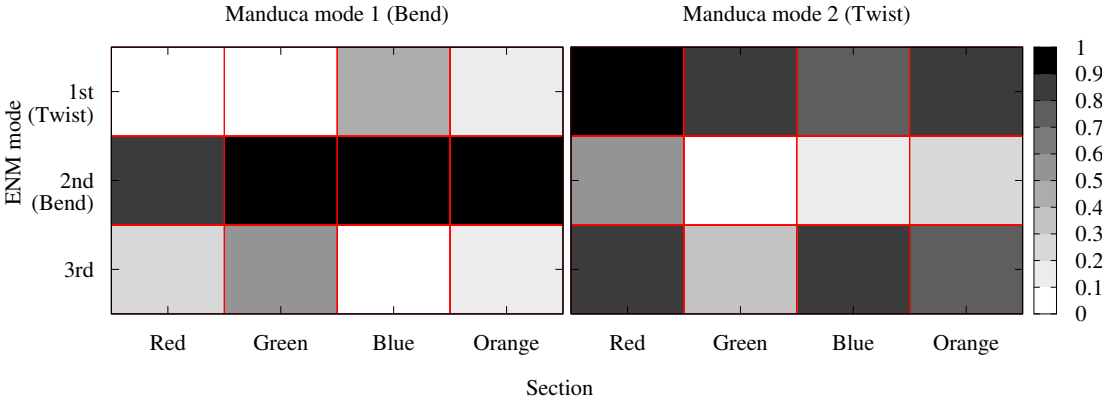


Figure 4.10: A comparison of the agreement in mobility profile of the four motor sections (as shown in figure 4.8 and figure 4.9) for the first two FFEA modes and first three ENM modes of M. sexta V-ATPase. The shading scale runs from white (dot product of eigenvectors yields 0) signifying total disagreement, to black (dot product yields 1) signifying perfect agreement.

lations, which contains only a few elements or springs; in this region the FFEA simulations are more resistant to torsion than the corresponding ENM simulation. As a result, in the *M. sexta* V-ATPase, the bending mode dominates in the FFEA, whereas the twisting motion dominates in the ENM.

Further insight into the agreement between FFEA and ENM models of the ATPase protein complex can be obtained by considering the relative motions of the different subunits of the motor. Figure 4.8 and figure 4.9 show the bulk mobility of four separate domains of the M. sexta V-ATPase motor, in terms of the rotational velocity vector, and the centre of mass velocity vector, for the first three modes in the FFEA and ENM representations. Figure 4.10 shows a quantitative comparison of the agreement in mobility between the four sections in the two representations, and demonstrates that, in agreement with the dot products presented in figure 4.7, the first two modes of flexibility of the ATPase are conserved between the ENM and FFEA.

4.3.2 Comparison of the FFEA dynamics in the M. sexta and the *S. cerevisiae* V-ATPases and the *T. thermophilus* A-ATPase

The eigenvalue spectrum obtained for the six FFEA models is shown in figure 4.11. The larger eigenvalues obtained for the *T. thermophilus* A-ATPase show that it is the most flexible of the three motors. Similarly, the *S. cerevisiae* V-ATPase is marginally more flexible than the *M. sexta*. The differences between the two V-ATPase reconstructions may be attributed to species variation and/or differences in resolution (11Å rather than 17Å) with a decreased volume associated with more detailed structural information permitting larger amplitude thermal fluctuations to occur within the FFEA model. Significant topological changes can affect the output of the simulations therefore it is important to verify the effect of changes in resolution. Figure 4.12 shows the mode comparison between the high resolution EMD-5335 structure (9.7 Å resolution) and the low resolution EMD-1888 structure[108] (16 Å resolution). This is a rather substantial change in resolution, but as expected the low order modes appear broadly unchanged. It is clear that the first three modes of the *T. thermophilus* A-ATPase are resilient to changes in resolution. Both simulations appear to agree on a fourth mode, although not on the ordering (fourth for the EMD-1888, fifth for the EMD-5335). The cumulative proportion of the eigenvalues of the first five modes relative to the total for all modes is given in figure 4.13. This shows that the lower modes in the flexible topologies (those with a severed stator-rotor connection) represent a significantly larger proportion of their total dynamics than the more rigid, uncut topologies.

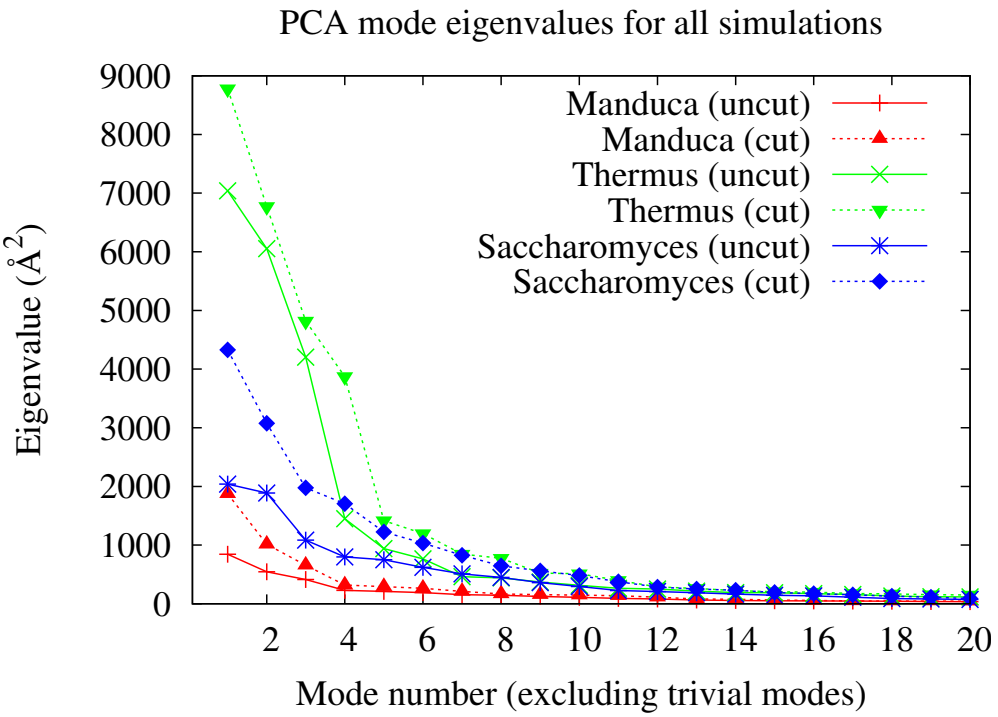


Figure 4.11: Eigenvalues for the first twenty PCA modes in the three cut and three uncut simulations.

Dot-product matrices comparing the eigenvectors of the M. sexta and *S. cerevisiae* V-ATPases (figure 4.14A) show that the major modes of flexibility of the two structures are almost identical. Comparing the eigenvectors of the M. sexta and *S. cerevisiae* V-ATPases with the *T. thermophilus* A-ATPase (figure 4.14B and C respectively) shows that the twisting mode is promoted in the A-ATPase relative to the two V-ATPases, presumably because the former contains only two stator filaments as opposed to three in the V-type complex. This result indicates the importance of connectivity within the structure of the rotary ATPases to the dynamics of these molecular motors.

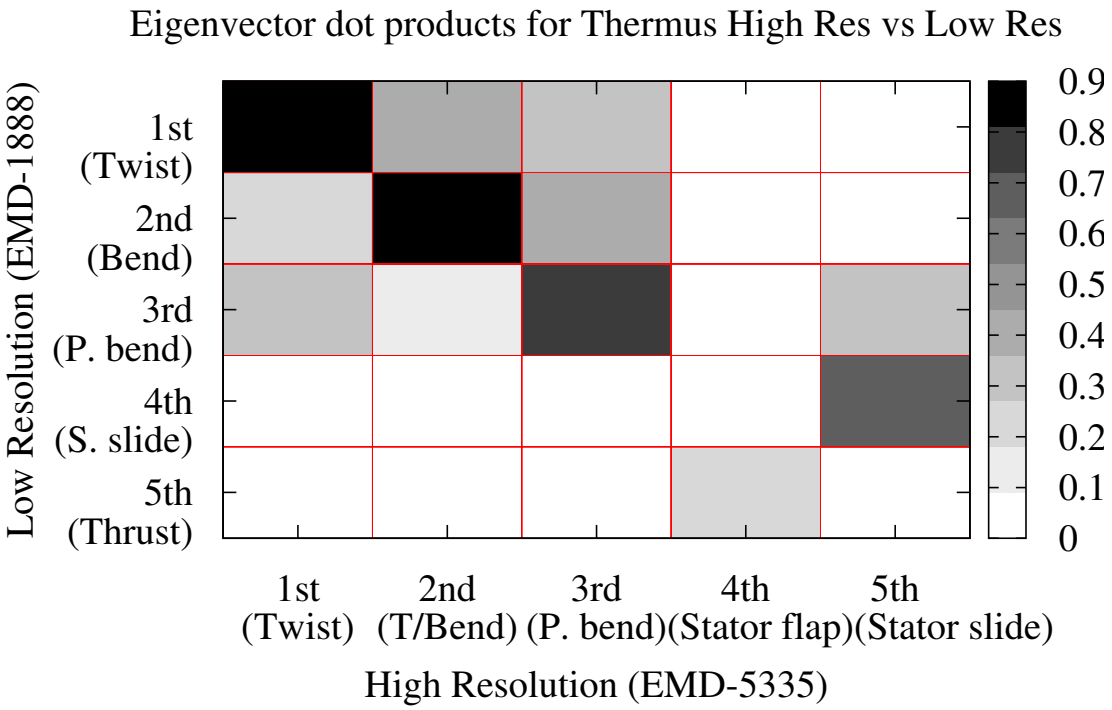


Figure 4.12: Comparison of the FFEA modes for the *T. thermophilus* A-ATPase at 16 Å resolution (EMD-1888) and 9.7 Å resolution (EMD-5335).

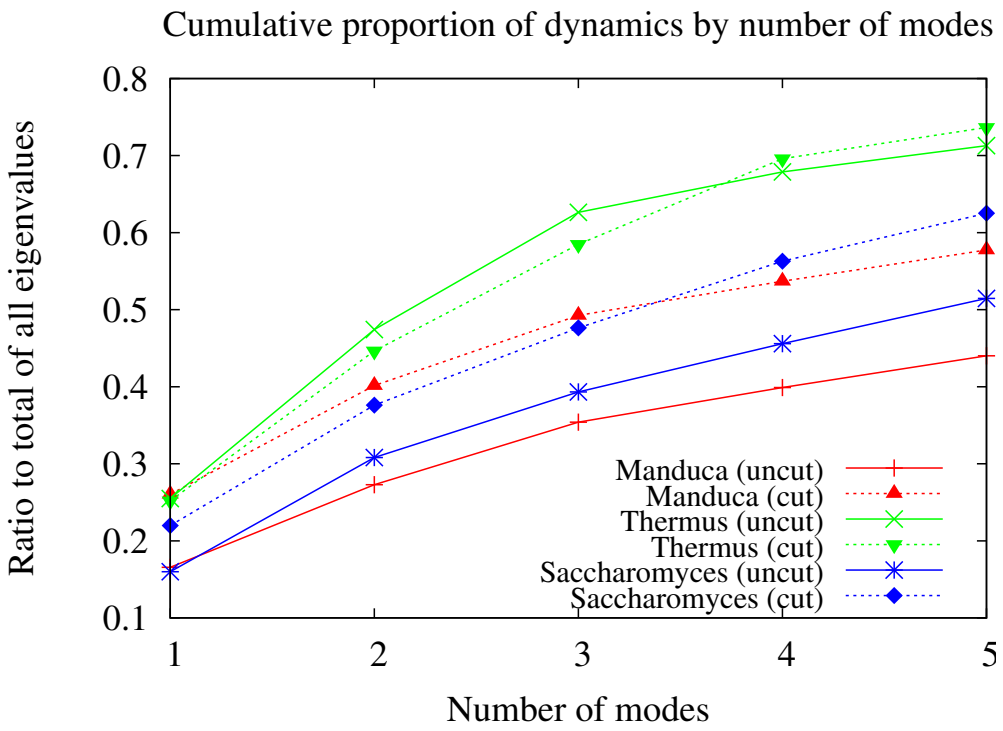


Figure 4.13: Proportion of the total dynamics represented by the first five modes of each simulation trajectory.

4.3.3 Changes in Rotary ATPase FFEA dynamics with Stator-Rotor connectivity

We then used FFEA simulations to explore the effect of disconnecting stator filament 1 (S1 in figure 4.1A) from the rotor (as indicated in figure 4.2) on the dynamics of the three rotary motors. A quantitative comparison of FFEA simulations with and without the connection was obtained by performing PCA on the FFEA trajectories. Comparing the eigenvalue spectrums obtained (see figure 4.11) shows that all three motors have enhanced flexibility when this connection is severed. Taking dot products between the eigenvectors extracted by PCA, figure 4.14D shows that for M. sexta the original twist (figure 4.5A, mode 2) and x-y bend (figure 4.5A, mode 3) motions become mixed when the connection is severed, but that the most important dynamic mode (bending in the y-z plane) persists. For the *T. thermophilus* A-ATPase, all of the top three modes of flexibility are preserved when the connection is severed, as shown in figure 4.14E. Since this motor has only two stator connections as opposed to three, its stiffness is dominated by the central rotor axle. Consequently, changes to stator connectivity have a negligible effect on the dynamics. However, for the *S. cerevisiae* V-ATPase, the major modes of flexibility are more severely affected by severing the stator-rotor connection. In all three of the principal modes, the flexibility is dominated by motion of the unconnected stator local to the point of severance. In the first mode, this motion appears to be coupled to a rotation of the *c*-ring, the second mode is similar to the y-z plane bending mode dominant in the connected system and the third involves a twist of the motor around the central rotor axle. However, since the large flexibility of the stator local to the severance

point dominates in all of these modes, the magnitude of the correlations quantified by the dot-product matrices is reduced (see figure 4.14F). In the higher resolution *S. cerevisiae* V-ATPase in which the stator elements of the structure are better defined, stator 1 becomes sufficiently flexible when it is disconnected from the central axle that its independent motion dominates the dynamics of the motor, and the collectiveness of the dynamics across all three of the top modes is reduced. Nevertheless, the principal modes of the disconnected *S. cerevisiae* V-ATPase do still retain aspects of the bending and twisting modes present in all other systems investigated.

In order to further test the FFEA model for its wider application, it was applied to the V-ATPase dissociation mechanism, by fully severing the V_o, V_1 and C subunits as described in section 4.2.5. Figure 4.16 shows the results of the FFEA simulation of V_o, V_1 and the C subunit dissociating. The separation distance between the centres of mass of V_o and V_1 fluctuates as the two domains detach and reattach. The C subunit breaks away first, weakening the cohesion of the structure which finally dissociates completely.

4.4 Discussion

A common method for visualising low frequency motions of large proteins is Elastic Network Modelling, in which each pseudo-atom is linked to its neighbouring particle by a spring with a defined elasticity. In order to quantitatively test the level of agreement between FFEA and ENM, the eigenvectors extracted from both techniques were compared by taking dot products. The two modelling methods show that bending and twisting of the motor dominates the dynamics in both

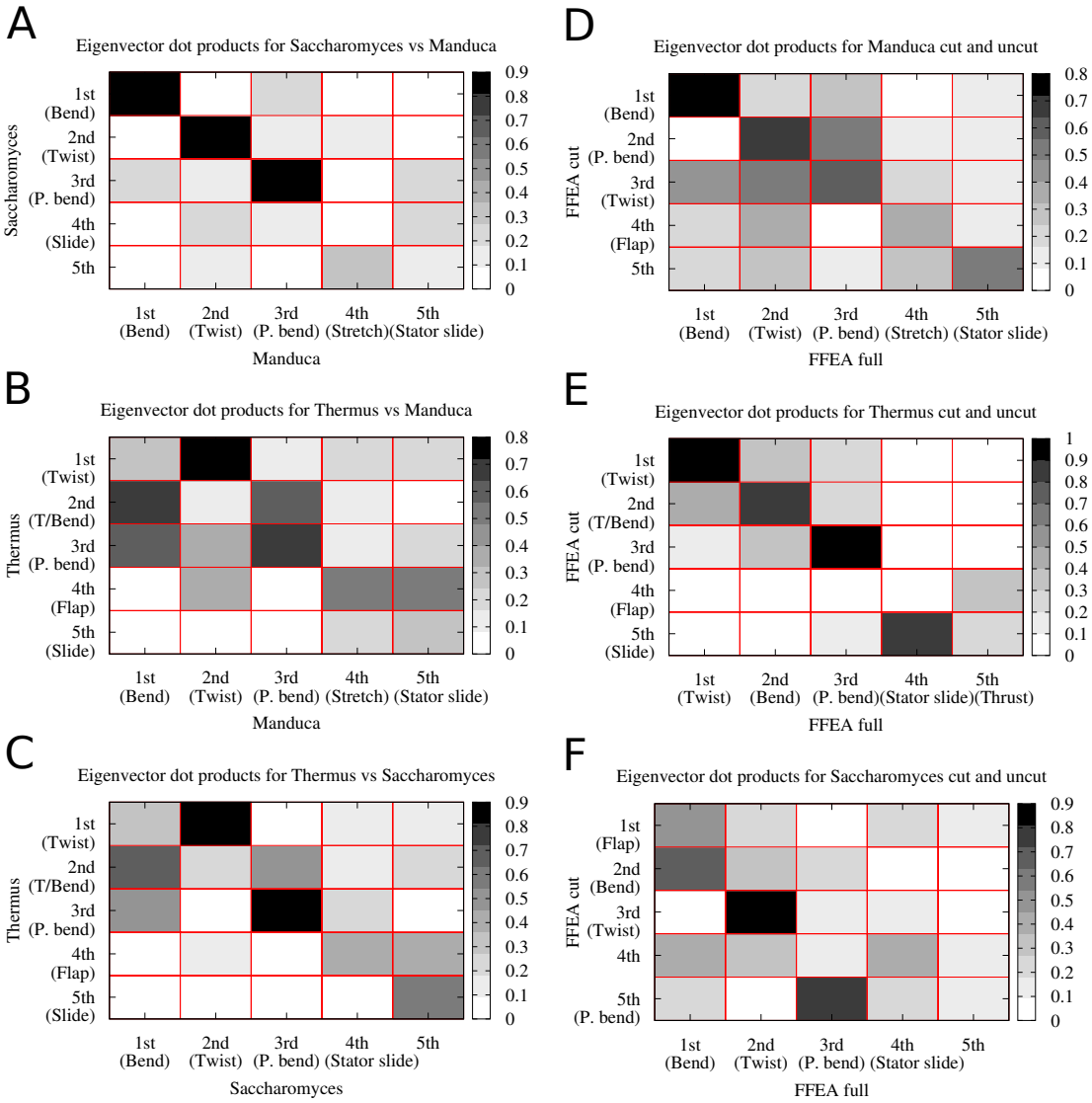
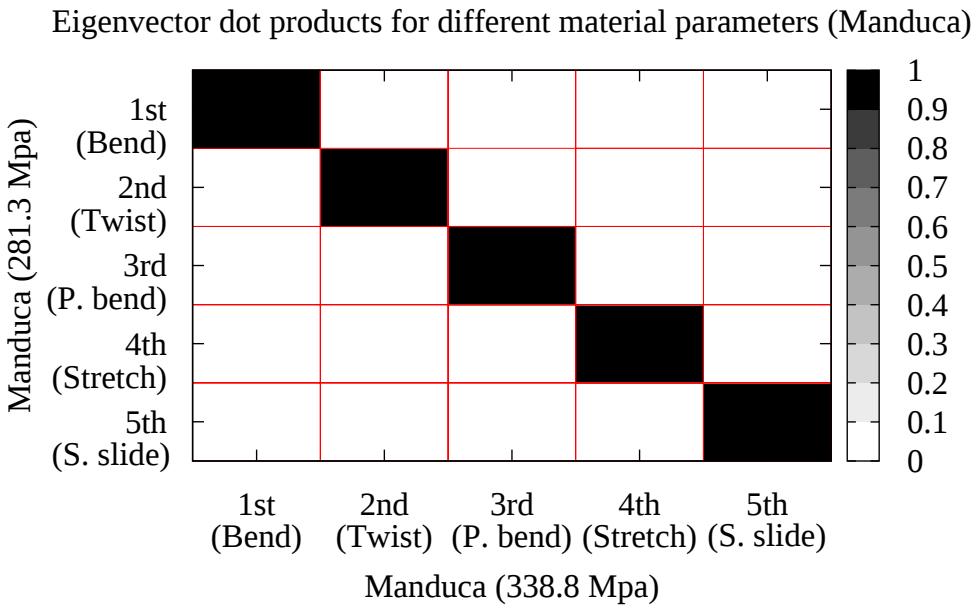


Figure 4.14: Left: Comparison of the FFEA modes for the M. sexta V-ATPase with those of *S. cerevisiae* V-ATPase (**A**), M. sexta V-ATPase with *T. thermophilus* A-ATPase (**B**) and *S. cervisiae* with *T. Thermophilus* (**C**). Right: Comparison of the FFEA modes for the *M. sexta* V-ATPase (**D**), the *T. thermophilus* A-ATPase (**E**) and the *S. cerevisiae* V-ATPase (**F**) with and without the stator connection.

A



B

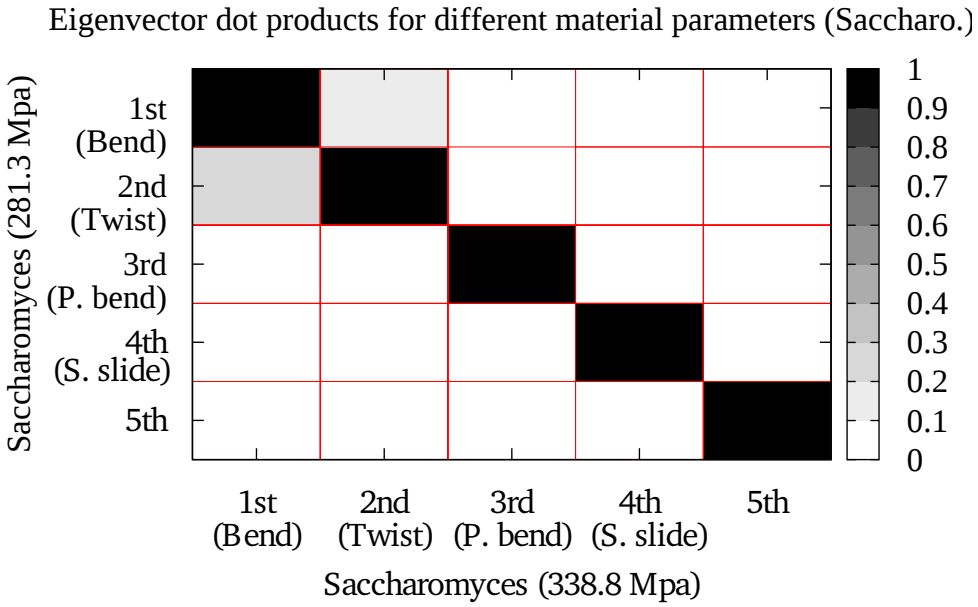


Figure 4.15: A comparison of the PCA modes for *M. sexta* (**A**) and *S. cerevisiae* (**B**) for a Young's modulus of 281.3Mpa and 338.8MPa. The eigenvectors remain almost identical over a 20% increase in Young's modulus, indicating that a different choice of elastic parameters would not change the overall results presented in this chapter.

cases (figures 4.5, 4.6 and 4.8) but diverge after the first two modes.

The single particle cryo-EM reconstructions for the V and A-ATPase have revealed an intricate network of subunits that contribute to the stator of each motor, which allow for the efficient power transfer between the two motor domains [109; 110; 111]. However, implicit within these motors is the need for the central rotor axle and *c*-ring to be able to rotate relative to the V_1/A_1 motor domain and stator network. Importantly, in addition to the a/c interface that is proposed to be the site of proton transport, there is a clearly visible connection between stator 1 in both the A-and V-ATPase and either the central rotor axle or the *c*-ring (figure 4.1). The interface between the a/c subunits is still poorly defined due to the lack of structural data on subunit *a*, however the role of this interface within the family of rotary ATPases in proton transport makes it likely to be a transient interface whereby the *c*-ring can rotate against the *a* subunit. The second interface which is seen in the reconstructions involves a connection between stator 1 and the *c*-ring or rotor axle and its role is unknown. This connection can be seen in both the cryo-EM and negative stain reconstructions and is found in a region away from the detergent which covers the membrane bound regions. In order to see how this connection may influence the inherent flexibility within the complex and the ability of these large complex macro-molecular motors to function, an FFEA approach was used to calculate how the major modes of flexibility are affected by severance of this connection. While disconnecting stator 1 from the central axle increased the flexibility of all three motors, for the higher resolution *S. cerevisiae* structure the major modes of flexibility become dominated by the local motion of the stator, and the collective nature of the dynamics was lost.

The reasons for the linkage between the stator network and the central rotor axle or rotor ring are yet to be determined experimentally, but there are a number of possible roles that this may play. The first possible role is to maintain the *a* subunit *c*-ring interface during rotation. The nature of this interface means it must allow for the *c*-ring to rotate against subunit *a*, but not permit proton leakage, especially if operating close to equilibrium (for which the free energy of ATP hydrolysis is approximately the same as the energy of the established pH gradient), in which a back flow of protons could occur. Moreover, the generation of a pH gradient results in an increased backward rotation pressure on the V-ATPase and a fall in the ATP/ADP ratio equilibrium can result in the proton motive force exceeding the free energy of ATP hydrolysis. The ability of an inactive V-ATPase to still maintain a high proton gradient means that the *c*-ring/*a*-subunit interface must not allow for the back flow of protons when there is a fall in the ATP/ADP ratio. This resistance to backwards rotation occurs through an as yet undetermined process, however the presence of this linkage may act as a ratchet mechanism, permitting rotation in pump mode but stopping rotation in synthesis mode, allowing for the build up of a proton gradient. The apparent flexibility inherent within the V-ATPase system [102] during catalysis can move the stator connection away from the *c*-ring/rotor axle and would permit proton translocation. Upon the lowering of the cellular ATP/ADP ratio, the stator would adopt the position seen in the apparent "low energy/low ATP" ground state (4.1), which would stop proton leakage through reverse rotation of the *c*-ring.

The V-ATPase has been shown to be regulated through a process which involves the dissociation of V_1 from V_o[112]. The linkage present between stator 1

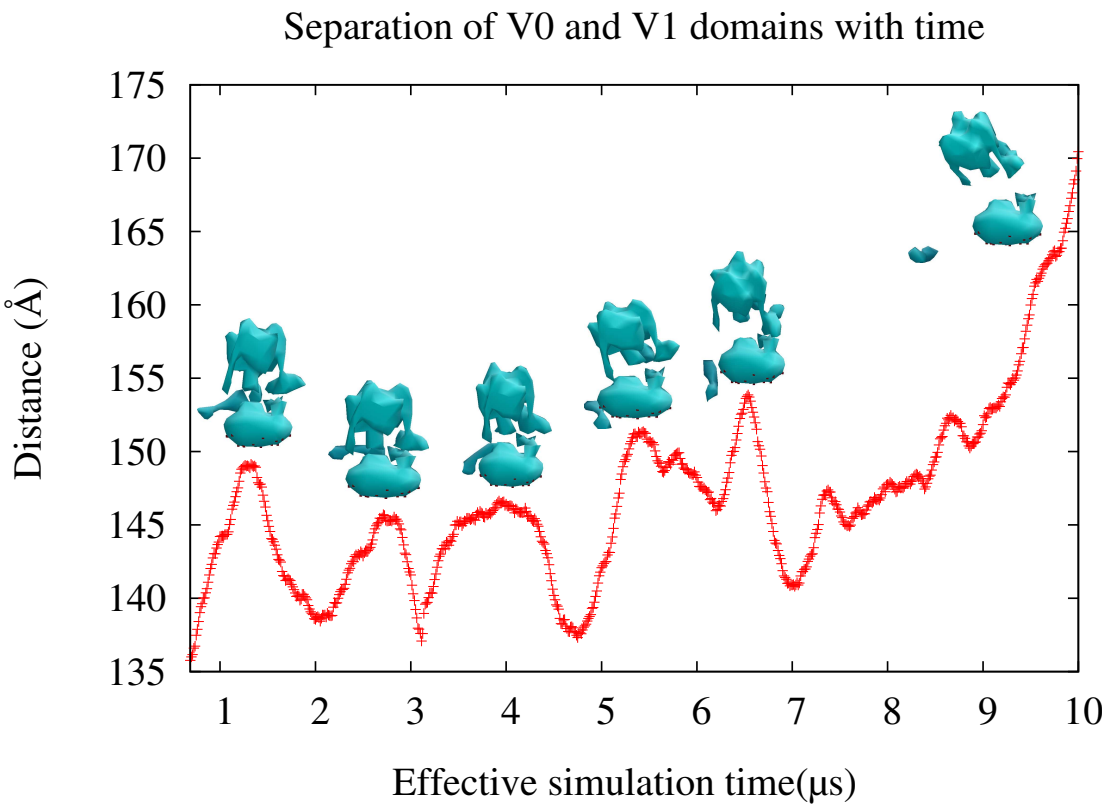


Figure 4.16: The separation of the $V_\circ$ and V_1 domains of the S. Cervisiae with simulation time during dissociation. Simulation snapshots to illustrate the configuration of the $V_\circ$, V_1 and C subunit during the simulation.

and the central rotor axle/c-ring may play a role in this process in tethering V_1 to $V_\circ$. Alternatively it may play a role in re-association of the V-ATPase complex, a process which is currently poorly understood. FFEA can provide insight into this process through simulation of the dissociation of the $V_\circ$ and V_1 domains and the C subunit. Figure 4.16 shows an example trajectory of such a system. Interestingly, the model predicts the departure of subunit C from the complex before full dissociation occurs, which is consistent with subunit C being implicitly involved in this process as it is predicted to, in response to cellular signals, be removed from the complex causing dissociation. Moreover the stator elements show apparent rigidity in the dissociated complex as shown in the cryo-EM reconstruction of the isolated V_1 domain.

4.5 Conclusion

A quantitative comparison of the results of a standard ENM and FFEA approach to simulating the dynamics of two rotary V-ATPases and one rotary A-ATPase has been performed, revealing agreement in the first two modes of motion. Additional simulations suggest that the apparent connection between the stator network and the central rotor axle and c-ring acts to both restrict rotation and limit the available flexibility within the system. The additional flexing seen upon the breakage of this connection may be used to accommodate the cycling between the different ATP bound states of the A_1/V_1 motor, in particular that of the open state which adopts a lower position. Finally, an FFEA simulation of the V-ATPase dissociation mechanism was performed by fully severing the $V_\circ$, V_1 and C subunits. The simulation emphasised the role the C subunit plays in

maintaining the structural integrity of the motor.

Chapter 5

Axonemal dynein c

5.1 Introduction

5.1.1 Motor proteins

While chapter 4 dealt with rotary molecular motors, this chapter is concerned with the application of FFEA to a member of the dynein family of motor proteins. Motor proteins are nanoscopic biomolecular machines involved in essential cellular processes such as mitosis, vesicle tansport and cell motility. As with the V-ATPase proton pump discussed in chapter 4, the action of these motors is fuelled by consumption of ATP. The ATP molecule binds to a region of the motor protein called the motor domain. Subsequent hydrolysis of the ATP into ADP breaks the phosphate bond, releasing chemical energy and resulting in conformational change of the motor protein, referred to as its "power-stroke".

There exist three families of motor proteins in eukaryotes (cells with nuclei): kinesin, dynein and myosin. Kinesin and dynein both bind to microtubules,

whereas myosin binds to actin filaments. Myosin is involved in the contraction of muscle fibres. Dyneins walk towards the minus ends of microtubules thus generally towards the cell nucleus [113], while kinesins typically walk towards the plus end of microtubules thus generally towards the cell periphery [114]. Despite walking on different tracks, kinesin and myosin have similarities in size (around 100 kDa molecular weight) and mechanism. Dynein is a much larger protein (greater than 1 MDa), and as such is likely to have a different mechanism [115; 116].

5.1.2 Dynein

The two main classes of dynein molecular motors in eukaryotes are cytoplasmic and flagellar [118]. Cytoplasmic dyneins are involved in vesicle transport [119; 120], positioning of the nucleus during cell division [121; 122], collecting organelles and proteins from the cell periphery and transporting them towards the interior for degradation [123; 124], and the separation of chromosomes during mitosis, among many others.

Flagellar dynein is responsible for producing the propagating bending motions of cilia and flagella, such as the beating of sperm tails [117]. It is located in a cytoskeletal structure known as the axoneme. Figure 5.1 gives a diagramatic breakdown of a '9 + 2' axoneme (typical in eukaryotes [115]), so-called because it is composed of nine microtubule doublets (forming the cylindrical exterior) and two singlet microtubules down the central axis. An array of different flagellar dyneins are positioned on the interior and exterior sides of each microtubule doublet, referred to as the inner arm and outer arm dyneins respectively. These

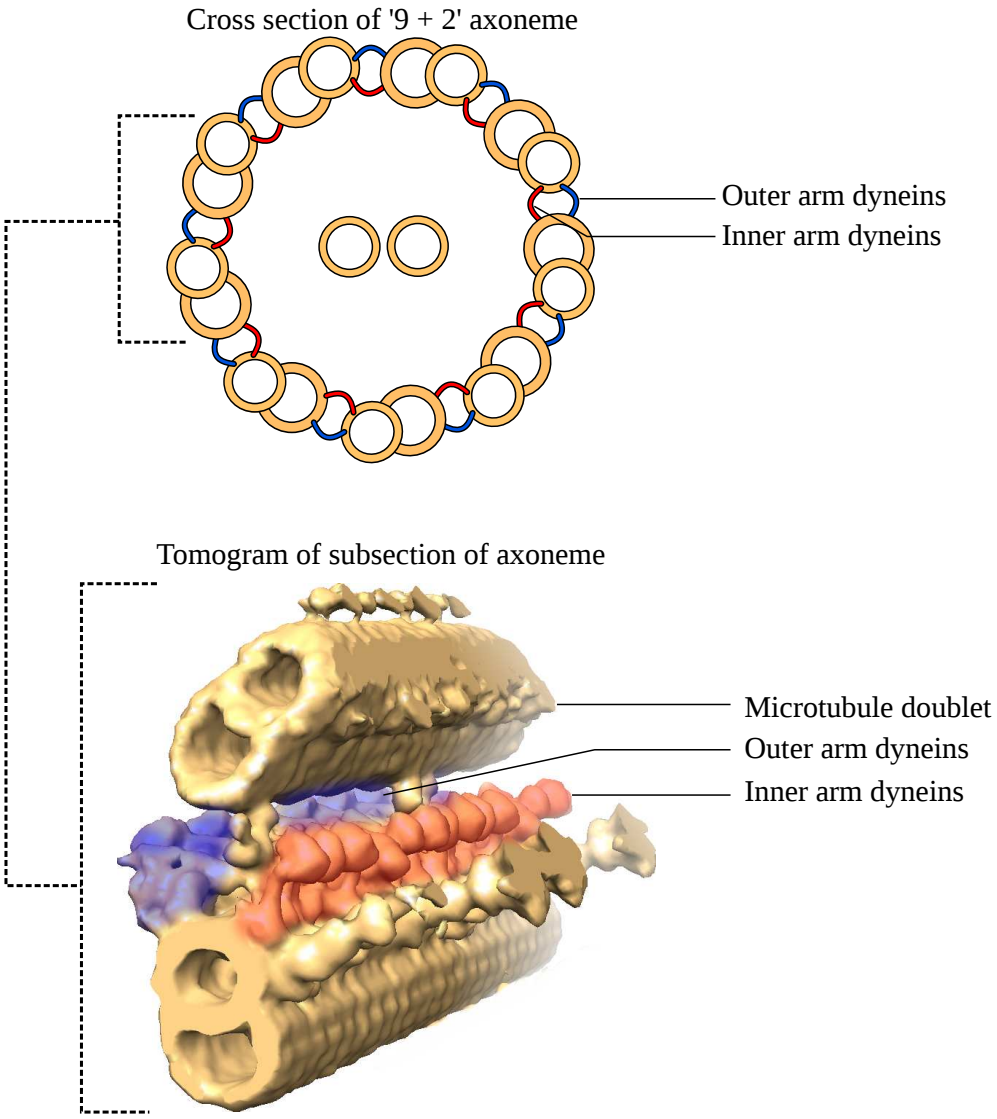


Figure 5.1: Top: A cross-sectional view of a '9 + 2' axoneme, with microtubules shown in orange, inner arm dyneins in red and outer arm dyneins in blue. Bottom: A cryo-EM tomogram of a microtubule doublet with the motor domains of the inner and outer arm dyneins labelled as in the cross-section.

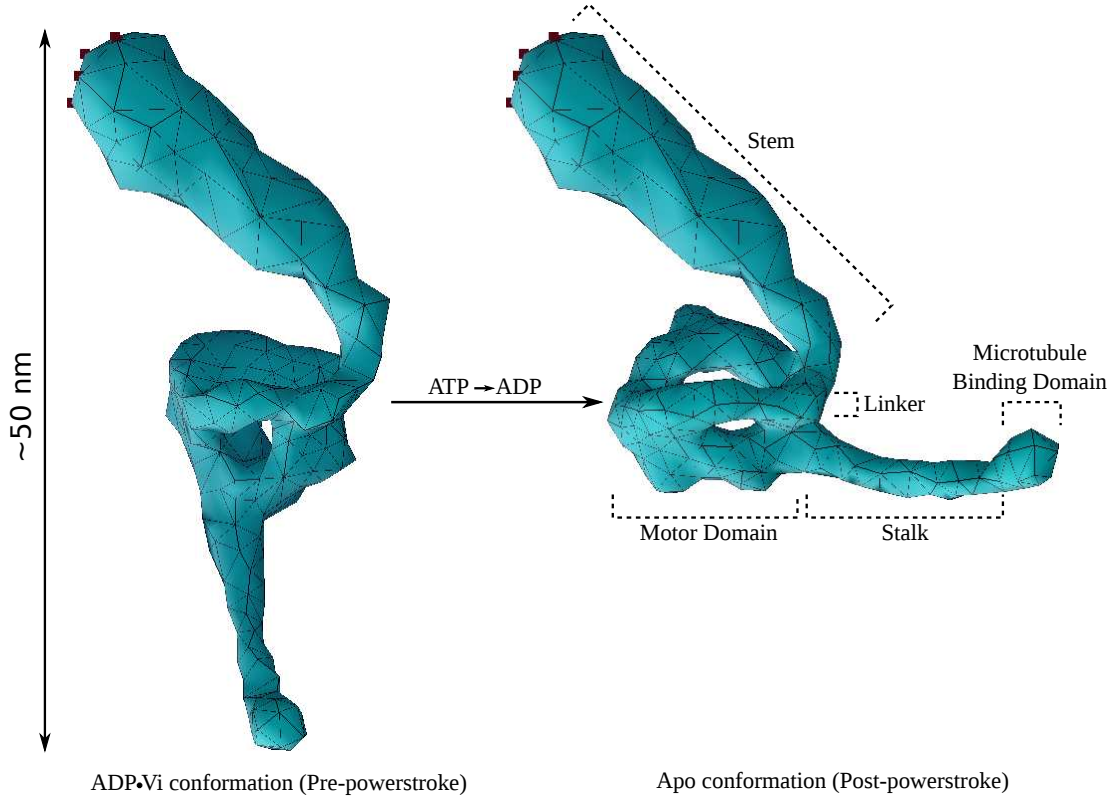


Figure 5.2: The two conformational states of axonemal dynein c as obtained from Cryo-EM experiments [44; 117]. Left: The ADP•Vi state (pre-powerstroke, mimic of ADP•Pi state). Right: The apo state, no nucleotide (no ATP or ADP bound).

are labelled in red and blue in figure 5.1 There are 11 [125] different subspecies of dynein in *Chlamydomonas* flagella, of which 8 are inner arm dyneins and 3 outer arm dyneins[125].

Constructing a realistic simulation model of a fully functioning axoneme would provide a valuable insight into the collective operation of these motors. Towards this goal, this chapter develops a simulation procedure for a single subspecies of axonemal dynein, dynein c, interacting with the axoneme.

Cryo-EM data has been used to build a 3-d structure for axonemal flagellar dynein c [44; 117]. The two experimentally derived conformations are shown in figure 5.2. The ADP•Vi structure is a transition state mimic of ADP•Pi which binds strongly to the microtubule binding sites, and corresponds to the pre-powerstroke conformation. The apo structure is the no nucleotide state (i.e. no ATP or ADP bound to the motor domain). The motor is formed of four regions: the motor domain (globular ring region in the centre), the stalk (which binds to the microtubule track), the stem (which binds to cargo) and the linker (which changes connectivity across the motor domain when undergoing ATP induced conformational change). The small globular domain at the end of the stalk is the microtubule binding domain. Dynein c binds strongly to the microtubule binding sites via this domain, but slips backwards under heavy loads[126], allowing the microtubule doublets to slide past each other during the oscillatory bending motion of the flagella.

While much is known about the axoneme at the molecular level, the mechanism that coordinates the dynein motors to produce the rythmic oscillatory motion is as yet unknown, but is believed to be regulated by some 'switching' event (which causes the flagellum to bend the other way)[127]. This switch-

ing event has been hypothesised to occur through elastic recoil, relative sliding of microtubules passing a 'switch point', or the flagellum reaching a threshold curvature or threshold transverse force, among other potential causes[128]. A functioning simulation model of the axoneme could provide important insights into the nature of the switching event, and a means of testing hypotheses.

5.1.3 Prior work on modelling dynein c

Negative stain electron microscopy experiments by Burgess *et al.* produced images of dynein in solution for both the ADP•Vi and apo conformations, from which could be extracted histograms of the length (from stem tip to stalk tip) and angle (between stem and stalk) distributions [118]. Using this data, R. C. Oliver determined inhomogeneous elastic parameters for both conformations that matched the same statistical variance [44]. The mean values for the ADP•Vi and apo states in the length distribution and apo in the anglular distribution also showed good agreement. The mean angle for the ADP•Vi state was found to differ by around 20 degrees from that measured by Burgess *et al.*, and could be due to distortions caused by the drying process of the 3-d structure onto the 2-d film[44]. The mean angle is determined by the cryo-EM derived structure and not by the elastic parameters.

The Young's moduli determined for the stalk and stem are given in table 5.1 (all remaining parameters for this chapter are detailed in table 5.2). The material parameters for the motor domain were determined by a linear interpolation between the stalk and the stem to ensure a smooth transition in material properties. The Poisson ratio did not have a significant effect on distribution and was fixed

at 0.35 for all elements in the mesh.

State	Stalk Y (MPa)	Stem Y (MPa)
ADP•Vi	175	125
Apo	400	100

Table 5.1: Inhomogeneous parameters (Young's modulus) for dynein stalk and stem as determined by Oliver [44].

5.1.4 Aims

The aim of the work presented in this chapter is to progress the development of a fully functioning simulation model of the axoneme in order to understand the mechanism that coordinates the dynein motors to produce the characteristic oscillatory motion of flagella. Towards this aim, we attempt to determine what characteristics dynein c may have *in vivo* by simulating its functioning in its biologically relevant environment, the axoneme. This chapter is concerned with measures such as the step length of the motor and the potential reach of the microtubule binding domain, and what effect interaction with the axoneme might have.

5.2 Validation of elastic parameters

As a validation of the restructured code, the simulations described in section 5.2 were repeated and the length and angle distributions recalculated and checked against the experimental data in reference [118]. Eighteen 2μs simulations (nine for each state) were run with the different material parameters described in reference [44]. The results for the best fit elastic parameters (table 5.1) are given in

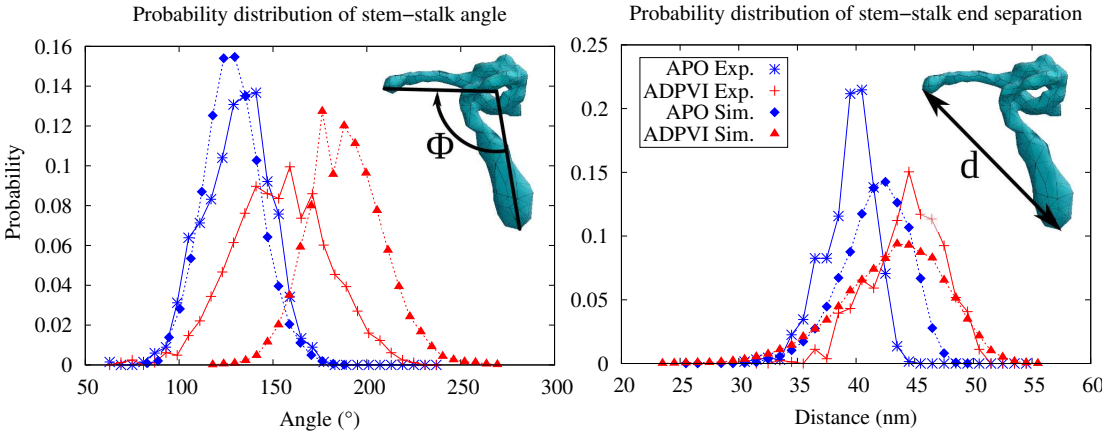


Figure 5.3: Length and angle distributions for dynein c calculated from 2μs critically damped simulations against distributions obtained from experiment [118].

figure 5.3, showing very similar agreement with those obtained by R. C. Oliver using a different implementation of FFEA. The same discrepancy in mean angle for the ADP•Vi state is observed, as discussed in the previous section. The distance d and angle ϕ (as indicated in figure 5.3) were extracted from the simulation trajectories by transforming each frame such that the view is straight through the motor domain ring, as it was in the experiments. The transformation was carried out via the Gram-Schmidt process [129].

Density	1500 kg m^{-3}
Poissons Ratio	0.35
Shear Viscosity	1 mPa.s
Bulk Viscosity	1 mPa.s
External Viscosity	1 μ Pa.s
Time step	10 fs
Temperature	290 K

Table 5.2: FFEA simulation parameters for all simulations in Chapter 5.

5.3 Meshing the Axoneme

The density map of the two microtubule doublets[1] represents two of the nine microtubule doublets in *Chlamydomonas reinhardtii* flagella. A surface mesh was extracted from the density map with a linearly interpolated marching cubes algorithm (section 2.3.1) at an isolevel of 119 and culling debris with volume of 1514 voxels or less. Voxel averaging was not appropriate for this map as the fine detail is easily destroyed even for $2 \times 2 \times 2$ averaging. The resultant surface mesh was then coarse-grained using the face-pair-collapse method (section 2.3.2) with a 25Å minimum edge length in the interaction region. This process is illustrated in figure 5.4, showing the effect of coarse-graining the surface mesh and cleaning up debris. Note that despite a 95% reduction in the number of faces (739720 to 35108) using a very basic algorithm, the final mesh still retains all the essential shape and features of the original mesh.

5.4 Alignment of Dynein in the axoneme

In order to simulate dynein's interaction with the axoneme, it was necessary to determine its position and orientation within the axoneme. The full axoneme tomogram (as seen in figures 5.1 and 5.5) is problematic because, despite being obtained from the system in its apo state, the motor domains are lowered towards the cargo microtubule, and as such do not correspond to the apo dynein conformation obtained from cryo-EM of a solution of free dyneins. To remedy this, two higher resolution boxed tomograms of the apo and the ADP•Vi state can be used. The apo tomogram is used to find the position along the axoneme both boxed

[1] A kind gift of Dr Takashi Ishikawa, created from the data in reference [130].

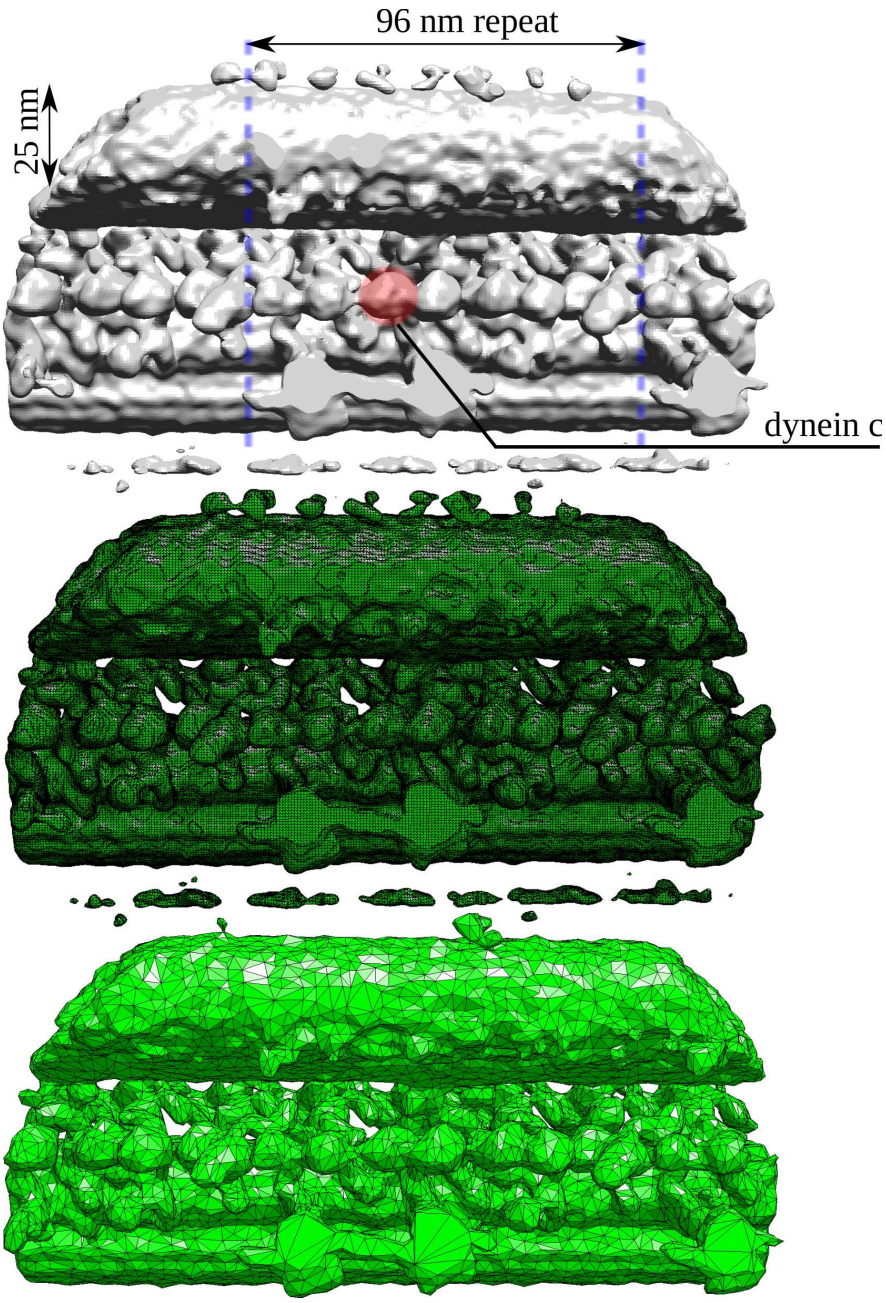


Figure 5.4: Preparation of the static axoneme surface mesh from the experimentally derived tomogram. Top: The tomogram rendered at an isolevel of 119. The location of the dynein c motor domain is highlighted in red. The diameter of the minor microtubule in the doublet is 25 nm. The length of the repeat pattern along the axoneme is 96 nm. Middle: The high resolution surface mesh extracted through linearly interpolated marching cubes. Bottom: The surface mesh after coarse-graining and clean-up of surrounding debris.

regions represent. Then, the ADP•Vi tomogram was located in the same place. A 'fit selection' was cut from the ADP•Vi tomogram (at isolevel 1.79). Using a similar process to that used for alignment of the ATPase models in chapter 4, the ADP•Vi mesh was aligned within this fit selection by minimising the sum of the distances between each pair of nodes. As the tomogram effectively represents a time average of the axoneme, the stalks of the various motors are averaged out. Therefore only the stem and motor domain of the dynein model are involved in the fitting process. Figure 5.5 shows the fitting process.

5.5 Interaction of dynein c with axoneme

5.5.1 Method

Using the elastic parameters given in table 5.1, and the remaining parameters in table 5.2, eighteen simulations (nine for each conformation) were run at varying vdW surface interaction energies: 0, 10^{12}, 10^{13}, 10^{14}, 10^{15}, 2×10^{15}, 3×10^{15}, 10^{16} and 10^{17} Jm^{-4}. The simulations were run till the trajectories exceeded 3μs in the case of non-zero interaction energy (and 6μs for the non-interacting simulations). The node positions were output every 100ps. The nearest neighbour lookup grid was set at $25 \times 30 \times 50$ with a cell edge length of 4 nm. The axoneme and dynein meshes were patterned as shown in figure 5.6 and designated 'STATIC', while the apo and ADP•Vi models were designated 'DYNAMIC' (see section 2.3.4).

The node positions of the ADP•Vi mesh over the entire trajectory were binned in a fixed $50 \times 60 \times 100$ cell box with dimensions $1000 \times 1200 \times 2000$ Å producing a density map of the simulation data. The resulting maps (at similar surface

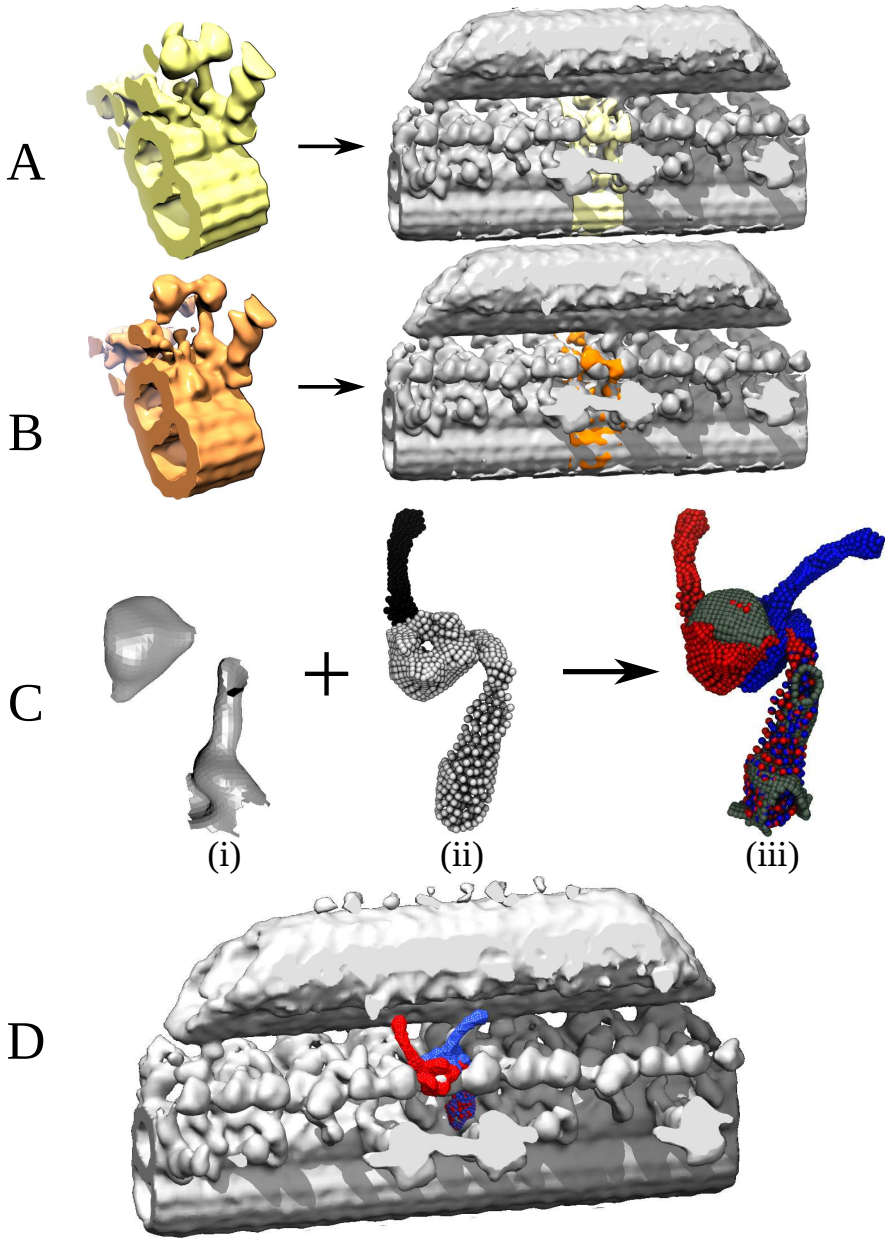


Figure 5.5: Alignment of the dynein models within the axoneme. **A**: Best fit positioning of a higher resolution boxed apo tomogram in the main axoneme tomogram. **B**: Positioning of boxed ADP•Vi tomogram for the same region, located according to the position of **A**. **C**: Alignment of ADP•Vi dynein model in a selection of the boxed ADP•Vi tomogram. **(i)** The fit selection, cut out from B, excluding the neck region. **(ii)** Nodes of the ADP•Vi mesh model, with active nodes (those involved in the alignment) in white, and inactive nodes in black. **(iii)** The final calculated alignment of the ADP•Vi model (red) in the fit selection (grey). The alignment of the apo state (blue) is determined from the shared stem. **D**: Both dynein states aligned in the axoneme tomogram.

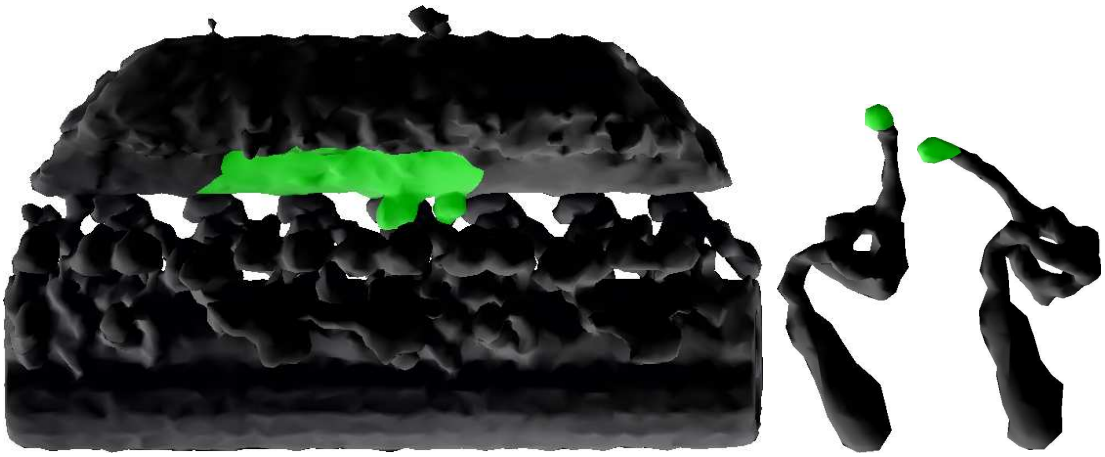


Figure 5.6: vdW surface patterning of the FFEA models. Black indicates non-interacting faces, and green indicates interacting.

levels) are compared with the corresponding motor domain and stem from the original axoneme tomogram in figure 5.7.

The positions of the nodes corresponding to the microtubule binding domain (tip of the stalk) were then extracted from all apo and ADP•Vi trajectories and binned in histograms of the same dimension as above. The maps were normalised by the total number of nodes binned, producing a probability density map of the binding domain's position at various interaction energies between the binding domain and the microtubule track. For the purpose of data visualisation, the density levels corresponding to 10%, 30%, 60% and 95% total probability were calculated for each map. This was achieved by sorting the probability-per-bin for all bins in descending order, and summing the bin probabilities until the cumulative probability matched the desired total probability. As an exact match would be rare, the level was chosen by linearly interpolating between the bin probability directly before and after the desired total probability is attained.

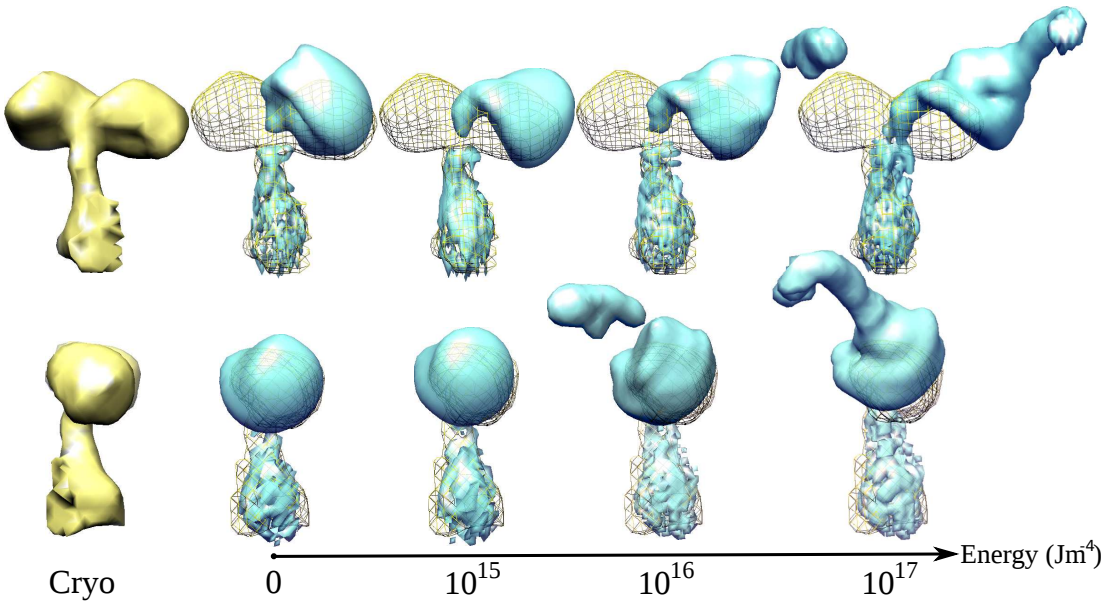


Figure 5.7: Density maps of ADP•Vi dynein (blue) generated from simulations at varying surface interaction energy, compared with the relevant section of the axoneme tomogram (yellow, solid and wire frame) seen from the front (top row) and side (bottom row). The isolevels were chosen to give similar surface levels. Note that the stalk is averaged out at lower interaction energies due to high mobility.

This interpolation (h) and the resulting isolevel (L) is given in equation (5.1):

$$h = \frac{(P - S_{i-1})}{(S_i - S_{i-1})}$$
$$L = B_{i-1} + h(B_i - B_{i-1}) \tag{5.1}$$

where P is the desired cumulative probability, i is the bin index, S_{i-1} and S_i are the sum of bin probabilities before and after S has surpassed P respectively, and B_{i-1} and B_i are the actual probabilities of being in bin $i-1$ and i respectively. The process of determining levels corresponding to $P = 0.1, 0.3, 0.6, 0.95$ is given in figure 5.8 using the $\epsilon_{LJ} = 10^{15}$ Jm^{-4} ADP•Vi run as an example. This analysis was carried out for each simulation's probability density map, and the four contour surfaces plotted. The resultant probability maps are given in figures 5.9 and 5.10 (for only three interaction energies, in the interest of space).

The step length of the motor at each energy was determined by calculating the mean position of the stalk head in each conformational state independently and finding the distance between these two positions. The standard deviation of the head about the mean in each state ($\Delta_{ADP\bullet Vi}$ and Δ_{apo}) was also calculated for each state. The standard deviation in the step length was therefore calculated as $\sqrt{\Delta^2_{ADP\bullet Vi} + \Delta^2_{apo}}$. The step length at each interaction energy is given in figure 5.11 with the standard error as error bars. The standard error ($\frac{\Delta}{\sqrt{N}}$, where Δ is the standard deviation and N the number of data points) is very small as the simulations are converged. The variation in step length is therefore a feature of the steric repulsion and attraction to the microtubule surface.

A helical set of points representing the positions of the microtubule binding sites was then created, as shown in figure 5.12A. By comparing each of these

sites with their corresponding bins in the probability density histograms, it was possible to count how many sites were visited by the microtubule binding domain in each of the four probability levels ($P = 0.1, 0.3, 0.6, 0.95$). This analysis was carried out for all the simulations, and the results are presented in figures 5.12B and 5.12C.

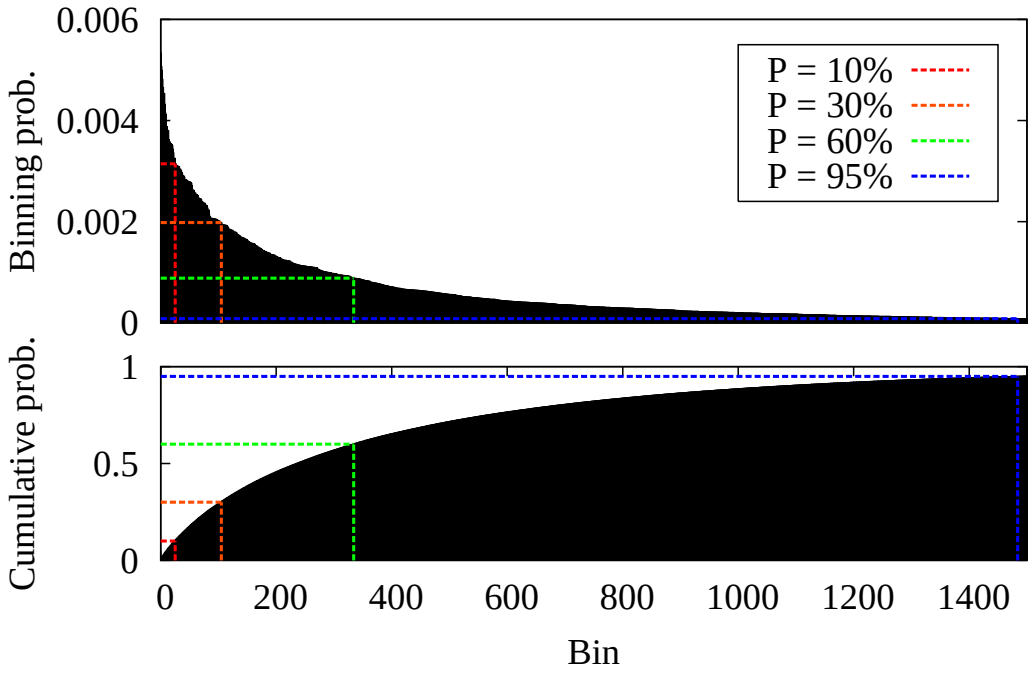


Figure 5.8: Isolevel choice based on cumulative binning probability for ADP•Vi run $\epsilon_{LJ} = 10^{15}$ Jm^{-4}. Top: The probability of the microtubule binding domain lying in each bin, sorted in descending order. Bottom: The cumulative probability of the binding domain lying in all preceding bins. This graph represents a running total of the binning probabilities in the top graph. When the cumulative probability reaches the desired probability (10%, 30%, 60% or 95%), the probability of the bin at which this occurs determines the isolevel.

5.5.2 Results

The results of the simulations and analyses described in section 5.5.1 are presented in figures 5.7, 5.11, 5.9, 5.10 and 5.12.

Figure 5.7 shows the density maps generated from the ADP•Vi trajectories at various interaction energies. It is presented as a sanity check that the simulations do not predict an entirely different map to that seen in the axoneme tomogram. We see from this figure that at energies of 10^{16} Jm^{-4} and above, the stalk is strongly fixed to the microtubule and is no longer time averaged out as it should be. This suggests that energies of order 10^{15} Jm^{-4} give the closest match to the real interaction.

Figure 5.11 gives the calculated step length of the motor at different interaction energies with the MT, with error bars indicating the standard deviation in step length. This indicates that the presence of the axoneme has an effect on the step length of the motor. The step length is increased from around 24.7 nm (in the non interacting case) to around 27.5 nm (for $\epsilon_{LJ} = 10^{13}\mathrm{Jm}^{-4}$), due to steric effects preventing the binding domain from adopting its preferred equilibrium position. Strong attraction to the microtubule increases this further, to 28.7 nm, at the expense of exploration of the stalk head (the standard deviation drops from around 6.7 nm to around 1.4 nm).

Qualitatively, from figures 5.9 and 5.10 we can also see that the ADP•Vi state has an enormous head search volume as compared to apo, and is able to reach around the other side of the doublet, accessing binding sites on both microtubules. We note that even in the non-interacting case, the apo state dynein has a markedly smaller stalk head search (standard deviation 3.9 nm) as compared to the ADP•Vi

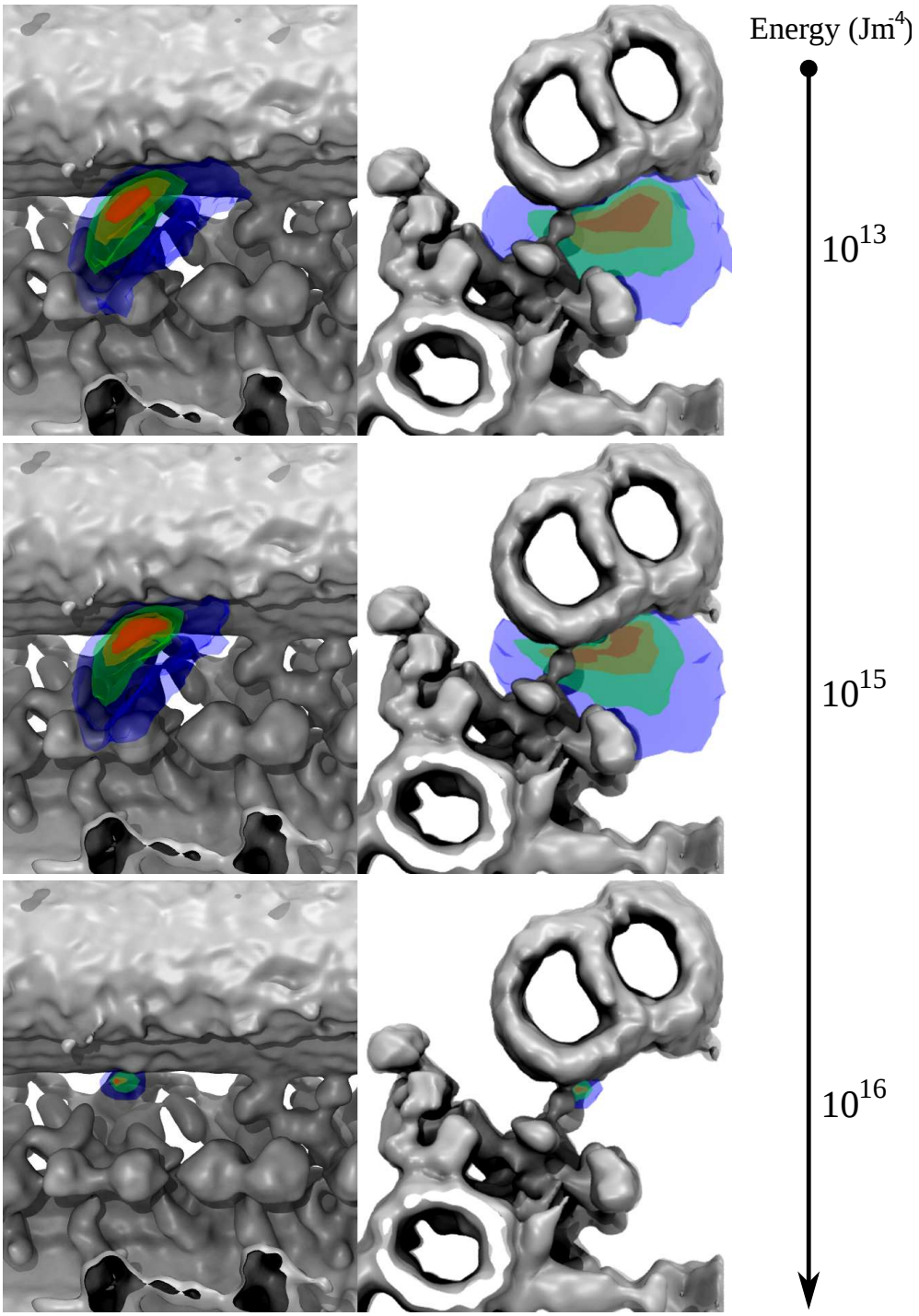


Figure 5.9: Probability density maps of microtubule binding domain spatial search as calculated from simulation trajectories of ADP•Vi dynein at interaction energies $\epsilon_{LJ} = 10^{13}, 10^{15}, 10^{16}$ (top, middle and bottom, respectively). The head spends 95% of its time within the blue surface, 60% within the green, 30% within the orange and 10% within the red. Two viewing angles are shown for each energy: perpendicular to the axoneme (left) and down the centre of the axoneme (right).

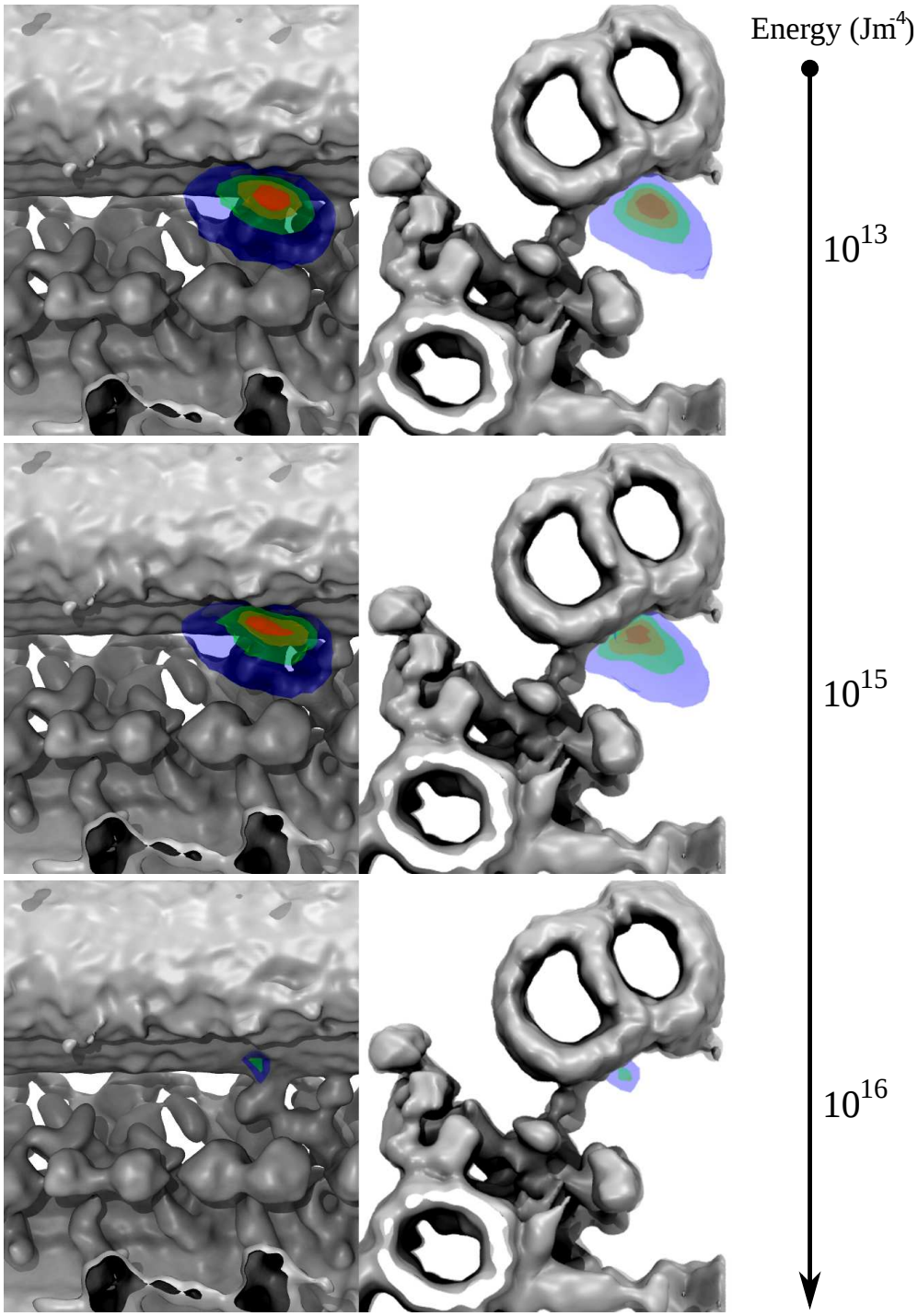


Figure 5.10: Probability density maps of microtubule binding domain spatial search as calculated from simulation trajectories of apo dynein at interaction energies $\epsilon_{LJ} = 10^{13}, 10^{15}, 10^{16}$ (top, middle and bottom, respectively). The head spends 95% of its time within the blue surface, 60% within the green, 30% within the orange and 10% within the red. Two viewing angles are shown for each energy: perpendicular to the axoneme (left) and down the centre of the axoneme (right).

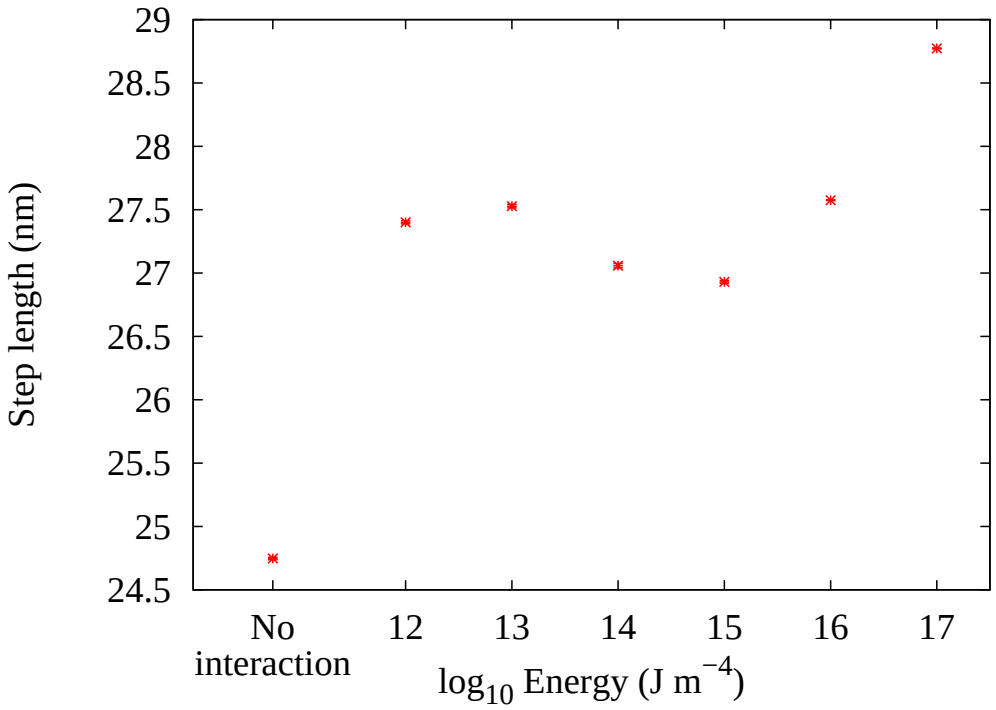


Figure 5.11: Step length of axonemal dynein c in the presence of the axoneme. Error bars show the standard error for each interaction energy. The standard errors in the means were of order $\frac{1}{100}$ of a nm.

state (standard deviation 5.6 nm), but the presence of the axoneme amplifies this difference. Indeed, it would appear (from figure 5.11) that the hard-body interaction alone can bias the configuration towards a larger step length.

At very strong interaction energies (10^{16}Jm^{-4} upwards) there appears to be a strongly favoured position for the stalk head to adopt. As the MT mesh has retained more or less all the detail of the original tomogram, this is due to a feature of the experimental data. This could be an experimental artifact, or real (although there is not necessarily a microtubule binding site at that location).

Finally, it is useful to consider a quantitative measure of the reach of the stalk head, in terms of number of microtubule binding sites. The results of this calculation are given in figure 5.12B and 5.12C (5.12A shows the pattern of binding sites on the microtubule doublet). In the flagellum microtubules are fused together into doublets, but the protofilaments and spacings are similar to those in singlets. We have therefore used 8 nm axial spacing between binding sites, with a 12 nm pitch[131]. As there are typically 13 protofilaments making up the MT [131], this results in a helical distribution with each protofilament staggered by 0.9 nm relative to its neighbour.

The bar charts show how many sites the stalk head explores at each probability level in each conformation. For example, at 10^{15} Jm^{-4} the ADP•Vi has the ability to reach around 18 binding sites (95% probability), although it spends 30% of its time exploring the same 5 binding sites. For the same energy in the apo state, the head only searches 7 binding sites at 95% and 2 at 30%. This corresponds with the large difference in search volume between the two conformations observable in figures 5.9 and 5.10.

The range of interaction energies allows us to see the effect of introducing the

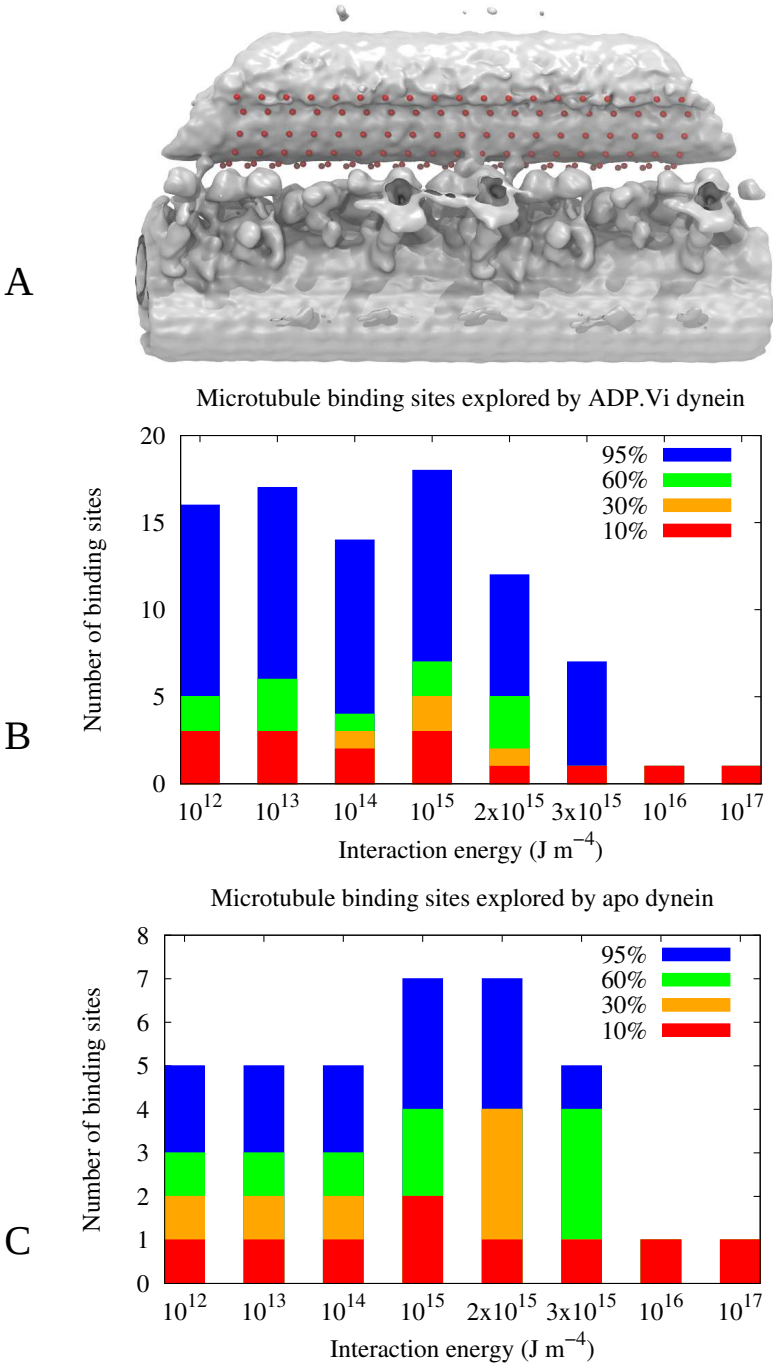


Figure 5.12: **A**: Position of microtubule binding sites on microtubule surface (8 nm separation, 0.9 nm staggered), indicated by red spheres. **B**: The number of microtubule binding sites explored by the microtubule binding domain of dynein c in its ADP•Vi state. **C**: The number of microtubule binding sites explored by the microtubule binding domain of dynein c in its apo state.

microtubule doublet. As discussed above, figure 5.7 shows 10^{16} Jm^{-4} to be likely too strong an interaction. As we know from experiment that the head remains close to the MT surface, we treat the results for 2×10^{15} Jm^{-4} and 3×10^{15} Jm^{-4} as most likely closest to modelling the real interaction strength. Therefore the model as parameterised suggests that dynein's microtubule binding domain can explore 10 to 15 binding sites in its ADP•Vi conformation, and 7 sites in the apo state.

Interestingly, figure 5.12B suggests that there is an optimum binding energy with respect to sampling of MT binding sites. In the above simulations, this corresponds to interaction energies of order 10^{15}Jm^{-4}.

5.6 Patterning microtubule binding sites

In order to investigate the effect on the ADP•Vi stalk head search in the presence of interacting microtubule binding sites, the microtubule was patterned with more strongly interacting patches, as shown in figure 5.13. Seven 2μs simulations were run at energies of $1 \times 10^{15}, 2 \times 10^{15}, 3 \times 10^{15}, 4 \times 10^{15}, 5 \times 10^{15}, 10^{16}$ and 10^{17} Jm^{-4}. The number of MT binding sites explored by the stalk head was calculated as previously.

Comparing figure 5.14 with figure 5.12 we can see that the results are very similar, suggesting that the binding sites are sufficiently close together on the scale of the binding domain that the initial uniformly attractive surface used in section 5.5 is a good approximation. Furthermore, the microtubules themselves slide past each other during the functioning of the axoneme, meaning that the location of the binding sites is likely to be smeared out in terms of the interaction

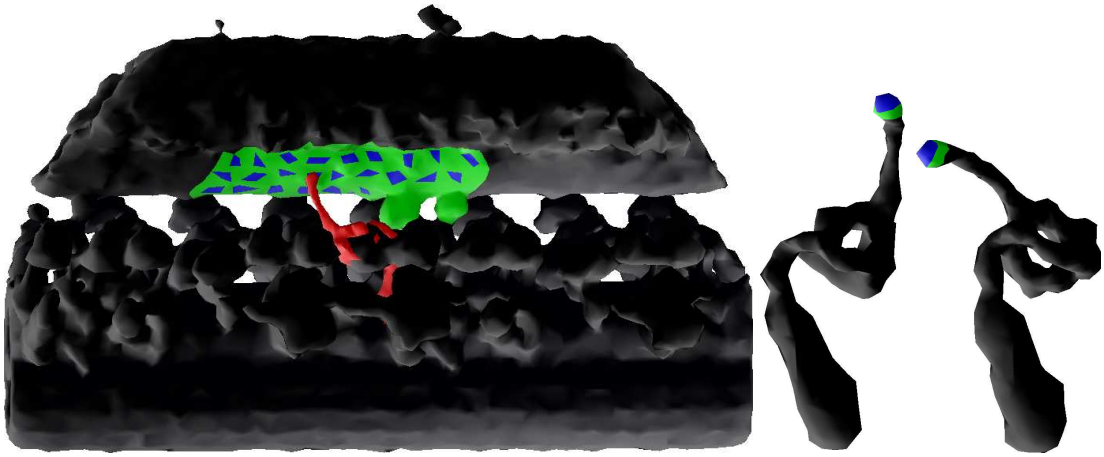


Figure 5.13: vdW surface patterning of the FFEA models for simulations including the microtubule binding sites. As in figure 5.6, black indicates non-interacting faces. However, now there are two types of interacting faces: green and blue. Green faces represent steric repulsion interactions. Blue faces correspond to microtubule binding sites on the axoneme (8 nm repeating [126] with 0.9 nm stagger), and are strongly interacting.

experienced by the stalk head (multiple dynein motors can move the microtubule at 5.1μm s^{-1} [126]).

5.7 Switching between apo and ADP•Vi states

The previous section is concerned with simulating dynein c in its two conformations independently. However, in reality this motor frequently undergoes conformational changes as part of its function. A simulation model of a fully functioning axoneme therefore requires the molecular motors to be able to change conformation during the course of the simulation. While this is a relatively trivial problem for different conformations of the same mesh (we need simply define a new equilibrium structure), it is non-trivial in this case due to the two dynein states having

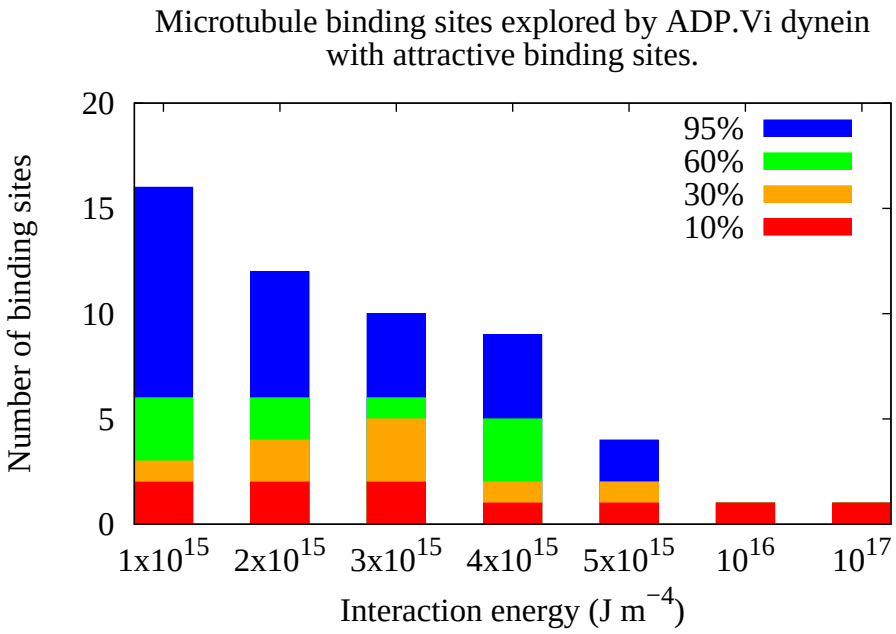


Figure 5.14: Number of microtubule binding sites explored by dynein's binding domain in the case of the vdW patterned axoneme (shown in figure 5.13). Note that the numbers are similar to those from figure 5.12B.

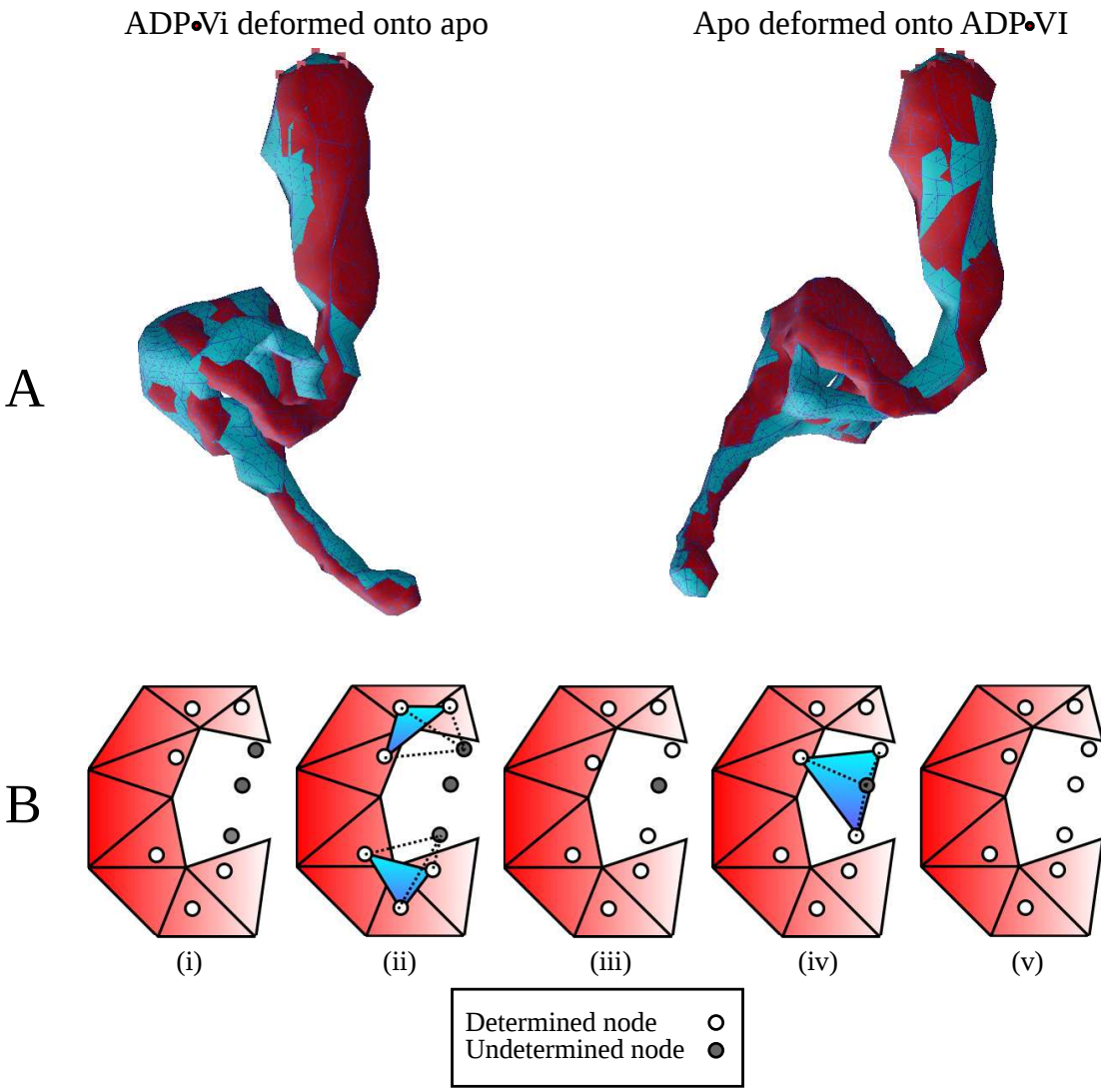


Figure 5.15: Row **A**: Deformation of the ADP•Vi mesh onto the rest state apo mesh and vice versa. Control nodes were mainly along the stalk and stem, while the motor domain was free to adopt a suitable orientation. Row **B**: Illustration of the mixed mapping process (in 2-d, for simplicity). The α state is shown in red, and the nodes of the β state shown as circles. White circles indicate that the node's position has been determined, and grey circles are yet to be determined. **(i)** Those β nodes that lie directly in an element of α are determined by the barycentric coordinates of their position in that triangle. Those nodes lying outside α are undetermined. **(ii) and (iii)** New triangles are formed from existing determined nodes. Those with the shortest distance to an undetermined node are used to fix its position. **(iv)** In this case it is possible to form a triangle from determined nodes that encompasses the remaining undetermined node. This is preferential. **(v)** All node positions for β have been determined in terms of elements in α or determined nodes in β.

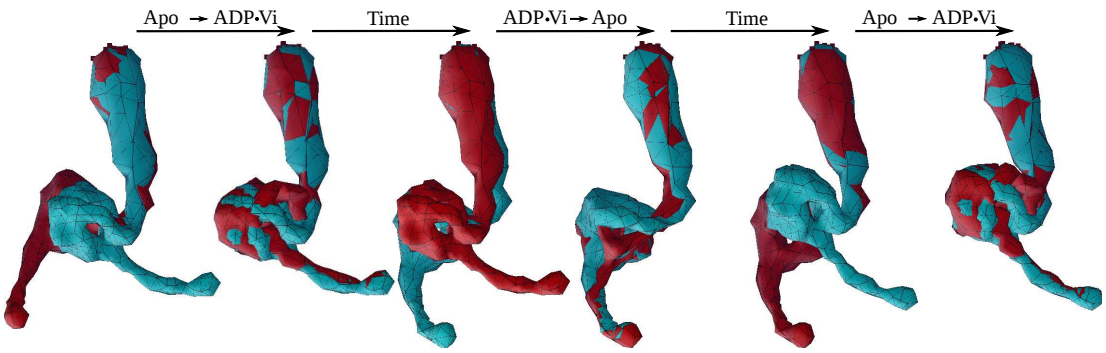


Figure 5.16: Key frames from a simulation of dynein undergoing changes in conformational state, showing both the active (blue) and inactive (red) FFEA models. We begin (far left) with apo as the active model. The nodes of the currently inactive ADP•Vi model are then mapped to give the closest valid fit to the apo model. ADP•Vi now becomes the active model, and is allowed to fluctuate away from this position. After some number of steps, the nodes of the (currently inactive) apo model, are mapped to give the closest valid fit to the ADP•Vi model. Apo becomes the new active model, and the process continues anew.

different meshes with different topologies, and which do not overlap. As such, there is no clear one-to-one mapping between the node positions in one mesh and those in the other.

If our protein is in some conformation, α, and we wish it to switch to another conformation, β, then we need some mapping procedure which, for a given array of node positions $\vec{r}_\alpha$, returns a set of node positions $\vec{r}_\beta$ corresponding to the *closest valid* configuration of the β topology to the shape defined by $\vec{r}_\alpha$. We can write this simply as:

$$M_{\alpha\beta}\vec{r}_\alpha = \vec{r}_\beta \tag{5.2}$$

Note that if the node position vectors, $\vec{r}_\alpha$ and $\vec{r}_\beta$, are of equal length then $M_{\alpha\beta}$ is a square matrix. In general this is not the case, and $M_{\alpha\beta}$ is some complicated, singular mapping. Furthermore, complications arise from the requirement that

$M_{\alpha\beta}$ only produce *valid* conformations of β. This is defined as a mesh that has no inverted elements and no large local deformations leading to instabilities in subsequent simulation.

Initial attempts at mapping used the same barycentric mapping scheme as in section 4.2, using a β state deformed onto the α rest state to determine the mapping. However, this approach works very poorly when a large number of nodes in β do not lie in (or sufficiently near to) elements of α. Indeed, this was found to be the case for mapping between the apo and ADP•Vi dynein meshes (as can be seen in row A of figure 5.15) whose motor domains and linkers have very different topologies and no overlap. Trying to define $\vec{r}_\beta$ purely in terms of $\vec{r}_\alpha$ created very large distortions in the motor region whenever $\vec{r}_\alpha$ deviated significantly from its rest state.

Instead, a two-stage approach to mapping between these states was taken:

1. Use the barycentric mapping for only those nodes in β that lie within an element of α during determination of the mapping.

2. Map the remaining β nodes by relation to other, already determined nodes (both *alpha* and *beta*).

Step 2 is illustrated in row B of figure 5.15. We repeatedly iterate through the list of nodes until a node is found that is connected to 4 other fully determined nodes ('determined' meaning their positions have already been mapped either in step 1 or in the course of step 2). If the node lies within the tetrahedron formed by these 4 determined nodes, then its position is now also determined. This process continues until all such nodes in β have been determined. The requirement that the node must lie in the tetrahedron formed by 4 other determined nodes is then

gradually relaxed to allow nodes within a threshold distance, d_t:

$$d_t = \sqrt{a_{max}^2 + b_{max}^2 + c_{max}^2} \tag{5.3}$$

where

$$a_{max} = \begin{cases} a - 1 & \text{if } a > 1 \\ 0 & \text{if } 0 \leq a \leq 1 \\ a & \text{if } a < 0 \end{cases} \tag{5.4}$$

where a is one of the barycentric coordinates of the given node in the coordinate system of the tetrahedron. b_{max} and c_{max} are defined likewise. The purpose of equation (5.3) is to penalise distance from the surface of the tetrahedron, rather than distance from its origin. This continues until all the nodes in β have their positions determined in terms of the positions of other nodes (from either α or β). While this mapping can be comparatively slow to calculate, it only needs to be generated once for any given transition. Once calculated, the mapping is very fast and can be incorporated into the simulation. It thus forms part of the preprocessing stage of the simulation.

The ADP•Vi mesh was deformed onto the apo state by creating a constant attractive force between selected nodes on the two meshes. For example, the nodes of the stalk and stem of the former were pushed towards the stalk and stem nodes of the latter. The motor domain and linker, being the problematic region, was mostly allowed to adopt whatever configuration it settled in, although a few control nodes were used to ensure that the ring holes were aligned. The mixed mapping was then generated as detailed above. A similar procedure was used to obtain the mapping between the apo and ADP•Vi states. Row A of figure

5.15 shows these two deformed states.

Two FFEA models for the dynein are simulated, one 'DYNAMIC' and one 'FROZEN'. When switching conformation, the states are swapped, and the node positions mapped. The vdW map for the 'FROZEN' state is also set to non-interacting. Figure 5.16 shows key frames from an example FFEA trajectory with conformational switching, with the active ('DYNAMIC') motor shown in blue, and the inactive ('FROZEN') motor shown in red. In the course of the simulation, the active model fluctuates around, exploring its conformational space. If the probability of switching is p_s per step, then the probability of not switching for N_s steps is $(1 - p_s)^{N_s}$. The number of steps to run in each conformation is therefore:

$$N_s = \frac{\ln(r)}{\ln(1 - p_s)} \tag{5.5}$$

where r is a random variable chosen in the interval $[0, 1]$. Upon switching, the nodes are mapped to the other state, and a new value of N_s is chosen. Loosely speaking, p_s could be interpreted as being a function of the ATP concentration. To test the stability of the mapping, simulations were run with conformational switching at $p_s = 10^{-4}, 10^{-5}, 10^{-6}$ and 10^{-7}. This found that above $p_s = 10^{-5}$ the irregularities in mesh caused by the mapping did not have sufficient time to relax, and became amplified by the frequent switching, often causing the simulation to become unstable after around 6 transitions. This means that the mapping is not suitable if the model is required to undergo conformational changes more frequently than around once a nanosecond. This is therefore not a problem for dynein as the motor operates on time scales far exceeding this.

5.8 Conclusions

FFEA was used to model dynein c and its interaction with the axoneme. The presence of the axoneme was found to increase step length by between 3 and 4 nm, depending on the energy of attraction. The exploration of the MT binding domain was also examined for the apo and ADP•Vi states independently, finding that the latter has a potentially very large search range, particularly around the microtubule (more so than along its length). The number of microtubule binding sites explored by the MT binding domain in each state was also calculated for various interaction energies, and found to be between 10 and 15 in the ADP•Vi state, and around 7 in the apo state. The simulations suggest there is an optimum interaction energy for sampling of MT binding sites (of order $10^{15}\mathrm{Jm}^{-4}$).

Simulations were run with the microtubule doublet's vdW surface patterned at 8 nm intervals to mimic the distribution of microtubule binding sites, and the number of sites explored was calculated once more. Finally, to simulate the motor's power stroke, a mapping algorithm was developed to quickly switch between the apo and ADP•Vi states, and this was tested at several switching probabilities to determine sources of instability.

Further work on this system is discussed in the Outlook section of this thesis (section 7.2).

Chapter 6

Electrostatics

6.1 Introduction

In this chapter we consider the electrostatic force, which plays a crucial role in the way proteins function[132]. The electrostatic potential, created by charges in the protein, directs ligands to active sites on the protein's surface[133] (aiding in protein recognition and binding), affects the way in which proteins orient and aggregate, and how they diffuse in a crowded environment[132].

6.2 Electrostatics in Biological computer models

Simulating the electrostatic interactions so crucial to biological systems is an enormous and very active field of research, and one which is summarised with some difficulty. Broadly speaking, the approaches fall into two categories: those which use *explicit*, atomistically detailed solvents (in which water molecules and salt

ions are simulated explicitly), and those which use a so-called *implicit* solvent, which approximates the solvent as a continuum using some sort of mean field approximation. The former of these approaches is used in Molecular Dynamics simulations, in which electrostatic interactions are described through semiempirical pairwise potentials (for example the TIP3P [134]), and generally give accurate results. The latter approach is also sometimes used in MD simulations[135] but is typically employed in coarse-grained models such as the one-bead model (see section 1.1), can give good results, but requires attention to be paid with regards to its limits of applicability[136].

For simulations using implicit solvent models, the general method is usually to solve Poisson's Equation (PEQ):

$$\nabla.(\epsilon(\underline{r})\nabla\phi(\underline{r})) = -\rho^Q(\underline{r}) \tag{6.1}$$

or more commonly the Poisson-Boltzmann Equation (PBE)[136] (see Appendix D):

$$\epsilon\nabla^2\phi(\underline{r}) + 2c_\infty e\sinh(\frac{e\phi(\underline{r})}{kT}) = -\rho^Q(\underline{r}) \tag{6.2}$$

which takes into account electrostatic charge screening due to salt ions in the solvent. In the above ϵ, ϕ and ρ^Q refer to the electric permittivity, electrostatic potential and charge density at a point $\underline{r}$, respectively. Here, κ is the inverse Debye screening length. In cases where $\frac{e\phi(\underline{r})}{kT}$ is small, the PBE can be linearised by approximating $\sinh[\phi(\underline{r})] \to \phi(\underline{r})$, creating the Linearised Poisson Boltzmann Equation (LPBE):

$$\nabla^2\phi(\underline{r}) + \kappa^2\phi(\underline{r}) = -\frac{\rho^Q}{\epsilon} \tag{6.3}$$

with the inverse Debye screening length:

$$\kappa = \left(\frac{2c_\infty e^2}{\epsilon k T} \right)^{\frac{1}{2}}. \tag{6.4}$$

Typically in biomolecules the electrostatic field is unscreened in the interior, so that the electrostatic field satisfies the PEQ there, but strongly screened in the exterior due to the presence of salt ions. In the case where κ^{-1} is small compared to the dimensions of the molecule the exterior solution can be approximated (see appendix E) by the boundary condition:

$$\frac{d\phi}{dn} = -\kappa\phi \tag{6.5}$$

which assumes a locally flat surface at $\underline{r}$ (low radius of curvature) compared with the screening length κ^{-1}. The PBE is classed as an elliptic partial differential equation, meaning that there is long-range communication across the solution domain. Consequently solving the PBE or PEQ is generally costly, even for moderately sized systems. There are three major classes of numerical scheme that are most often used for solving continuum electrostatics:

- *Finite Difference* (FD): Approximates the continuum as a regular 3-d cartesian grid, and uses finite difference approximations for the derivatives in the governing differential equations. Accuracy can be increased by using a finer grid, or higher order approximations for the derivatives. There exist many iterative FD solvers for electrostatics: Successive over-relaxation, Conjugate gradient, Modified Incomplete Cholesky Conjugate Gradient (MICCG)[137], Algebraic Multigrid and Geometric Multigrid (GMG)[138]. The fastest

solvers tend to be the MICCG and GMG[138].

- *Finite Element Method* (FEM): As described in Chapter 1, section 1.2.1.

- *Boundary Element Method* (BEM): Since the LPBE is a linear equation, it is possible to convert the differential equation to an integral equation, and use Green's 2^{nd} identity to transform the volume integral to a surface one. This reduces the dimension of the calculation, resulting in smaller matrices than those found in FEM. However, these matrices are dense and non-symmetric. While use of highly parallelisable[139] methods such as the Fast Multipole Method (FMM) can produce an algorithm of time-complexity $\mathcal{O}(N)$, the amount of extra computation required is so high that the prefactor (implicit in the $\mathcal{O}(N)$) is constant but very large[140]. BEM also suffers from a severely reduced range of applicability (for example, it cannot handle inhomogeneity in general)[141]. Symmetric matrices *can* be produced through 'Symmetric-Galerkin BEM'[142] but the stability of the scheme is not guaranteed for problems with multiple coupled domains (i.e. the solution is not unique).

Various permutations and combinations of these methods also exist. 'Hybrid' schemes such as BEM-FD [143], BEM-GB[144] and BEM-FEM[145] can produce increased speed or accuracy, depending on the type of electrostatics problem being modelled. Some approaches even mix together implicit and explicit elements, such as a BEM PBE solver with explicit ions[146] (a clever scheme which allows the LPBE to still be used in systems with high ionic strength, for which the non-linear PBE would normally be required[143]). Others speed up calculations by approximating proteins as a collection of spheres and apply a spherical harmonics

expansion to approximate the interactions[147].

6.3 Incorporating Electrostatics into the Mesoscale Model

Numerous approaches to solving the electrostatics problem for the mesoscale model were investigated with varying success. Some test codes were written to investigate how effective a solution they might prove to be for this model. The following subsections list the advantages and disadvantages of these approaches as pertain directly to the model used in this project.

6.3.1 Finite Difference Approach

Method

In many ways, this is the simplest of the three classes of numerical solver investigated. Having discretised the continuum with a regular cartesian grid and applied finite difference approximations to the governing differential equation, the charge and dielectric constant are mapped onto this grid. Charges are "embedded" within specified elements, with their position in that element defined in terms of the three shape functions, s, t and u. The position of a charge q is therefore given by:

$$\underline{r}_q = s_q(\underline{n}_1 - \underline{n}_0) + t_q(\underline{n}_2 - \underline{n}_0) + u_q(\underline{n}_3 - \underline{n}_0) \tag{6.6}$$

for a linear tetrahedral element, where $\underline{n}_{0..2}$ refer to the positions of the element vertices. This definition of position allows the charges to move with the medium in which they are embedded. Mapping the charge onto the grid is carried out by trilinear interpolation, in which a fraction of the charge is allocated to the eight nearest grid nodes according to how close the charge lies to them, such that the total allocated charge is equal to that of the original point charge. Such a technique is used for implicit solvents in some MD simulation packages such as Amber[135] and found to give very accurate potentials for charges separated by at least two grid 'cells'[148].

Advantages

The use of a regular, cartesian mesh in FD makes many forms of optimisation possible. Equally, there exists an enormous body of literature on the topic, with its use in solving problems in electrostatics well established and understood. This is exemplified by the use of FD in packages such as DelPhi and AMBER[135], for continuum electrostatics calculations. Many rapid solution techniques have been developed, particularly as regards the Linearised Poisson-Boltzmann Equation (LPBE), for example Incomplete Modified Cholesky Preconditioned Conjugate Gradient (IMCPCG) solvers[149], and Geometric Multigrid[150]. Geometric multigrid is perhaps the most desirable for this model, as one its key properties is that it always takes the same number of iterations to reach convergence, regardless of grid size N. FD methods are also highly parallelisable. Additionally, adaptive techniques such as *Multilevel Adaptive Technique* (MLAT)[151] or *Fast Adaptive Composite* (FAC)[152] allow for multiple grid resolutions in different areas, saving memory and computing time, and therefore achieving higher accuracy

at lower cost. Furthermore, as the electrostatic potential does not vary greatly from one time step to the next, its recalculation may only need to be performed every 10 time steps (for example). This effectively increases the speed of the solver by an order of magnitude.

Disadvantages

A major disadvantage comes when mapping the dielectric regions to the grid. Unlike the point charges, the dielectric constant applies across the entire space occupied by the element. We must therefore rasterise tetrahedra onto the grid in order to endow cells with the appropriate dielectric constant and screening length. This can be achieved by splitting the tetrahedra into multiple 'slice' planar triangles along one of the cartesian coordinates of the grid, and then rendering each triangle as would be done in a regular 2-d rasteriser. Bresenham's line algorithm[153] can be used to speed this up a little, but it does not offset the main issue: we have many calculations for each triangle, a problem that is only exacerbated by having multiple resolution levels as in the MLAT[151] or FAC[152] methods, where it would be necessary (in this model) to rasterise each element repeatedly for each new level. Furthermore, as some elements will inevitably share some cells with other elements, the process cannot be parallelised in a simple manner. In short, mapping the charges and dielectric regions onto the grids could become the most significant computational cost (due to poor parallel scalability).

A related disadvantage is that the existing finite element mesh does not naturally fit onto the regular grid used in FD, and therefore must be mapped onto one. This means that important areas of interest, such as the protein surface, will

necessarily suffer further discretisation errors. The effect of this is that we need a (sometimes prohibitively) fine mesh to obtain accurate solutions, particularly near the protein surface where the gradient of the potential will be highest. This problem is exacerbated by salt ion screening of the potential in the surrounding solvent. There is an additional problem as regards the very large jump in the dielectric properties at the surface, which can lead to slow convergence from FD solvers[150]. It is also a very expensive method (in terms of memory and computation) even with efficient solvers such as MLAT and FAC. Grids with locally varying degrees of resolution are necessary to make this approach viable, but this results in so-called "hanging nodes" (nodes which are not fully connected) at the boundaries of adjacent regions of different resolution, limiting how quickly the resolution can change across space, and hampering parallelisation attempts.

6.3.2 Finite Element Approach

Method

Salt ions are typically only present in the exterior phase, where they screen the external potential producing an electrostatic field that is confined to the interior and a thin boundary region over the surface. For this problem we can derive the Finite Element formulation for the Poisson equation in the protein interior, with a boundary condition assuming a small but non-zero screening length κ^{-1}. We begin with the PEQ:

$$\nabla.(\epsilon(\underline{r})\nabla\phi(\underline{r})) + \rho^{Q}(\underline{r}) = 0 \quad , \quad \underline{r} \in \Omega \tag{6.7}$$

where Ω is the full protein domain, $\phi(\underline{r})$ is the electrostatic potential and $\epsilon(\underline{r})$ the dielectric constant at some position $\underline{r}$. We formulate a weak approximation to equation (6.7) by integrating over the volume with weight function $\omega(\underline{r})$:

$$\int_{\Omega} \omega \nabla.(\epsilon \nabla \phi)\, dV + \int_{\Omega} \omega \rho^{Q}\, dV = 0, \tag{6.8}$$

to which we apply the Divergence theorem, yielding:

$$\int_{\Gamma} \omega \epsilon \nabla \phi.\underline{n}\, dS - \int_{\Omega} \epsilon \nabla \omega.\nabla \phi\, dV, + \int_{\Omega} \omega \rho^{Q}\, dV = 0 \tag{6.9}$$

where dS refers to an integral over the surface, Γ, of the protein domain and $\underline{n}(\underline{r})$ is the unit normal pointing out of the surface at $\underline{r}$ (only for $\underline{r} \in \Gamma$). In this strongly screened limit the boundary condition at the surface is given by equation (6.5) yielding:

$$\frac{d\phi(\underline{r})}{dn(\underline{r})} = -\kappa\phi(\underline{r}) \quad \Rightarrow \quad \nabla\phi(\underline{r}).\underline{n}(\underline{r}) = -\kappa\phi(\underline{r}) \quad , \quad \underline{r} \in \Gamma. \tag{6.10}$$

Substituting this BC we can therefore write:

$$\int_{\Gamma} w\epsilon\nabla\phi.\underline{n}dS = -\int_{\Gamma} w\kappa\epsilon\phi dS. \tag{6.11}$$

The Finite Element formulation requires that the domain be discretised into a number of elements, with the potential defined only at the nodes of these elements, and interpolation between the nodes carried out via element shape functions. We therefore let the electrostatic potential be described by a superposition of the

element shape functions, ψ_i:

$$\phi(\underline{r}) \rightarrow \sum_{i}^{N_n} \phi_i \psi_i \tag{6.12}$$

where N_n is the number of nodes and ϕ_i is the electrostatic potential at node i, and such that $\int_\Omega \psi_i . \psi_j dV = 0 \quad \forall i, j$ on different elements. In the Galerkin formulation we choose the weight function ω to be a shape function ψ_j: $\omega(\underline{r}) \rightarrow \psi_j$: of the problem:

$$\int_\Omega \epsilon \left(\nabla\psi_j . \nabla\psi_i\right) \phi_i dV - \int_\Omega \psi_j \rho_i^Q \psi_i dV + \int_\Gamma \kappa\epsilon \psi_j \psi_i \phi_i dS = 0 \tag{6.13}$$

where we have discretised $\rho^Q(\underline{r}) \rightarrow \rho_i^Q \psi_i$ using the same shape function. Equation (6.13) can then be written as a matrix equation:

$$\left(K + M_\Gamma\right) \underline{\phi} = \underline{\rho}^Q \tag{6.14}$$

which can be solved for $\underline{\phi}$ to find the electrostatic potential at each node.

Advantages

The finite element mesh for the protein is ready made and in use by the program. Many of the operations required in solving the electrostatics FEM are already being carried out every time step (such as the calculation of inverse Jacobians, shape function derivatives, diffusion matrices etc.) so solving the electrostatics by this method seems natural and efficient, both computationally and memory-wise. It avoids entirely the problems associated with the regular grid in the finite difference approach, as this mesh conforms to the existing protein mesh surface

perfectly, and therefore all calculations of the potential will be "exact" for the given mesh (no further discretisation errors).

Disadvantages

This solution method applies only to the case where κ^{-1} is very small compared with the intermolecular distance. However, this approximation does not always hold and electrostatic interactions will, rather importantly, extend beyond the boundary of the protein. In the case of multiple interacting proteins, this produces interactions with nearby proteins. This could be included with the FEM formulation by extending the domain to the *entire domain* containing all proteins and solvent (which must be meshed). However, proteins are not stationary. Remeshing between arbitrarily positioned objects at arbitrary orientations in constant movement makes this method impractical. While 3-d meshing codes exist, the computational time required to produce good quality meshes over large complex 3-d domains hundreds of times a second is extremely large.

As noted earlier, the typical salt concentrations found in biological systems such as cells are such that electrostatic screening is relatively high (giving Debye screening lengths of the order of 1 or 2 nanometres). This is small on the length scales being considered in our coarse grained approach. Therefore an approach for the cases of intermediate screening is to mesh an exterior "shell" of high dielectric elements around the protein that extends to around 2 or 3 Debye lengths from the surface (the distance at which the potential will be effectively totally screened). When two or more proteins approach each other, the volume in which the meshes overlap is then mathematically constrained to have the same solution (of the potential). This idea was abandoned because it was unclear how accurate such

a solution would become when the proteins were very close together (the point at which electrostatics becomes most important) and whether or not the solution was guaranteed to converge in complicated systems.

6.3.3 Boundary Element Approach

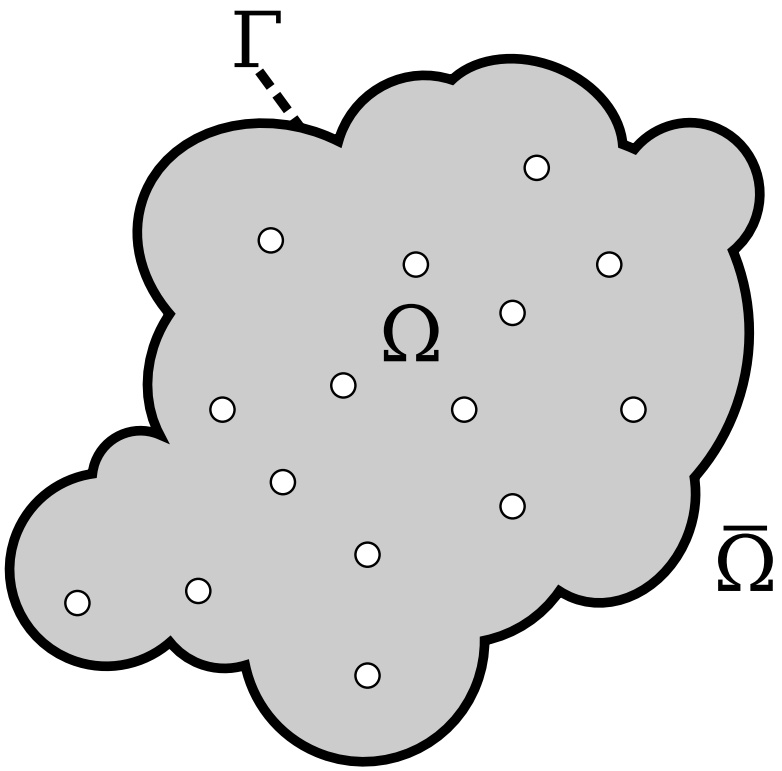


Figure 6.1: The single protein BEM case, illustrated here in 2-d for simplicity purposes. The solute (protein) is represented by the homogeneous region Ω of dielectric constant ϵ^{Ω}. The solvent is represented by the region $\bar{\Omega}$ of dielectric constant $\epsilon^{\overline{\Omega}}$, $\epsilon^{\overline{\Omega}} > \epsilon^{\Omega}$. The boundary (surface of the protein) is termed Γ.

An alternative approach that removes the need to construct an external mesh is to use a boundary method. We will first consider the case of a single molecule. This derivation follows essentially the same approach to that taken by McCam-

mon *et al.* for atomistic structures[154], including a continuous charge distribution for the FFEA case. We first consider the system shown in figure 6.1, in which a protein is modelled as a closed, homogeneous region, Ω, of low dielectric constant, ϵ^{Ω}, immersed in an infinite, homogeneous region, $\bar{\Omega}$, of high dielectric constant, $\epsilon^{\overline{\Omega}}$, both regions meeting at the boundary Γ. We assume that the charge distribution $\rho^{Q}(\underline{r})$ is confined to the interior of the protein. As most real biological solvents have significant salt concentrations, the electrostatic potential in the exterior region is assumed to be screened, with a Debye screening length of κ^{-1}. The interior region is assumed to have no significant screening effects on the length scale of the protein. We can therefore model the potential, ϕ, at any point, $\underline{r}$, in our system as follows:

$$\nabla^2\phi(\underline{r})^{\Omega} = -\frac{1}{\epsilon^{\Omega}}\rho^{Q}(\underline{r}) \quad , \quad \underline{r} \in \Omega \tag{6.15}$$

$$\nabla^2\phi(\underline{r})^{\overline{\Omega}} = \kappa^2\phi(\underline{r})^{\overline{\Omega}} \quad , \quad \underline{r} \in \bar{\Omega} \tag{6.16}$$

where we have modelled the interior region using the PEQ and the exterior region using the LPBE. Since we assume that there are no surface charges, the following BC apply at the surface:

$$\phi(\underline{r})^{\Omega} = \phi(\underline{r})^{\overline{\Omega}} \quad , \quad r \in \Gamma \tag{6.17}$$

$$\epsilon^{\Omega}\frac{\partial\phi^{\Omega}}{\partial n_{\Omega}} = \epsilon^{\overline{\Omega}}\frac{\partial\phi^{\overline{\Omega}}}{\partial n_{\Omega}} \quad , \quad \underline{r} \in \Gamma \tag{6.18}$$

where $\underline{n}_{\Omega}$ denotes an *outward* unit normal to the protein surface, $\frac{\partial\phi}{\partial n} \equiv \nabla\phi.\underline{n}$.

The first of these BCs enforces continuity of the potential on the surface, and the second BC enforces continuity of the normal component of the electric displacement.

We now derive a boundary integral formulation of this problem for the two regions, starting with the interior region, Ω. Start with Green's second identity:

$$\int_\Omega \left(\psi\nabla^2\phi - \phi\nabla^2\psi\right) dV = \oint_\Gamma \left(\psi\frac{\partial\phi}{\partial n} - \phi\frac{\partial\psi}{\partial n}\right) dS \tag{6.19}$$

where ϕ and ψ are any twice differentiable scalar functions, and the dV and dS denote an integral over the volume Ω and the surface Γ respectively. We shall let ϕ be the electrostatic potential (as before) and let ψ be G, where G is the fundamental solution of the Laplace equation or Green's function:

$$[-\nabla_j^2]G(\underline{x}_i - \underline{x}_j) = \delta(\underline{x}_i - \underline{x}_j). \tag{6.20}$$

From here on, we introduce a convenient compact notation, in which f_i denotes $f(x_i)$, and f_{ij} denotes $f(x_i - x_j)$. Using this concise notation, equation (6.20) becomes $[-\nabla_j^2]G_{ij} = \delta_{ij}$, with

$$G_{ij} = \frac{1}{4\pi|\underline{x}_i - \underline{x}_j|}. \tag{6.21}$$

Recall that, as always, i and j denote the points $\underline{x}_i$ and $\underline{x}_j$; they are not to be confused with indices labelling vector components. Using (6.21) in Green's second identity yields:

$$\int_\Omega \left(G_{ij}\nabla^2\phi_j + \phi_j\delta_{ij}\right) dV_j = \oint_\Gamma \left(G_{ij}\frac{\partial\phi_j}{\partial n_\Omega} - \phi_j\frac{\partial G_{ij}}{\partial n_\Omega}\right) dS_j \tag{6.22}$$

and substituting our expression for $\nabla^2\phi_j$:

$$\int_\Omega \left(-\frac{1}{\epsilon^\Omega} \sum_k^{n_q} G_{ij} q_k \delta_{jk} + \phi_j \delta_{ij} \right) dV_j = \oint_\Gamma \left(G_{ij} \frac{\partial \phi_j}{\partial n_\Omega} - \phi_j \frac{\partial G_{ij}}{\partial n_\Omega} \right) dS_j. \tag{6.23}$$

Using the sifting property of δ we then obtain our boundary integral formulation for the interior problem:

$$\phi_i^\Omega = \oint_\Gamma \left(G_{ij} \frac{\partial \phi_j}{\partial n_\Omega} - \phi_j \frac{\partial G_{ij}}{\partial n_\Omega} \right) dS_j + \frac{1}{\epsilon^\Omega} \sum_k^{n_q} G_{ik} q_k \quad , \quad i \in \Omega \tag{6.24}$$

where j is some arbitrary point on the surface Γ.

We perform essentially the same trick in order to obtain the boundary integral for the exterior region, except this time we let ψ be u (instead of G) in equation (6.19), where u is the fundamental solution of the LPBE:

$$\nabla^2 u_{ij} - \kappa^2 u_{ij} = -\delta_{ij}. \tag{6.25}$$

Using similar notation as before, with

$$u_{ij} \equiv u(\underline{x}_i, \underline{x}_j) = \frac{e^{-\kappa|\underline{x}_i - \underline{x}_j|}}{4\pi|\underline{x}_i - \underline{x}_j|}. \tag{6.26}$$

Since the field decays sufficiently rapidly at large distance, we can ignore the contributions from the surface integral at infinity and so using (6.25) in (6.19) yields:

$$\int_\Omega \left[u_{ij}\kappa^2\phi_j - \phi_j(\kappa^2 u_{ij} - \delta_{ij}) \right] dV_j = \oint_\Gamma \left(u_{ij} \frac{\partial \phi_j}{\partial n_{\bar{\Omega}}} - \phi_j \frac{\partial u_{ij}}{\partial n_{\bar{\Omega}}} \right) dS_j. \tag{6.27}$$

Again, using the sifting property of δ and cancelling terms leads to the boundary integral equation for the exterior problem:

$$\phi_i^{\overline{\Omega}} = \oint_\Gamma \left(\phi_j \frac{\partial u_{ij}}{\partial n_\Omega} - u_{ij} \frac{\partial \phi_j}{\partial n_\Omega} \right) dS_j \quad , \quad i \in \bar{\Omega}. \tag{6.28}$$

Note that we have reversed the sign of the "outward" normal, $\underline{n}_{\bar{\Omega}} = -\underline{n}_\Omega$, since the normal pointing "out" of the exterior region is in fact pointing into interior region Ω.

We now have boundary integral formulations for both Ω and $\bar{\Omega}$. However, we wish to couple these two regions, and therefore we seek the solution at the boundary between the two regions, Γ. We do this by allowing the point i to approach the boundary Γ:

$$c_i \phi_i^\Omega = \oint_\Gamma^{(CPV)} \left(G_{ij} \frac{\partial \phi_j}{\partial n_\Omega} - \phi_j \frac{\partial G_{ij}}{\partial n_\Omega} \right) dS_j + \frac{1}{\epsilon^\Omega} \int_\Omega G_{ik} \rho_k^Q dV_k \quad , \quad i \to \Gamma \tag{6.29}$$

$$c_i \phi_i^{\overline{\Omega}} = \oint_\Gamma^{(CPV)} \left(\phi_j \frac{\partial u_{ij}}{\partial n_\Omega} - u_{ij} \frac{\partial \phi_j}{\partial n_\Omega} \right) dS_j \quad , \quad i \to \Gamma. \tag{6.30}$$

This introduces a factor c_i to both equations (to account for the contribution of each region when i lies exactly on the boundary). Note also that these integral signs now refer to the Cauchy Principal Value (CPV) integral, as the regular integrals have singular components as $i \to j$. For a locally smooth (flat) surface, a point lying exactly at the boundary will effectively lie half in either region. Assuming such a surface, we set the value of the prefactors $c_i = \frac{1}{2}$. By introducing the boundary conditions (equations (6.17) and (6.18)), we can determine the

potential at any point in the system from two surface functions, ϕ^{Γ} and J^{Γ}:

$$\phi^{\Omega} = \phi^{\overline{\Omega}} = \phi^{\Gamma} \tag{6.31}$$

$$\frac{\epsilon^{\Omega}}{\epsilon^{\overline{\Omega}}} \frac{\partial \phi^{\Omega}}{\partial n_{\Omega}} = \frac{\partial \phi^{\overline{\Omega}}}{\partial n_{\Omega}} = J^{\Gamma}. \tag{6.32}$$

So that the interior equation now becomes:

$$\left[\oint_{\Gamma}^{(CPV)} \left(\frac{\partial G_{ij}}{\partial n_{\Omega}} + \frac{1}{2}\delta_{ij} \right) dS_j \right] \phi_i^{\Gamma} + \left[-\frac{\epsilon^{\overline{\Omega}}}{\epsilon^{\Omega}} \oint_{\Gamma}^{(CPV)} G_{ij} dS_j \right] J_i^{\Gamma} = \frac{1}{\epsilon^{\Omega}} \int_{\Omega} G_{ik} \rho_k^Q dV_k \tag{6.33}$$

and the exterior equation now becomes:

$$\left[\oint_{\Gamma}^{(CPV)} \left(\frac{\partial u_{ij}}{\partial n_{\Omega}} - \frac{1}{2}\delta_{ij} \right) dS_j \right] \phi_i^{\Gamma} + \left[- \oint_{\Gamma}^{(CPV)} u_{ij} dS_j \right] J_i^{\Gamma} = 0. \tag{6.34}$$

Thus we have two surface integral equations governing the unknown surface. We now introduce a discretisation of the boundary for which we can use the existing triangular surface mesh of the protein. The functions ϕ^{Γ} and J^{Γ} are represented as:

$$\phi^{\Gamma}(x_i) = \sum_{j=1}^{N^{\Gamma}} \phi_j \psi_j^{\Gamma}(xi) \tag{6.35}$$

and

$$J^{\Gamma}(x_i) = \sum_{j=1}^{N^{\Gamma}} J_j \psi_j^{\Gamma}(xi) \tag{6.36}$$

where ψ_j are the surface shape functions. This reduces the integral equations

(6.33) and (6.34) to the matrix equation (6.37):

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} \underline{\phi}^{\Gamma} \\ \underline{J}^{\Gamma} \end{pmatrix} = \begin{pmatrix} \underline{Q} \\ \underline{0} \end{pmatrix} \tag{6.37}$$

where we have converted our four integrals into matrices through a discretisation of the boundary Γ, into $N_{\triangle}$ triangular elements.

$$A_{ij} = \int_{\triangle(i)}^{(CPV)} \frac{\partial G_{ij}}{\partial n_{\triangle(i)}} dS_{\triangle(i)} + \frac{1}{2} I \tag{6.38}$$

$$B_{ij} = \int_{\triangle(i)}^{(CPV)} -\frac{\epsilon^{\overline{\Omega}}}{\epsilon^{\Omega}} G_{ij} dS_{\triangle(j)} \tag{6.39}$$

$$C_{ij} = \int_{\triangle(i)}^{(CPV)} \frac{\partial u_{ij}}{\partial n_{\triangle(i)}} dS_{\triangle(i)} - \frac{1}{2} I \tag{6.40}$$

$$D_{ij} = \int_{\triangle(i)}^{(CPV)} -u_{ij} dS_{\triangle(i)} \tag{6.41}$$

where I is the identity matrix and $\triangle(i)$ indicates that the integral is over every surface element $\triangle \in \Gamma$ for which $i \in \triangle$. $\underline{Q}$ is given by:

$$Q_i = \int_{\Omega} G_{ik} \rho_k^Q dV_k. \tag{6.42}$$

Note that in the case of n_q distinct point charges this would become the sum $Q_i = \sum_k^{n_q} G_{ik} q_k$.

Generalising this to the case of $N_p > 1$ interacting molecules is relatively simple. We now have many, closed, independent, low dielectric regions immersed in an infinite region of high dielectric constant (as before). We therefore have an equation (6.33) for each of the N_p separate proteins, while the surface integral in equation (6.34) becomes the sum of integrals over each of the N_p proteins so

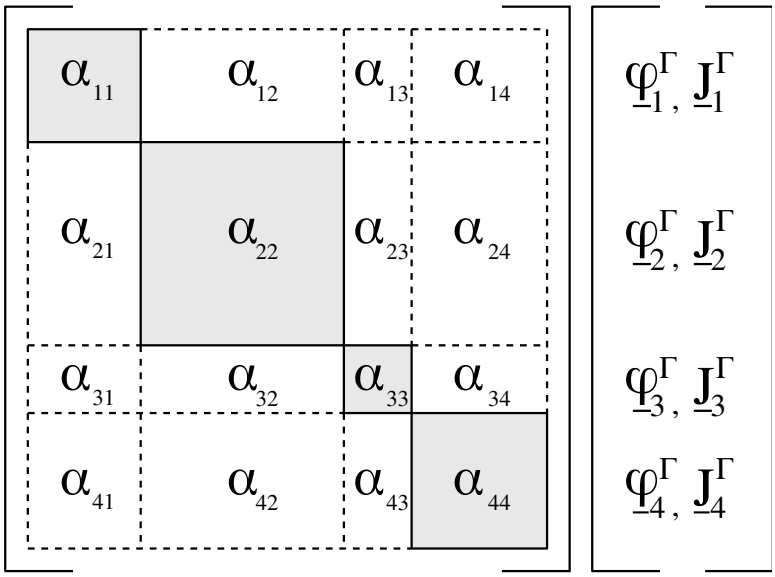


Figure 6.2: An example BEM matrix structure for a problem involving four interacting proteins. α refers to any of matrices A, B, C and D. α_{ij} indicates a portion of the matrix describing the interaction between nodes on the surface of some protein i, with nodes on the surface of some protein j. For the A and B matrices, the interaction block will only be non-zero for the diagonal case $i = j$. For the C and D matrices, all interaction blocks are fully dense. The size of each on-diagonal self-interaction block is $N \times N$ where N is the number of surface nodes in that protein.

that:

$$\oint_{\Gamma} \cdots \equiv \sum_{p=1}^{N_p} \oint_{\Gamma_p} \cdots \tag{6.43}$$

Each of the matrices A, B, C and D can be constructed in the form:

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots \\ A_{21} & A_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{6.44}$$

where A_{nm} represents the contributions from the surface integrals over protein m to points on the surface of protein n. The A and B matrices describe interactions between nodes through the protein interior. As the protein interior regions are unconnected, these matrices will have no 'knowledge' of interactions between nodes on different proteins. This means the full A and B matrices will be block diagonal, e.g.

$$\begin{pmatrix} A_{11} & 0 & 0 & 0 & \cdots \\ 0 & A_{22} & 0 & 0 & \cdots \\ 0 & 0 & A_{33} & 0 & \cdots \\ 0 & 0 & 0 & A_{44} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{6.45}$$

where $A_{11..N_pN_p}$ are dense, square, asymmetric matrices describing the self-interaction of a protein's surface with itself, through its interior. The C and D matrices, on the other hand, represent interactions through the solvent and as such have knowledge of every surface-surface interaction in the system. They are therefore very large, fully dense and asymmetric: extremely bad properties for a problem we wish to solve rapidly and efficiently. However, for cases where the screening

length is small the contributions from distant elements will be negligible so that sparsity can be recovered. Figure 6.2 gives an example of the meaning of different parts of a typical matrix for a problem involving four interacting proteins.

Evaluating the BEM

The main difficulty with the BEM comes from having to deal with the evaluation of integrals since the functions G_{ij} and u_{ij} are singular at $x_i = x_j$. Thus the integrals can be divided into 3 classes:

Points i and j lie on...	Integral type	Course of action
different (far apart) elements	Regular	Gaussian Quadrature
the same element	Singular	Coordinate transformation
different (very nearby) elements	Near-singular	Many proposed solutions[155; 156].

Within these broad cases there are further distinctions. In the case of the 'regular' integral, the integrand is finite everywhere within the limits of integration, and therefore an n_{gp}-point Gaussian Quadrature rule can be used to evaluate the integral. The value of n_{gp}, however, varies depending on how far apart points i and j are (at their closest positions). In the 'singular' integral case, the solution method naturally depends on the strength of the singularity. The functions u_{ij} and G_{ij} have $\frac{1}{r}$ singularities and can be evaluated by transforming to polar coordinates, leading to a 1-d integral that can be solved with Gaussian Quadrature (see below). The derivatives in A and C contain $\frac{1}{r^2}$ singularities, termed *hypersingular*, and require a different approach.

For an element with constant shape functions, the surface node (and therefore

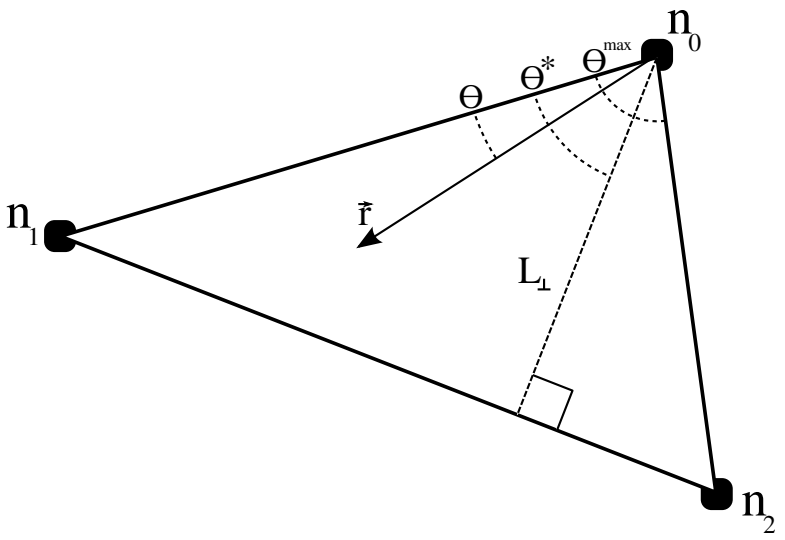


Figure 6.3: Diagram showing quantities used in an integral of a weakly singular function over a general triangle, with singularity at node n_0. In the course of the integral, the vector $\vec{r}$ sweeps through the region $\theta = 0 \,..\, \theta^{max}$ and $r = 0 \,..\, R(\theta)$. $L_\perp$ is the altitude of the triangle and θ^* is the angle between the vector $\vec{n}_1 - \vec{n}_2$ and the altitude vector of the triangle. The expressions for the quantities in this diagram are given in equation (6.50).

the singular portion of the integrand) lies at the centroid of the surface element. In order to evaluate the $\frac{1}{r}$ singular integral present in equation (6.41), the triangular element is split into three sub triangles, with the central singularity on one of these nodes. Then for each subtriangle $\triangle'$ we have:

$$-\int_{\triangle'} \frac{e^{-\kappa r}}{4\pi r} dS_{\triangle'} \Rightarrow -\frac{1}{4\pi} \int_0^{\theta^{max}} \int_0^{R(\theta)} \frac{e^{-\kappa r}}{4\pi r} r dr d\theta \tag{6.46}$$

where $r \equiv |\underline{r}|$ and $\underline{r} \equiv \underline{x}_i - \underline{x}_j$, the vector between point i and point j. Transforming to polar coordinates removes the $\frac{1}{r}$ singularity. Integrating with respect to r yields:

$$\frac{1}{4\pi\kappa} \int_0^{\theta^{max}} e^{-\kappa R(\theta)} d\theta \tag{6.47}$$

with

$$R(\theta) = \frac{L_\perp}{\cos(\theta - \theta^*)} \tag{6.48}$$

where $R(\theta)$ is the distance from the singular node to the opposite side of the triangle for some angle θ. We are now left with a non-singular 1-d integral which can be solved with Gaussian Quadrature:

$$\frac{\theta^{max}}{8\pi\kappa} \sum_{i=1}^{n_{gp}} w_i \exp\left(\frac{-\kappa L_\perp}{\cos(\frac{\theta^{max}}{2}(\xi_i + 1) - \theta^*)}\right) \tag{6.49}$$

where w_i are the weights and ξ_i the Gauss points for this n_{gp}-point rule, and

$$\theta^{max} = \cos^{-1}\left[\frac{\underline{r}_{01}.\underline{r}_{02}}{|\underline{r}_{01}||\underline{r}_{02}|}\right] \tag{6.50}$$

$$L_{01} = |\underline{r}_{01}| \tag{6.51}$$

$$L_{\perp} = \left|\underline{r}_{01} - (\underline{r}_{01}.\underline{r}_{12})\frac{\underline{r}_{12}}{|\underline{r}_{12}|^2}\right| \tag{6.52}$$

$$\theta^{*} = \cos^{-1}\left(\frac{L_{\perp}}{L_{01}}\right) \tag{6.53}$$

with $\underline{r}_{01} \equiv \underline{n}_1 - \underline{n}_0$ and $\underline{r}_{02} \equiv \underline{n}_2 - \underline{n}_0$. The meanings of $L_{\perp}$, θ, θ^* and θ^{max} are shown, for a general triangle, in figure 6.3.

In order to evaluate the integral found in (6.40) we transform to spherical polar coordinates, in which the gradient ∇u term becomes:

$$\nabla u = \hat{r}\frac{\partial u}{\partial r} + \hat{\theta}\frac{1}{r}\frac{\partial u}{\partial \theta} + \hat{\phi}\frac{1}{rsin(\theta)}\frac{\partial u}{\partial \phi} \tag{6.54}$$

$$= -\frac{e^{-\kappa r}}{4\pi}\left[\frac{1}{r^2} + \frac{\kappa}{r}\right]\hat{r}. \tag{6.55}$$

On a planar element, with singularity at the centroid, the vector between i and j lies always in the plane, and therefore $\nabla u.\hat{n}$ must be zero everywhere on the element. Hence, in the singular case, the 'flux' integral is zero (for constant elements).

The regular integrals are comparatively simple to deal with, as the integrands require no alterations to account for singularities. Special Gaussian Quadrature rules for triangles are used to evaluate these. For the non-singular case of the

integral found in equation (6.41):

$$-\int_{\triangle} \frac{e^{-\kappa r}}{4\pi r} dS_{\triangle} \approx -A_{\triangle} \sum_{i=1}^{n_{gp}} w_i \frac{e^{-\kappa r'}}{4\pi r'} \tag{6.56}$$

where $r \equiv |\underline{x}_{src} - \underline{x}_{obs}|$, x_{src} and x_{obs} are the positions of the source and observation points respectively, n_{gp} is the number of Gauss points, w_i is the weight of each point, $A_{\triangle}$ is the area of the triangle, and $r' \equiv |\underline{x}(\xi_1^i, \xi_2^i, \xi_3^i) - \underline{x}_{obs}|$ where the ξ are the Gauss points given in barycentric coordinates. And similarly for the non-singular case of the integral found in equation (6.40):

$$\int_{\triangle} \frac{\partial u(r)}{\partial n} dS_{\triangle} \approx -A_{\triangle} \sum_{i=1}^{n_{gp}} w_i \frac{e^{-\kappa r'}}{4\pi} \left[\frac{1}{r'^2} + \frac{\kappa}{r'}\right] \hat{r'}.\hat{n_{\triangle}}. \tag{6.57}$$

Advantages

As with the FEM, the BEM has an advantage over the FD approach in that it uses the surface of the FEM mesh already represented in the code, rather than introducing further discretisation errors (by using a grid). It can also be less expensive (computationally) than the FEM for systems with a low surface area to volume ratio, since only the surface faces need to be considered, and not the entire volume. Furthermore, it has one great advantage over both FD and FEM approaches in that domains of infinite extent (such as the exterior of the proteins) can be treated exactly in the integrals, rather than having to choose a "large enough" domain to mesh or grid over, or approximative boundary conditions, as is the case with FD and FEM[141].

Disadvantages

Despite having smaller resultant matrices to deal with, the matrices generated by the BEM are fully dense and nonsymmetric[141]. This can be contrasted with the very sparse, symmetric, positive definite matrices generated by the FD and FE methods. Evidently, this raises severe memory and computational issues, particularly as the system becomes larger.

6.4 Discussion: Coupled FEM-BEM Method

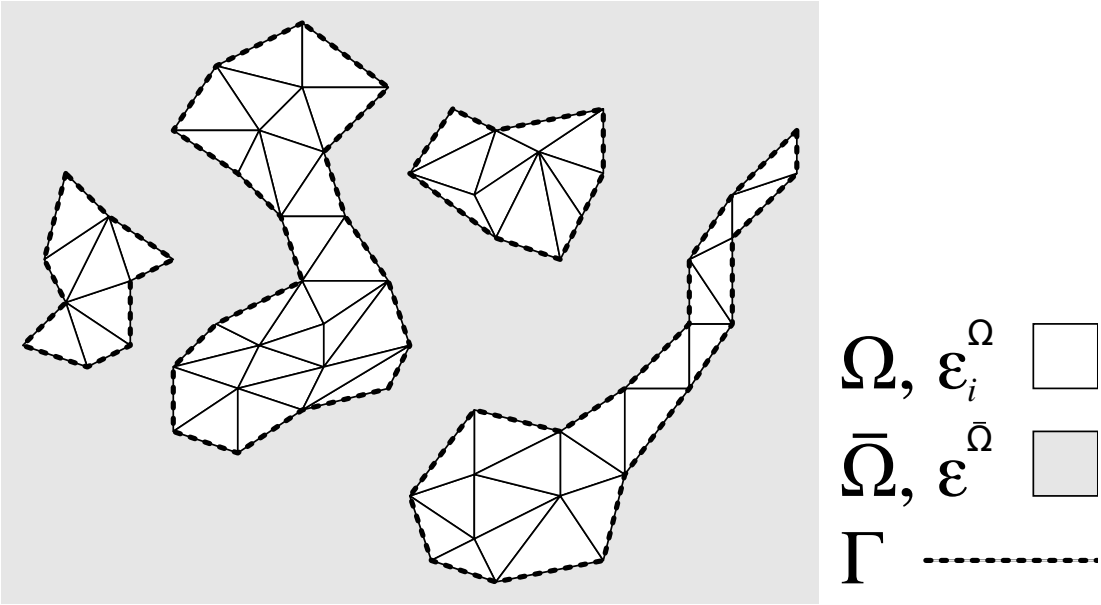


Figure 6.4: A hybrid FEM-BEM system shown in 2-d for simplicity. Multiple proteins Ω of low dielectric constant ϵ_i^{Ω} floating in an exterior solvent $\bar{\Omega}$ of high dielectric constant $\epsilon^{\overline{\Omega}}$. The inhomogeneous protein interiors are discretised by Finite Element meshes, in which each element i may have a different dielectric constant. The solvent is a homogeneous region described with a single dielectric constant. BEM is used to solve the exterior region, and FEM the interior. Coupling occurs through boundary conditions defined at the surfaces, Γ.

As we have seen, the methods discussed above each have at least one significant disadvantage for use with the FFEA model. The main disadvantages are: the non-conforming mesh in FD (very fine mesh required for accuracy), the need for frequent remeshing of 3-d space in FEM (extremely slow and prone to producing 'bad elements') and the construction of large dense matrices in BEM which, additionally, can only deal efficiently with homogeneous domains. In order to overcome these severe limitations, we therefore require a method which:

1. Uses only the existing protein meshes (i.e conforming boundary, but no meshing/gridding of the exterior)

2. Can handle the inhomogeneity of proteins (i.e. spatially varying dielectric constant)

3. Results in sparse matrices, whose size and construction cost grows as the number of nodes in the system (not as the system volume)

Requirement 1 eliminates FD and FEM, leaving only BEM as suitable. Requirements 2 and 3, however, are not satisfiable by a BEM approach, which assumes a uniform dielectric constant and produces dense matrices. In order to satisfy all three requirements we therefore consider a 'hybrid' approach as follows:

Firstly, note that only the interior regions of proteins are inhomogeneous, whereas we assume (by virtue of the LPBE) that the exterior (solvent) is a homogeneous region of high uniform dielectric constant. Secondly, note that the exterior has a high salt concentration resulting in heavily screened electrostatic interactions, as opposed to the low dielectric protein interiors in which electrostatic interactions are (comparatively) very long range. The interior BEM problem therefore results in fully dense matrices (every node may interact with every

other node) whereas the exterior BEM problem can be made extremely sparse by cutting out all interactions between nodes more than a few Debye lengths apart.

All three requirements may then be satisfied by a method which solves the interior problem via FEM, the exterior via BEM, and couples the two methods at the dielectric boundary (protein surface).

Note that the Fast Multipole Method (FMM) is generally used to speed up the matrix vector multiplication steps in BEM solvers by grouping together distant sources, but in this case the electrostatic screening simply cuts out distant sources altogether. This allows the algorithm to achieve similar time complexity, without the huge prefactor found in FMM[157].

The matrix construction phase of BEM requires the calculation of many integrals between a surface node and a surface element. For surface elements with linear shape functions, the nodes lie on the vertices of the triangular elements, and therefore the value of the matrix element corresponding to this node will depend on the results of integrals on every element of which it is a member. This introduces a high interconnectivity which is extremely difficult to parallelise efficiently, as a complex algorithm is required to subdivide the surface in a way that minimises the cross-dependencies and allows the work to be shared across processors. In order to avoid this problem, we use elements with piecewise constant shape functions (that may be subdivided into many smaller elements if a high level of accuracy is required). The surface nodes now lie at the centroid of the surface elements, meaning that the matrix element corresponding to this node depends only on integrals over this one element. This has the enormous benefit of making each node completely independent of integrals over elements other than the one of which it is a member, and therefore the elements may be

workshared across all processors, leading to high parallel efficiency in the matrix construction phase.

Having seen the highly desirable properties of this hybrid approach, there now remains the task of coupling the two numerical schemes. The approach used in this project begins by noting that the boundary condition introduced in equation (6.32) (for J^{Γ}, effectively a flux term) is of the same form as the surface flux term found in the FEM formulation in equation (6.9). We therefore intuitively seek to couple the FEM and BEM regions through this surface term.

The Galerkin formulation of the interior problem is:

$$\int_{\Gamma} \epsilon_i^{\Omega} \psi_j \nabla\phi.\underline{n}\, dS - \int_{\Omega} \epsilon_i^{\Omega} (\nabla\psi_j.\nabla\psi_i)\phi_i\, dV + \int_{\Omega} \psi_j \rho_i^{Q} \psi_i dV = 0 \tag{6.58}$$

and applying the BC from equation (6.32), we have:

$$\int_{\Gamma} \epsilon_i^{\Omega} \psi_j \nabla\phi.\underline{n}\, dS = \sum_{\triangle(i)} \epsilon^{\overline{\Omega}} J^{\Gamma}_{\triangle(i)} \int_{\triangle(i)} \psi_j\, dS_{\triangle(i)} \tag{6.59}$$

where the flux at the node i is given by the sum of the fluxes on the surface elements that meet at that point, $\triangle(i)$. For the case of piecewise constant elements, J^{Γ} is constant across each element so we can take it outside the integral, and are left simply with an integration of the shape function over the surface element which yields a third of the area of the triangle (if node j is a member of triangle $\triangle(i)$). The FEM equation (in this hybrid scheme) is therefore:

$$\sum_{\triangle(i)} \frac{\delta_{j\triangle(i)} \epsilon^{\overline{\Omega}} A_{\triangle(i)}}{3} J^{\Gamma}_{\triangle(i)} - \int_{\Omega} \epsilon_i^{\Omega} (\nabla\psi_j.\nabla\psi_i)\phi_i\, dV + \int_{\Omega} \psi_j \rho_i^{Q} \psi_i dV = 0 \tag{6.60}$$

where $A_\triangle$ is the area of the surface element $\triangle(i)$, and $\delta_{j\triangle(i)}$ is equal to unity if node j is on triangle $\triangle(i)$ and zero otherwise. In our matrix representation, equation (6.14) is now:

$$K\underline{\phi} + E\underline{J}^\Gamma = \underline{q} \tag{6.61}$$

where E is the matrix given by the first term in equation (6.60), for which only rows corresponding to surface elements contain non-zero entries. A helpful property of E is that it is constant and only needs to be calculated once in the initialisation stages of the simulation. Thus by dividing the elements into interior (Ω) and surface (Γ) nodes, we can write the linear system in the form:

$$\begin{pmatrix} K_{\Omega\Omega} & K_{\Omega\Gamma} & 0 \\ K_{\Gamma\Omega} & K_{\Gamma\Gamma} & E \\ 0 & C & D \end{pmatrix} \begin{pmatrix} \underline{\phi}^\Omega \\ \underline{\phi}^\Gamma \\ \underline{J}^\Gamma \end{pmatrix} = \begin{pmatrix} \underline{q}^\Omega \\ \underline{q}^\Gamma \\ \underline{0} \end{pmatrix} \tag{6.62}$$

where the matrices C and D come from the BEM solution of the external LPBE given in equations (6.40) and (6.41). Solving this system of equations can be carried out using non-symmetric matrix solvers such as the Biconjugate Gradient Stabilized (BiCGSTAB) method with an appropriate pre-conditioner[158]. Rather than directly solving for the fully constructed matrix, it may be more efficient to use an iterative approach in which we solve the system through alternate application of BiCGSTAB for the nonsymmetric BEM region and regular Conjugate Gradient for the symmetric positive definite matrices in the FEM regions.

In order that the solution to equation (6.62) be rapid, it is important to avoid dense matrices such as those found in a general BEM formulation. We therefore take advantage of the electrostatic screening in order to cut out interactions more

than a few Debye-lengths in separation. This allows us to greatly sparsify the matrices C and D, cutting solution times greatly. Additionally, C and D are diagonally dominant, leading to better convergence rates.

As illustrated in figure 3.4, sparsification of the matrices takes place by using a large cubic grid stacks (implemented internally as linked lists). The position of each node in turn is tested to see which cell it lies in. Then the data object representing that node is "pushed" onto the appropriate stack. When it is time to perform the BEM integrals for each node, it is necessary only to check the current list (in which the node lies) and the surrounding 26 cells (for a 1.5 Debye-length cell length). All interactions with nodes in cells further away are known to be zero (due to screening) so there is no need to perform costly integrals or create unnecessary fill-in for the C and D matrices. It was found that 1.5 Debye-lengths for a cell side length in the lookup-table gave the best compromise between memory cost (storing N^3 pointers for the stacks) and computational cost (processing more near-zero interaction pairs). Note that only one pointer for each cell needs to be stored, pointing to the topmost object in the stack, and one pointer per node (pointing to the next object in the list after itself), so the memory required by this nearest neighbour lookup grid is actually reasonably small, even for spatially large systems.

6.5 Boundary Coupling through flux

A simple iterative scheme for solving equation 6.62 is as follows:

1. Choose initial 'guess' at flux on all surfaces.

2.

$$K\underline{\phi} + E\underline{J}^{\Gamma} = M\underline{q}. \tag{6.63}$$

Solve the interior electrostatics in each protein domain, using this flux $\underline{J}$, in order to obtain the surface potential of the system, $\underline{\phi}^{\Gamma}$.

3.

$$D\underline{J}^{\Gamma} = -C\underline{\phi}^{\Gamma}. \tag{6.64}$$

Solve the exterior electrostatics for the system, using the surface potential $\underline{\phi}^{\Gamma}$ in order to obtain the flux $\underline{J}$.

4. Repeat steps 2 and 3 until convergence.

However, in practice this scheme does not converge. The problem is that the interior PEQ with a Neumann boundary condition from the surface flux has multiple solutions for the potential, as it is the condition of zero potential at infinity that fixes the uniqueness of the potential. In order to rectify this problem, it is necessary to determine the surface potential, ϕ^{Γ}, from the external solution. The potential in the rest of the domain can then be obtained by solving:

$$K^{\Omega}\underline{\phi}^{\Omega} = M^{\Omega}\underline{q}^{\Omega} - K^{\Gamma}\underline{\phi}^{\Gamma} \tag{6.65}$$

where, as before, Ω refers to the interior nodes of the meshes, and Γ to the surface nodes. Having obtained an estimate for the interior potential, the surface flux is calculated from:

$$\underline{J}^{\Gamma} \equiv \frac{\partial\phi}{\partial n} \equiv \nabla\phi.\underline{n} = \sum_{i}^{N} \nabla\psi_i\phi_i.\underline{n} \tag{6.66}$$

where $\underline{n}$ is the normal of the surface element, and

$$\nabla\psi_i = \boldsymbol{J}^{-1}\frac{d\psi_i}{d\eta_i} \tag{6.67}$$

where

$$\boldsymbol{J} = \begin{pmatrix} \frac{\partial s}{\partial x} & \frac{\partial t}{\partial x} & \frac{\partial u}{\partial x} \\ \frac{\partial s}{\partial y} & \frac{\partial t}{\partial y} & \frac{\partial u}{\partial y} \\ \frac{\partial s}{\partial z} & \frac{\partial t}{\partial z} & \frac{\partial u}{\partial z} \end{pmatrix} \tag{6.68}$$

is the Jacobian of the transformation from the deformed element coordinate system (x, y, z) to that of the right angled tetrahedron (see figure 6.7) which need not be constant in (s, t, u) (see Section 6.5.2). Having thus obtained the flux $\underline{J}^{\Gamma}$, the BEM equation may then be solved for the exterior:

$$C\underline{\phi}^{\Gamma} = -D\underline{J}^{\Gamma} \tag{6.69}$$

yielding the surface potential, which can be fed back into equation (6.65) until convergence.

6.5.1 Test problem: charged sphere

In order to verify the numerical algorithm we constructed two test problems for which an analytic solution is known. These consisted of a spherical protein with either a point charge at its centre or a uniform charge density in the interior. For the first problem of the point charge, the solution converged (rapidly) to the correct solution. However, for the uniformly charged sphere a large discrepancy became apparent (see figures 6.5 and 6.6). In such a system it was found that

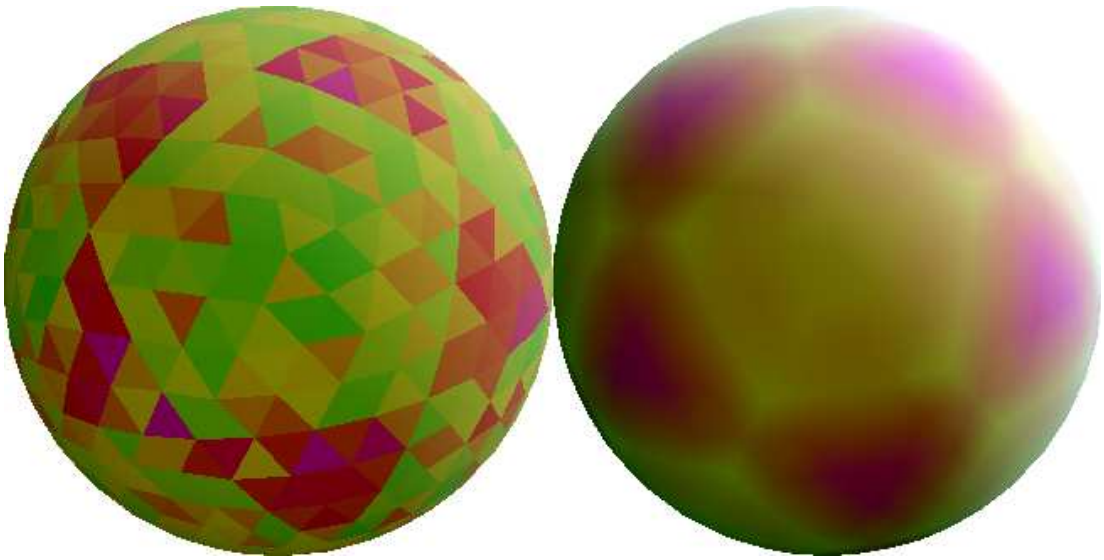


Figure 6.5: Two spheres illustrating the "football" effect. The right sphere shows the surface potential obtained from solving a uniformly charged sphere decomposed into linear elements. On the left we have the same sphere, but with each face coloured according to the volume of the surface element it belongs to. This demonstrates that the surface pattern actually occurs due to the inaccuracy of evaluating the surface flux using linear elements, as larger elements tend to cause an underestimation in the gradient of the potential within them.

the surface potential did not converge to a uniform value across the surface. Instead, a football-like pattern formed, with areas of high and low potential. Figure 6.6 shows that this is not a small effect - the spread and deviation of the potential from the analytic solution approaching the surface is enormous for the low screening cases. Examination of the underlying mesh revealed that the pattern was correlated with the volumes of the surface elements (see figure 6.5). The source of this pattern arises from errors in the flux calculation. Specifically, it is due to a large underestimate of the flux in 'longer' elements. This is because the potential is assumed to be linear across an element, so that the gradient is constant, whereas it actually becomes steeper as it approaches the surface. To rectify this, it was decided that the electrostatics implementation needed to use higher order, quadratic, shape functions.

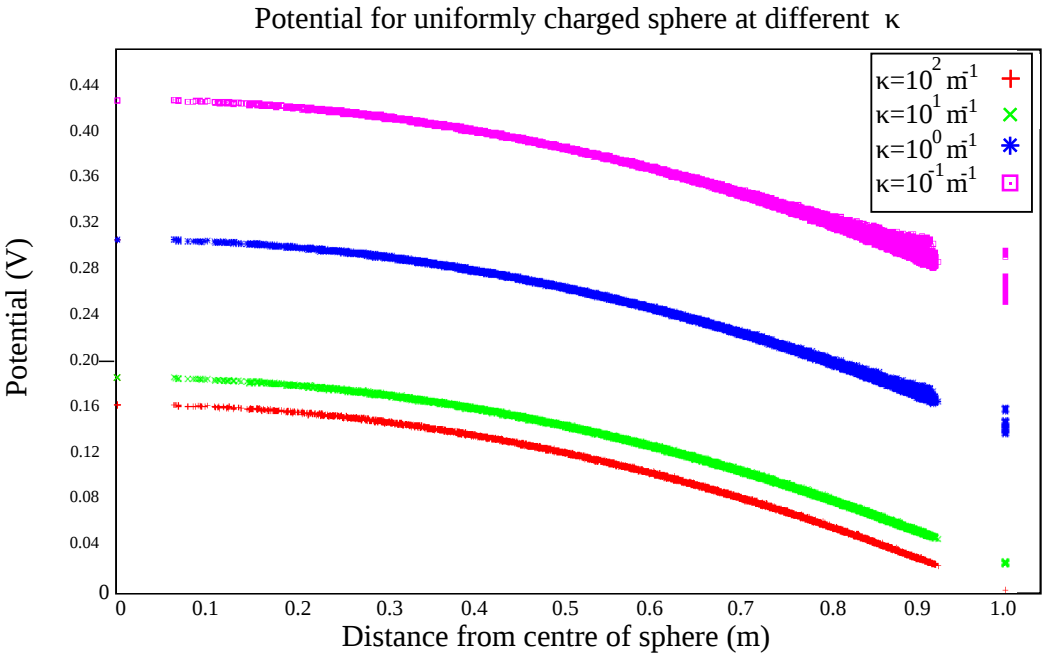


Figure 6.6: The plot shows the electrostatic potential profile of a uniformly charged sphere (radius 1 m) for varying values of the inverse screening length, κ. It can be seen that as soon as the screening is sufficiently low for parts of the surface to 'see' other parts, we obtain a spreading of the potential, giving us the 'football' pattern illustrated in figure 6.5. By comparing the potential deep inside the sphere to that as it approaches the surface, it is clear that this spreading is very much a surface effect, and not a system wide effect as we would expect for a bad interior solution scheme.

6.5.2 Second order electrostatics scheme

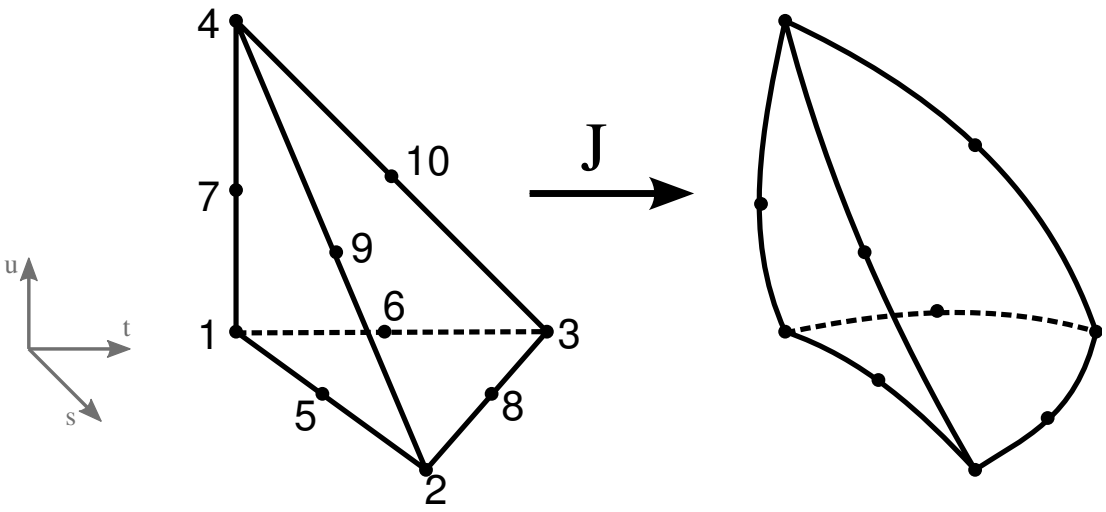


Figure 6.7: The transformation from (s, t, u), the space in which the element is a right angled unit tetrahedron, to its real space (x, y, z), deformed state, for a 10 node, second order (quadratic) tetrahedral element. Due to differing levels of compression in different parts of the element, the Jacobian of the transformation is not constant in space.

Instead of using 4-node linear elements for the potential ϕ, we instead use a 10-node quadratic element with nodes at each vertex and at the midpoints of each edge (see figure 6.7). Conversion to this second order scheme involves replacing the four simple shape functions of the linear case ($\psi_1 = 1 - s - t - u, \psi_2 = s, \psi_3 =$

$t, \psi_4 = u$) with ten quadratic shape functions:

$$\psi_1 = (1 - 2(s - t - u))(1 - s - t - u) \tag{6.70}$$

$$\psi_2 = (2s - 1)s \tag{6.71}$$

$$\psi_3 = (2t - 1)t \tag{6.72}$$

$$\psi_4 = (2u - 1)u \tag{6.73}$$

$$\psi_5 = 4s(1 - s - t - u) \tag{6.74}$$

$$\psi_6 = 4t(1 - s - t - u) \tag{6.75}$$

$$\psi_7 = 4u(1 - s - t - u) \tag{6.76}$$

$$\psi_8 = 4st \tag{6.77}$$

$$\psi_9 = 4su \tag{6.78}$$

$$\psi_{10} = 4tu. \tag{6.79}$$

The second order mass matrix is then calculated as:

$$M_{ij} = \int \psi_i \psi_j dx dy dz = \int \psi_i \psi_j |\det J(s, t, u)| ds dt du \tag{6.80}$$

where J is the Jacobian of the transformation from real space to (s, t, u) space as illustrated in figure 6.7 and given in equation (6.68). The Jacobian J is no longer constant[1] but depends on position within the element. Therefore the integrals are no longer solvable analytically, and we must resort to Gaussian quadrature schemes to evaluate them numerically. There is a similar dependence for the

[1] J is no longer constant in general (under isoparametric deformations). However, if the tetrahedra faces remain flat then the Jacobian will remain constant, as this reduces to the linear case.

calculation of the Poisson matrix, $K_{ij} = \int \nabla\psi_i.\nabla\psi_j dxdydz$. Beginning from the form of the Jacobian given in equation (6.68), and taking a position within the element of $\underline{r} = (x, y, z)$, with element nodes $\underline{n}_{1..10}$, we have:

$$\begin{aligned}\frac{\partial \underline{r}}{\partial s} &= \sum_{i=1}^{10} \frac{\partial \psi_i}{\partial s}\underline{n}_i \\ &= 4(\underline{n}_1 + \underline{n}_2 - 2\underline{n}_5)s + 4(\underline{n}_1 - \underline{n}_5 - \underline{n}_6 + \underline{n}_8)t \\ &+ 4(\underline{n}_1 - \underline{n}_5 - \underline{n}_7 + \underline{n}_9)u + (4\underline{n}_5 - 3\underline{n}_1 - \underline{n}_2).\end{aligned} \tag{6.81}$$

$$\begin{aligned}\frac{\partial \underline{r}}{\partial t} &= \sum_{i=1}^{10} \frac{\partial \psi_i}{\partial t}\underline{n}_i \\ &= 4(\underline{n}_1 - \underline{n}_5 - \underline{n}_6 + \underline{n}_8)s + 4(\underline{n}_1 + \underline{n}_3 - 2\underline{n}_6)t \\ &+ 4(\underline{n}_1 - \underline{n}_6 - \underline{n}_7 + \underline{n}_1 0)u + (4\underline{n}_6 - 3\underline{n}_1 - \underline{n}_3).\end{aligned} \tag{6.82}$$

$$\begin{aligned}\frac{\partial \underline{r}}{\partial u} &= \sum_{i=1}^{10} \frac{\partial \psi_i}{\partial u}\underline{n}_i \\ &= 4(\underline{n}_1 - \underline{n}_5 - \underline{n}_7 + \underline{n}_9)s + 4(\underline{n}_1 - \underline{n}_6 - \underline{n}_7 + \underline{n}_1 0)t \\ &+ 4(\underline{n}_1 + \underline{n}_4 - 2\underline{n}_7)u + (4\underline{n}_7 - 3\underline{n}_1 - \underline{n}_4).\end{aligned} \tag{6.83}$$

This second-order scheme greatly improved the accuracy of the J^Γ calculation and as such reduced the unphysical patterning of potential across the surface of the uniformly charged sphere to negligible levels, as shown in figure 6.8). In the case of very low screening ($\kappa = 10^{-1}$), the potential spread is approximately

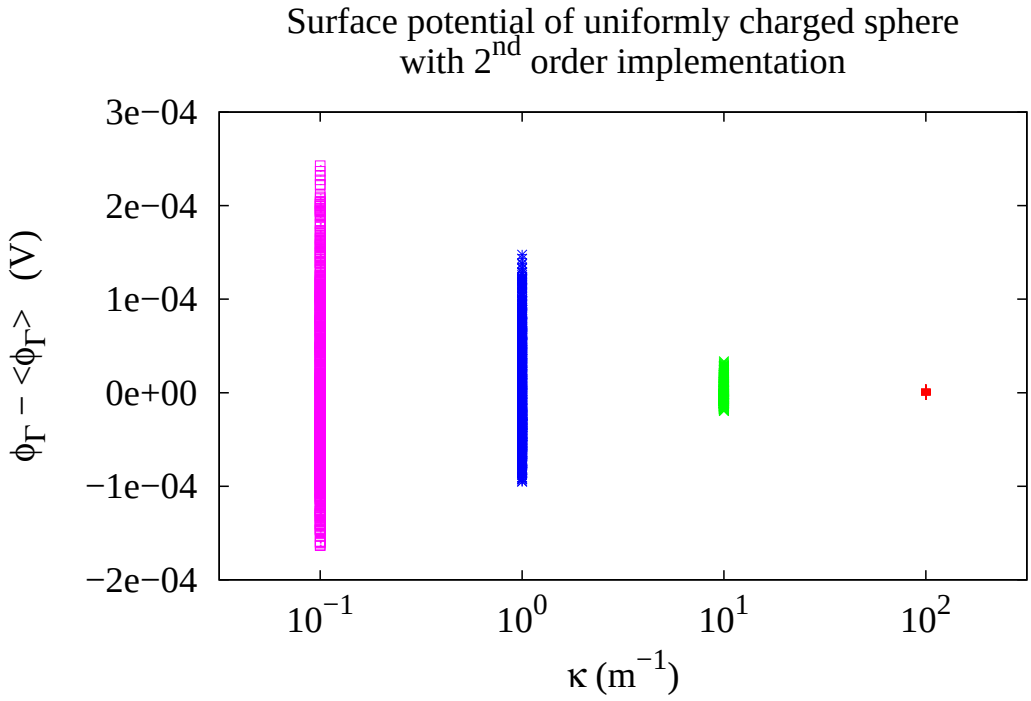


Figure 6.8: The spread of potential values on the surface nodes of the uniformly charged sphere. The mean surface potential $\langle\phi_\Gamma\rangle$ has been subtracted in order to compare the spreads at different kappa. The spreading is approximately 100 times lower in the second order implementation, therefore only surface potential is shown here (as opposed to across the entire sphere as shown in figure 6.6).

4×10^{-4} V in the second order implementation. Comparing this with the spread in the first order implementation shown in figure 6.6 (approximately 5×10^{-2} V) we see that the spreading of surface potential in the second order implementation is approximately 100 times less. The spreading in the worst (lowest screening) case is therefore now negligible when compared with the magnitude of the potential in the rest of the sphere. Note that such spreading cannot in practice be eliminated completely due to the discretisation of the sphere domain.

6.5.3 Technical Implementation

While solving the accuracy problem, this solution method introduces problems in maintaining parallelisation efficiency. Due to the increased number of degrees of freedom and interconnectivity required for the second order scheme, and the need to use Gaussian quadrature in the matrix assembly, it is even more important to ensure that such work does not count towards the serial fraction of the code. Since both the interior matrices (referred to in previous sections as K) and the exterior matrices (C and D) change with time (due to thermal fluctuation of the mesh), it is necessary to rebuild them. Parallel construction of sparse matrices is complex and not a present feature of the software libraries appropriate for this project, particularly with regards to matrices having an unknown sparsity pattern.

Sparse Matrices with known, fixed sparsity patterns

The full interior matrix, K, is the sum of all the individual element matrices (4×4 and 10×10 in the linear and quadratic cases respectively). Each row and column refers to a particular node in the mesh and therefore, providing the

mesh connectivity is not changed, it is known at the beginning of the simulation what position in the full K matrix each entry will contribute to. During the initialisation stage, all contributions from all element matrices are aggregated into a sorted list from which the matrix structure can be determined. The resulting data structure knows not only which elements are non-zero (the sparsity pattern), but also how many contributions there will be to each element in each build, and also the list of the memory locations of all these contributions. The construction of K can then proceed as follows:

1. Divide up the elements over all available processors, and calculate the individual matrices. Note that these are symmetric matrices, so the computation and memory costs can be reduced quite significantly.

2. Divide all nonzero entries to the K matrix over all available processors. Each entry can then be calculated by summing up all the values pointed to in its list of contributing elements (in the individual element matrices).

Given that the matrix construction phase occurs once per time step, whereas the matrix-vector multiplication operations can happen many times (in the Conjugate Gradient (CG) solver algorithm) it is important to construct the matrix in a form that is both succinct and sorted (to allow for parallel matrix-vector operations). It is for this reason that alternative, less complex means of storing sparse matrices were not used (for example, an unsorted list of non-zero entries).

Sparse Matrices with unknown sparsity patterns

While solutions to the interior problem can benefit from the constant mesh connectivity, the exterior problem cannot. The BEM matrices, C and D, are in

principle dense and have only been rendered sparse through a nearest neighbour look-up table that discounts distant face pair interactions on the basis of electrostatic screening. This means that there is no a priori way of knowing the sparsity pattern, or even how much memory should be allocated ("playing it safe" and allocating the full memory that could be used in the case of no screening will break the memory limit for even a moderately sized system). For this reason, a very different data structure is required in the parallelisation of the BEM matrices which is unavoidably less efficient than in the case of known sparsity patterns explained above.

The matrix is stored as an array of `Vectors`[1]. The size of the array is fixed by the total number of faces in the system, as there must be at least one contribution (the self contribution) in each row. The number of additional interactions contributing to each row is not fixed, so the rows are stored such that they can expand in memory as required. Fortunately, there is only one contribution for each face pair interaction. Furthermore, the ordering of entries within each row is not important since the matrix-vector multiply is parallelised on a row by row basis.

The construction of C and D can then be carried out by dividing up all the faces in the system over the available processors. Then, for each face, we search the nearest neighbour list and calculate the interaction with all nearby faces. Finally, we add the results to the row corresponding to the primary face, in any order.

[1] An extendable array whose elements are stored contiguously in memory.

6.6 Conclusions

Electrostatic forces play a crucial role in protein-protein interactions. Most continuum based solvers treat proteins as a region of low dielectric constant immersed in a solvent with high dielectric constant. The electric potential inside a protein is modelled by the Poisson equation, while the solvent is modelled by the Poisson-Boltzmann equation (due to salt ions present in the solvent that create a screening effect). Three numerical methods were considered for solving the electrostatic potential in the FFEA model: Finite Difference, the Finite Element Method, and the Boundary Element Method. Each of these methods on their own were found inadequate for the requirements of FFEA. A suitable electrostatics implementation for the FFEA model was found by solving the protein interior with FEM and the exterior solvent with BEM, coupling the two methods via the electric flux at the protein surface. This method was found to give the correct potential for a spherical protein with a single point charge at its centre. A second-order FEM scheme was required to give the correct potential for a uniformly charged spherical protein.

Chapter 7

Conclusions

7.1 Summary

Fluctuating Finite Element Analysis (FFEA) is a coarse-grained simulation technique designed for protein simulation in the mesoscale. FFEA describes protein matter as a viscoelastic continuum subject to thermal noise, allowing it to probe far longer length and time scales than available with coarse-grained MD. This thesis presented work on the development of FFEA, mainly in terms of software development (chapter 2), extensions to the model (chapters 3 and 6) and applications to biological systems (chapters 4 and 5).

Chapter 1 described the range of length and time scales spanned by biological processes, and the corresponding simulation techniques appropriate to each scale. The spectrum can be crudely divided into the nanoscale, the mesoscale and the macroscale. The mesoscale (lengths of 10 nm to 1 μm and times of ns to hundreds of μs) is the domain of cytoplasmic crowding and molecular motors, making it a very important and interesting scale in biology. There is a gap in suitable

techniques for the lower mesoscale, which is small enough for thermal fluctuations to be important, but too large for nanoscale techniques like MD (even coarse-grained). FFEA is a suitable technique to use for modelling biological processes on this scale.

Chapter 2 discussed the creation of the FFEA simulation package and tools, including parallelisation of the code and benchmarking to determine the most appropriate solvers for different mesh topologies. The stability of the model with respect to time step and minimum edge length was investigated. The automated pipeline for creation of FFEA input files from atomistic or low density cryo-EM structure data was also presented, along with purpose-built visualisation software for editing and preparing FFEA simulations.

Chapter 3 presented a finite element formulation of van der Waals interactions and discussed the practical issues surrounding its implementation. This formulation introduced steric repulsion and dispersion forces/hydrophobic attraction to the model. A surface-surface interaction was chosen to avoid duplication (continuum elastic properties already conceptually incorporate vdW forces within the protein interior) and singularity issues. The stability of the scheme was investigated with 140 simulations of 128 myoglobins interacting with a polystyrene substrate for different values of Young's modulus, Poisson ratio and interaction energy.

Chapter 4 used PCA to provide a comparison of FFEA with a related technique, Elastic Network Model (ENM), using three rotary ATPases as a test system, finding agreement on the first two modes. The effect on dynamics of differing numbers of stators was also examined using PCA on simulation trajectories. The effects of severing the connection between the *c*-ring and stator (potentially an

experimental artifact) were similarly investigated. Finally, some simulations of the V_o and V_1 domains dissociating were run, predicting that the structural integrity of the motor is dependent on the C subunit. The method and results are published in reference [86].

Chapter 5 presented a simulation study of flagellar dynein operating in the confines of the axoneme. The simulations suggest that the step length of axonemal dynein c is increased by 3 to 4 nm through interaction with the microtubule doublet. The number of binding sites explored by the microtubule binding domain was found to be between 10 and 15 in the ADP•Vi state, and around 7 in the apo state. A scheme was developed for mapping between incompatible meshes, and used to simulate conformational switching of the dynein motor during the simulations.

Chapter 6 introduced electrostatics to the FFEA model, in the form of a coupled FEM-BEM scheme, using FEM to solve for the electrostatic potential in the protein interiors, and BEM for the potential in the exterior solvent. A second order FEM implementation was required to overcome surface flux problems. The implementation was tested with a uniformly charged sphere.

7.2 Outlook

At the time of submission of this thesis there are 2535 EMDB map entries in the EMDataBank online resource[103]. This represents an enormous source of structural data that is poorly exploited, if at all, by present simulation techniques. The recent development of a means to detect electrons directly (providing higher resolution than the indirect light-based CCD devices and faster readout time than

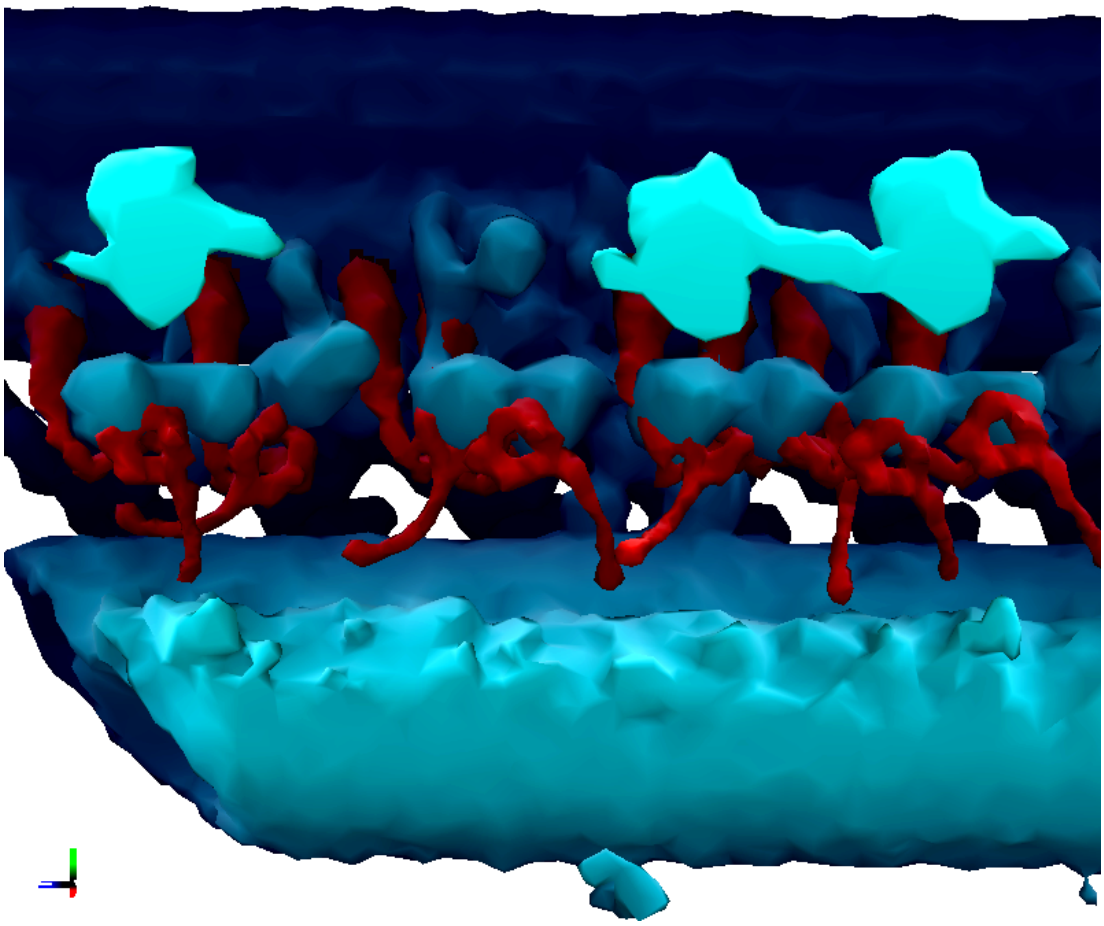


Figure 7.1: Multiple inner arm dyneins operating in a '9 + 2' axoneme. In practice each dynein motor is different, whereas the shown simulation is using the same dynein c model for each.

film[159]) represents a great breakthrough in cryo-EM technology, particularly in the area of electron cryo-tomography which can image the complex organisation of cellular compartments at subnanometer resolution[160]. This will undoubtedly lead to many more entries in the EMDB in the near future, giving important information regarding the structural organisation of the cell at the mesoscale. FFEA provides a natural method for simulating this low resolution data, allowing computational biophysicists access to the diverse and important processes that occur on the mesoscale.

There are ongoing simulations of axonemal dynein c's proposed mechanochemical cycle, which is believed to be based on the mechanochemical cycle of cytoplasmic dynein[161]. Ultimately, this work is intended to contribute to a fully functioning simulation model of the axoneme, allowing visualisation and study of the complex nature of its mechanism, which is as yet poorly understood. Figure 7.1 gives a purely illustrative image of this goal (in practice the motors are all different variants of dynein), with all molecular motors present and pulling on dynamic microtubule doublets. Important steps have been made towards this objective (such as parameterisation of elastic properties and conformational mapping between states) but further work will be needed. Most notably, a proper hydrodynamical treatment of the exterior solvent will be essential in the case of multiple functioning dynein motors in the confines of the axoneme, as hydrodynamic coupling will certainly influence the dynamics. Such a model could then be integrated with higher length scale continuum models of the axoneme (such as finite element models[162]) and contribute towards a full multiscale understanding of the axonemal switching process.

FFEA has the potential to act as the 'missing link' between coarse-grained

MD simulations and macroscopic simulation techniques, providing a significant advance towards computational modelling of the multiscale nature of life.

Appendix A

Algorithm pseudo-code

For each Protein object:

Read *parameters*, *nodes* and *topology* input files

Construct protein object based on these parameters

Loop for *num_steps* steps:

For each Protein object:

For each element in the protein structure:

Construct the Jacobian matrix

Invert the Jacobian matrix and calculate the shape function derivatives

Construct the bulk and shear viscous matrices

Construct the bulk and shear elastic stress tensors

Construct the fluctuating stress tensor

Calculate the forces on each of the four nodes due to this element

Calculate net force on each node in mesh by aggregating

contributions from each element

Calculate the forces arising from vdW, Electrostatics and Hydrodynamic drag

Solve the linear equation set $M\vec{a} = \vec{f}$, where M is the mass matrix,
$\vec{a}$ is the (unknown) acceleration vector solution and $\vec{f}$ is the force vector before application of the mass matrix

Integrate equation of motion using a numerical integration scheme

Update the nearest neighbour face look up table

If this is a "check" step:

Output the current node positions to the *trajectory* file
and measurements to the *measurement* file

Clean up: Release memory resources and close any open files

Appendix B

Example `.ffea` script

```
<param>
        <restart = 1>

        <trajectory_out_fname = transition_run_trajectory.out>
        <measurement_out_fname = transition_run_measurement.out>

        <kT = 4e-21>
        <dt = 1e-14>
        <check = 10000>
        <num_steps = 9420000>
        <rng_seed = time>

        <max_iterations_cg = 1000>
        <epsilon = 1e-11>
```

```
<calc_es = 0>

<kappa = 1e9>

<es_update = 1>

<epsilon_0 = 1>

<dielec_ext = 1>


<do_stokes = 1>

<stokes_visc = 1e-06>


<es_h = 4>

<es_N_x = 25>

<es_N_y = 30>

<es_N_z = 50>


<wall_x_1 = PBC>

<wall_x_2 = PBC>

<wall_y_1 = HARD>

<wall_y_2 = HARD>

<wall_z_1 = PBC>

<wall_z_2 = PBC>


<calc_vdw = 1>

<vdw_forcefield_params = ADPVI_axoneme.lj>


<num_blobs = 3>
```

```
</param>
<system>
        <blob>
                <state = DYNAMIC>

                <stokes = ADPVI_Dynein.stokes>
                <pin = ADPVI_Dynein.pin>
                <vdw = ADPVI_Dynein.vdw>
                <solver = CG>
                <material = ADPVI_Dynein.mat>
                <surface = ADPVI_Dynein.surf>
                <nodes = ADPVI_Dynein.node>
                <topology = ADPVI_Dynein.top>
                <scale = 1>
        </blob>
        <blob>
                <state = FROZEN>

                <stokes = APO_Dynein.stokes>
                <pin = APO_Dynein.pin>
                <vdw = APO_Dynein_non_interacting.vdw>
                <solver = CG>
                <material = APO_Dynein.mat>
                <surface = APO_Dynein.surf>
                <nodes = APO_Dynein.node>
```

```
                <topology = APO_Dynein.top>
                <scale = 1>
        </blob>
        <blob>
                <state = STATIC>


                <vdw = axoneme.vdw>
                <nodes = axoneme.node>
                <surface = axoneme.surf>
                <scale = 1>
        </blob>
</system>
```

Appendix C

Example simulation setup with FFEA_tools

The following example procedure is intended to illustrate how one might simulate a protein interacting with a hard rough surface.

Convert the protein pdb file to $50 \times 50 \times 50$ voxel density map, with each atom rendered as a linear density sphere of van der Waals radius 10Å:

```
FFEA_tools pdbtomap protein.pdb protein.mrc 50 50 50 10
```

Mesh the density map for the isosurface at level 0.01, filling interior cavities (defined as closed volumes of less than 20 voxels) and coarsening each $2 \times 2 \times 2$ block of voxels. Interpolate surface node positions based on difference between local density and isolevel. Output the surface as a `.off` file.

```
FFEA_tools meshmap -map protein.map -out protein.off -format off -level
0.01 -coarse 2 -fill_cavities 20 -interpolate yes
```

Produce all the necessary FFEA input files for this protein mesh, giving it the viscosity of water, a density 1.5 times that of water, and shear and bulk moduli of 120MPa and 640MPa respectively. Note that homogeneous material parameters are used here for simplicity, but they can in practice be different in each element of the protein mesh.

```
FFEA_tools makeffea -mesh protein.off -stokes_radius 1e-9 -density 1500.0
-shear_visc 1e-3 -bulk_visc 1e-3 -shear_mod 120e6 -bulk_mod 640e6 -dielec
1.0
```

The FFEA_viewer can now be used to select which faces on the protein mesh are interacting, and what sort of surface type they have (hydrophobic, hydrophilic etc.). Now, similarly, we produce a mesh of the surface, obtained from Cryo-EM density map rough_surface.ccp4. In this case, however, the -cull_floaters option is removing any joined group of voxels that number less than 120. This has the effect of 'cleaning' the density map from the clutter of noisy artifacts, leaving only the desired surface mesh. This surface will be STATIC, so the material parameters do not matter.

```
FFEA_tools meshmap -map rough_surface.ccp4 -out surface.off -format off
-level 110 -coarse 3 -fill_cavities 20 -cull_floaters 120 -interpolate
yes
```

```
FFEA_tools makeffea -mesh surface.off -stokes_radius 1 -density 1
-shear_visc 1 -bulk_visc 1 -shear_mod 1 -bulk_mod 1 -dielec 1
```

A `.ffea` script file, as given in Appendix B, can now be produced, instructing the `FFEA` program to simulate two proteins, one `DYNAMIC` and one `STATIC`, for 1×10^9 steps (sampling every 1×10^4), interacting via a van der Waals interaction and using a nearest neighbour lookup table of size $100 \times 60 \times 40$ nm. The simulation can then be run:

```
FFEA surf_interaction.ffea
```

And the resultant trajectory can be visualised with the viewer.

```
FFEA_viewer surf_interaction.ffea
```

Appendix D

Derivation of the Poisson-Boltzmann Equation

D.1 Nonlinear form

Begin with the Poisson Equation:

$$\nabla^2\phi(\vec{r}) + \frac{\rho(\vec{r})}{\epsilon} = 0 \tag{D.1}$$

describing the electrostatic potential ϕ at some position $\vec{r}$ in a medium of permittivity $\epsilon = \epsilon_r\epsilon_0$. The density of charge in the system is made up of fixed charges:

$$\rho(\vec{r})_{fixed} = \sum_{k}^{n_{fixed}} Q_k\delta(\vec{r} - \vec{r}_k) \tag{D.2}$$

and mobile charges, the ions and counter ions in the (overall electroneutral) solvent. For n_s ion species, each with a valency z_i and concentration $c_i(\vec{r})$, we

calculate the charge density of the system to be:

$$\rho(\vec{r}) = \sum_{i}^{n_s} z_i e c_i(\vec{r}) + \sum_{k}^{n_{fixed}} Q_k \delta(\vec{r} - \vec{r}_k) \tag{D.3}$$

where as usual e is the elementary charge. We must now find an expression for the concentration of ionic species in the solvent, $c(\vec{r})$. The energy of an amount of charge $z_i e$ positioned at some location $\vec{r}$ is $z_i e\phi(\vec{r})$, and therefore the probability of finding it there must be proportional to $\exp(-\frac{z_i e\phi(\vec{r})}{kT})$, assuming an equilibrium state (for the Boltzmann statistics). The concentration of ions at a location $\vec{r}$ must be proportional to the probability of those ions being at that point and therefore:

$$c_i(\vec{r}) \propto \exp(-\frac{z_i e\phi(\vec{r})}{kT}). \tag{D.4}$$

In the limit of $\vec{r}$ being infinitely far away from any macro ions, the potential ϕ drops to zero, and the concentration becomes the bulk concentration, c_i^0:

$$c_i(\vec{r}) = c_i^0 \exp(-\frac{z_i e\phi(\vec{r})}{kT}). \tag{D.5}$$

Substituting all this into Poisson's equation yields the Poisson-Boltzmann equation (nonlinear form):

$$\epsilon\nabla^2\phi(\vec{r}) + \sum_{i}^{n_s} z_i e c_i^0 \exp(-\frac{z_i e\phi(\vec{r})}{kT}) + \sum_{k}^{n_{fixed}} Q_k \delta(\vec{r} - \vec{r}_k) = 0. \tag{D.6}$$

If we so wish, we can choose 2 species (ions and counter-ions) of equal and opposite charge (i.e. $z_1 = -1\ ,\ z_2 = +1$), and expand the sum into two exponential terms with positive and negative arguments. We then use the following definition of

hyberbolic sine:

$$2\sinh(x) = e^{x} - e^{-x} \tag{D.7}$$

to combine these two terms, yielding the usual form of the nonlinear PBE:

$$\epsilon\nabla^2\phi(\vec{r}) + 2c_\infty e\sinh(\frac{e\phi(\vec{r})}{kT}) + \sum_{k}^{n_{fixed}} Q_k\delta(\vec{r} - \vec{r}_k) = 0 \tag{D.8}$$

where c_∞ is the bulk concentration for both at infinity.

D.2 Linear form

As this equation is unpleasant to solve (even numerically), we seek to simplify the equation through the approximation that the electrostatic potential is always small. In physical terms, we restrict ourselves to the regime in which the energy due to a charge in the given potential is much less than the thermal energy in the system:

$$e\phi(\vec{r}) << kT \tag{D.9}$$

such that we may use the first term in the taylor expansion of sinh:

$$\sinh(x) \approx x \ , \ x << 1 \tag{D.10}$$

yielding the linearised PBE:

$$\nabla^2\phi(\vec{r}) + \kappa^2\phi(\vec{r}) = -\frac{1}{\epsilon}\rho_{fixed} \tag{D.11}$$

where all the constants have been absorbed into the inverse Debye screening length,

$$\kappa = \left(\frac{2c_\infty e^2}{\epsilon kT} \right)^{\frac{1}{2}} \tag{D.12}$$

Appendix E

Origin of the Robin Boundary Condition

Two commonly used boundary conditions are the Dirichlet (fixed value on boundary) and Neumann (fixed gradient on boundary) BCs. The Robin BC is merely a (weighted) linear combination of both of these. Its general form is as follows:

$$c_1\phi + c_2\frac{\partial\phi}{\partial n} = c_3 \tag{E.1}$$

at the boundary. If we assume that the screening length is very short (κ very large), then the region of interest is therefore found very close to the surface of the protein. At such a distance we say that the surface is locally flat and the solution to the PBE is that for the 1-d case:

$$\phi(x) = A\exp(-\kappa x). \tag{E.2}$$

Taking the derivative we have:

$$\frac{d\phi(x)}{dx} = -\kappa\phi(x) \tag{E.3}$$

from whence we obtain the (approximate) boundary condition

$$\frac{d\phi}{dn} = -\kappa\phi \tag{E.4}$$

as used in chapter 6. Note that this has the form of the general Robin BC in equation (E.1) with $c_1 = \kappa$, $c_2 = 1$ and $c_3 = 0$.

References

[1] W. Holzapfel, U. Finkele, W. Kaiser, D. Oesterhelt, H. Scheer, H. U. Stilz, and W. Zinth, "Observation of a bacteriochlorophyll anion radical during the primary charge separation in a reaction center," *Chemical Physics Letters*, vol. 160, no. 1, pp. 1–7, 1989. xi, 1, 2, 3

[2] R. O. Dror, R. M. Dirks, J. Grossman, H. Xu, and D. E. Shaw, "Biomolecular simulation: a computational microscope for molecular biology," *Annual Review of Biophysics*, vol. 41, pp. 429–452, 2012. 1

[3] A. Warshel, "Computer simulations of enzyme catalysis: methods, progress, and insights," *Annual Review of Biophysics and Biomolecular Structure*, vol. 32, no. 1, pp. 425–443, 2003. xi, 2, 3

[4] A. Jayaraman, C. K. Hall, and J. Genzer, "Computer simulation study of molecular recognition in model DNA microarrays," *Biophysical*, vol. 91, no. 6, pp. 2227–2236, 2006. xi, 2, 3

[5] B. M. Johnston, P. R. Johnston, S. Corney, and D. Kilpatrick, "Non-Newtonian blood flow in human right coronary arteries: transient simulations," *Journal of Biomechanics*, vol. 39, no. 6, pp. 1116–1128, 2006. xi, 3, 6

[6] S. McGuffee and A. Elcock, "Atomically detailed simulations of concentrated protein solutions: the effects of salt, pH point mutations, and protein concentration in simulations of 1000-molecule systems," *Journal of the American Chemical Society*, vol. 128, no. 37, pp. 12098–12110, 2006. xi, 3, 6, 8

[7] D. Shaw, P. Maragakis, K. Lindorff-Larsen, S. Piana, R. Dror, M. Eastwood, J. Bank, J. Jumper, J. Salmon, Y. Shan, *et al.*, "Atomic-level characterization of the structural dynamics of proteins," *Science*, vol. 330, no. 6002, p. 341, 2010. 2, 5

[8] J. Šponer, J. E. Šponer, A. Mládek, P. Banáš, P. Jurečka, and M. Otyepka, "How to understand quantum chemical computations on DNA and RNA systems? A practical guide for non-specialists," *Methods*, vol. 64, no. 1, pp. 3–11, 2013. 4

[9] M. Frisch, G. Trucks, H. Schlegel, G. Scuseria, M. Robb, J. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. Petersson, *et al.*, "Gaussian 09, Gaussian," *Inc., Wallingford, CT*, 2009. 4

[10] S. J. Clark, M. D. Segall, C. J. Pickard, P. J. Hasnip, M. I. Probert, K. Refson, and M. C. Payne, "First principles methods using CASTEP," *Zeitschrift für Kristallographie*, vol. 220, no. 5/6/2005, pp. 567–570, 2005. 4

[11] A. D. Laurent and D. Jacquemin, "TD-DFT benchmarks: A review," *International Journal of Quantum Chemistry*, vol. 113, no. 17, pp. 2019–2039, 2013. 4

[12] T. A. Soares, P. H. Hünenberger, M. A. Kastenholz, V. Kräutler, T. Lenz, R. D. Lins, C. Oostenbrink, and W. F. van Gunsteren, "An improved nucleic acid parameter set for the GROMOS force field," *Journal of Computational Chemistry*, vol. 26, no. 7, pp. 725–737, 2005. 4

[13] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, "CHARMM: A program for macromolecular energy, minimization, and dynamics calculations," *Journal of Computational Chemistry*, vol. 4, no. 2, pp. 187–217, 1983. 4

[14] R. Salomon-Ferrer, D. A. Case, and R. C. Walker, "An overview of the Amber biomolecular simulation package," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 3, no. 2, pp. 198–210, 2013. 4

[15] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel, *et al.*, "GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit," *Bioinformatics*, p. btt055, 2013. 4

[16] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten, "Scalable molecular dynamics with NAMD," *Journal of Computational Chemistry*, vol. 26, no. 16, pp. 1781–1802, 2005. 4

[17] D. Marx and J. Hutter, *Ab initio molecular dynamics: basic theory and advanced methods*. Cambridge University Press, 2009. 5

[18] G. Groenhof, "Introduction to QM/MM simulations," pp. 43–66, Springer, 2013. 5

[19] Y. Zhang, H. Liu, and W. Yang, "Free energy calculation on enzyme reactions with an efficient iterative procedure to determine minimum energy paths on a combined ab initio QM/MM potential energy surface," *The Journal of Chemical Physics*, vol. 112, no. 8, pp. 3483–3492, 2000. 5

[20] H. Lin and D. G. Truhlar, "QM/MM: what have we learned, where are we, and where do we go from here?," *Theoretical Chemistry Accounts*, vol. 117, no. 2, pp. 185–199, 2007. 5

[21] J. L. Klepeis, K. Lindorff-Larsen, R. O. Dror, and D. E. Shaw, "Long-timescale molecular dynamics simulations of protein structure and function," *Current Opinion in Structural Biology*, vol. 19, no. 2, pp. 120–127, 2009. 5

[22] R. M. Voorhees, A. Weixlbaumer, D. Loakes, A. C. Kelley, and V. Ramakrishnan, "Insights into substrate stabilization from snapshots of the peptidyl transferase center of the intact 70S ribosome," *Nature Structural & Molecular Biology*, vol. 16, no. 5, pp. 528–533, 2009. 5

[23] A. R. Mitchell and D. F. Griffiths, *The finite difference method in partial differential equations*. John Wiley, 1980. 6

[24] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007. 6

[25] B. M. Johnston, P. R. Johnston, S. Corney, and D. Kilpatrick, "Non-Newtonian blood flow in human right coronary arteries: steady state simulations," *Journal of Biomechanics*, vol. 37, no. 5, pp. 709–720, 2004. 6

[26] E. Holzbaur and R. Vallee, "Dyneins: molecular structure and cellular function," *Annual Review of Cell Biology*, vol. 10, no. 1, pp. 339–372, 1994. 6

[27] V. Tozzini, "Minimalist models for proteins: a comparative analysis," *Q. Rev. Biophys.*, vol. 43, no. 3, pp. 333–371, 2010. 7

[28] S. J. Marrink, A. H. de Vries, and A. E. Mark, "Coarse grained model for semiquantitative lipid simulations," *The Journal of Physical Chemistry B*, vol. 108, no. 2, pp. 750–760, 2004. 7

[29] S. Takada, "Gø-ing for the prediction of protein folding mechanisms," *Proceedings of the National Academy of Sciences*, vol. 96, no. 21, pp. 11698–11700, 1999. 7

[30] M. M. Tirion, "Large amplitude elastic motions in proteins from a single-parameter, atomic analysis," *Physical Review Letters*, vol. 77, no. 9, p. 1905, 1996. 7

[31] P. Chacón, F. Tama, and W. Wriggers, "Mega-Dalton biomolecular motion captured from electron microscopy reconstructions," *Journal of Molecular Biology*, vol. 326, no. 2, pp. 485–492, 2003. 7

[32] F. Tama, W. Wriggers, and C. L. Brooks III, "Exploring global distortions of biological macromolecules and assemblies from low-resolution structural information and elastic network theory," *Journal of Molecular Biology*, vol. 321, no. 2, pp. 297–305, 2002. 7

[33] S. McGuffee and A. Elcock, "Diffusion, crowding & protein stability in a

dynamic molecular model of the bacterial cytoplasm," *PLoS computational biology*, vol. 6, no. 3, p. e1000694, 2010. 8

[34] Z. Liang, Z. Gimbutas, L. Greengard, J. Huang, and S. Jiang, "A fast multipole method for the Rotne–Prager–Yamakawa tensor and its applications," *Journal of Computational Physics*, vol. 234, pp. 133–139, 2013. 8

[35] J. P. Hernández-Ortiz, J. J. de Pablo, and M. D. Graham, "Fast computation of many-particle hydrodynamic and electrostatic interactions in a confined geometry," *Physical Review Letters*, vol. 98, no. 14, p. 140602, 2007. 8

[36] R. Groot and P. Warren, "Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation," *Journal of Chemical Physics*, vol. 107, no. 11, p. 4423, 1997. 8, 9

[37] Z. G. Mills, W. Mao, and A. Alexeev, "Mesoscale modeling: solving complex flows in biology and biotechnology," *Trends in Biotechnology*, vol. 31, no. 7, pp. 426–434, 2013. 9

[38] G. Gompper, T. Ihle, D. Kroll, and R. Winkler, "Multi-particle collision dynamics: A particle-based mesoscale simulation approach to the hydrodynamics of complex fluids," 2008. 9

[39] B. Duenweg and A. J. C. Ladd, "Lattice Boltzmann Simulations of Soft Matter Systems," vol. 221 of *Advances in Polymer Science*, pp. 89–166, Springer-Verlag Berlin, 2009. 9

[40] B. Uma, T. Swaminathan, P. Ayyaswamy, D. Eckmann, and R. Radhakrishnan, "Generalized Langevin dynamics of a nanoparticle using a finite element approach: Thermostating with correlated noise," *The Journal of Chemical Physics*, vol. 135, no. 11, p. 114104, 2011. 10

[41] P. J. Atzberger, P. R. Kramer, and C. S. Peskin, "A stochastic immersed boundary method for fluid-structure dynamics at microscopic length scales," *Journal of Computational Physics*, vol. 224, no. 2, pp. 1255–1292, 2007. 10

[42] Reddy, J N, *An Introduction to the Finite Element Method (3rd Edition)*. McGraw-Hill Education, 2005. 10

[43] R. C. Oliver, D. J. Read, O. G. Harlen, and S. A. Harris, "A stochastic finite element model for the dynamics of globular macromolecules," *Journal of Computational Physics*, vol. 239, pp. 147–165, 2013. 12, 14, 17

[44] R. Oliver, *A stochastic finite element model for the dynamics of globular proteins*. University of Leeds, 2013. xvii, 12, 15, 47, 100, 101, 102, 103

[45] Y. Wang and G. Zocchi, "The folded protein as a viscoelastic solid," *EPL (Europhysics Letters)*, vol. 96, no. 1, p. 18003, 2011. 13, 60

[46] J. Drenth, *Principles of protein X-ray crystallography*. Springer, 2007. 16

[47] J. Cavanagh, W. J. Fairbrother, A. G. Palmer III, and N. J. Skelton, *Protein NMR spectroscopy: principles and practice*. Academic Press, 1995. 16

[48] J. Dubochet, M. Adrian, J.-J. Chang, J.-C. Homo, J. Lepault, A. W. Mc-

Dowall, and P. Schultz, "Cryo–electron microscopy of vitrified specimens," *Quarterly Reviews of Biophysics*, vol. 21, no. 02, pp. 129–228, 1988. 16

[49] J. Lipfert and S. Doniach, "Small-angle X-ray scattering from RNA, proteins, and protein complexes," *Annu. Rev. Biophys. Biomol. Struct.*, vol. 36, pp. 307–327, 2007. 16

[50] J. Frank, "Single-particle imaging of macromolecules by cryo–electron microscopy," *Annual Review of Biophysics and Biomolecular Structure*, vol. 31, no. 1, pp. 303–319, 2002. 16

[51] H. Fischer, I. Polikarpov, and A. F. Craievich, "Average protein density is a molecular-weight-dependent function," *Protein Science*, vol. 13, no. 10, pp. 2825–2828, 2004. 17, 37

[52] T. Cellmer, E. R. Henry, J. Hofrichter, and W. A. Eaton, "Measuring internal friction of an ultrafast-folding protein," *Proceedings of the National Academy of Sciences*, vol. 105, no. 47, pp. 18320–18325, 2008. 17, 37

[53] M. Radmacher, M. Fritz, J. P. Cleveland, D. A. Walters, and P. K. Hansma, "Imaging adhesion forces and elasticity of lysozyme adsorbed on mica with the atomic force microscope," *Langmuir*, vol. 10, no. 10, pp. 3809–3814, 1994. 17, 37, 70

[54] A. Ikai, R. Afrin, and H. Sekiguchi, "Pulling and pushing protein molecules by AFM," *Current Nanoscience*, vol. 3, no. 1, pp. 17–29, 2007. 17

[55] D. W. Walker, "The design of a standard message passing interface for

distributed memory concurrent computers," *Parallel Computing*, vol. 20, no. 4, pp. 657–673, 1994. 20

[56] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel Computing*, vol. 22, no. 6, pp. 789–828, 1996. 20

[57] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *Computational Science & Engineering, IEEE*, vol. 5, no. 1, pp. 46–55, 1998. 20

[58] C. Nvidia, "Compute unified device architecture programming guide," 2007. 21

[59] J. A. Anderson, C. D. Lorenz, and A. Travesset, "General purpose molecular dynamics simulations fully implemented on graphics processing units," *Journal of Computational Physics*, vol. 227, no. 10, pp. 5342–5359, 2008. 21

[60] G. Amdahl, "Limits of Expectation," *International Journal of Supercomputer Applications and High Performance Computing*, vol. 2, pp. 88–94, SPR 1988. 21

[61] B. Stroustrup *et al.*, *The C++ programming language*. Pearson Education India, 1995. 21

[62] A. T. Cohen, "Data abstraction, data encapsulation and object-oriented programming," *ACM SIGPLAN Notices*, vol. 19, no. 1, pp. 31–35, 1984. 22

[63] A. George, M. T. Heath, J. Liu, and E. Ng, "Sparse Cholesky factorization on a local-memory multiprocessor," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 2, pp. 327–340, 1988. 24

[64] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, vol. 49. National Bureau of Standards Washington, DC, 1952. 25

[65] J. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain." http://www.cs.cmu.edu/ quake-papers/painless-conjugate-gradient.pdf, 1994. 25

[66] J. M. Bull, "Measuring synchronisation and scheduling overheads in OpenMP," vol. 8, p. 49, Citeseer, 1999. 26

[67] University of Leeds, *Advanced Research Computing - ARC1*. 27

[68] N8 HPC, *N8 High Performance Computing - Polaris facility*. 27

[69] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, pp. 3–30, January 1998. 28

[70] M. Saito and M. Matsumoto, "SIMD-oriented fast Mersenne Twister: a 128-bit pseudorandom number generator," pp. 607–622, Springer, 2008. 28

[71] M. Matsumoto and T. Nishimura, "Dynamic creation of pseudorandom number generators," *Monte Carlo and Quasi-Monte Carlo Methods*, vol. 2000, pp. 56–69, 1998. 29

[72] J. M. Kollman, L. Pandi, M. R. Sawaya, M. Riley, and R. F. Doolittle, "Crystal structure of human fibrinogen," *Biochemistry*, vol. 48, no. 18, pp. 3877–3886, 2009. xiii, 39

[73] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," vol. 21, pp. 163–169, ACM, 1987. 40

[74] Blender Foundation, Blender Institute, Amsterdam, *Blender*. 41

[75] J. Schöberl, J. Gerstmayr, and R. Gaisbauer, "NETGEN - automatic 3d tetrahedral mesh generator.." http://www.hpfem.jku.at/netgen/, May 2003. 42

[76] H. Si, "TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator." http://tetgen.berlios.de/. 42

[77] W. Humphrey, A. Dalke, and K. Schulten, "VMD – Visual Molecular Dynamics," *Journal of Molecular Graphics*, vol. 14, pp. 33–38, 1996. 44

[78] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, and T. E. Ferrin, "UCSF Chimera - a visualization system for exploratory research and analysis," *Journal of Computational Chemistry*, vol. 25, no. 13, pp. 1605–1612, 2004. 44

[79] T. D. Grant, J. R. Luft, J. R. Wolfley, H. Tsuruta, A. Martel, G. T. Montelione, and E. H. Snell, "Small angle X-ray scattering as a complementary tool for high-throughput structural studies," *Biopolymers*, vol. 95, no. 8, pp. 517–530, 2011. 47

[80] R. Ellis, "Macromolecular crowding: an important but neglected aspect of the intracellular environment," *Current Opinion in Structural Biology*, vol. 11, no. 1, pp. 114–119, 2001. 47

[81] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case, "Development and testing of a general amber force field," *Journal of Computational Chemistry*, vol. 25, no. 9, pp. 1157–1174, 2004. 49

[82] L. A. Girifalco and V. G. Weizer, "Application of the Morse potential function to cubic metals," *Physical Review*, vol. 114, no. 3, p. 687, 1959. 49

[83] W. E. Rice and J. O. Hirschfelder, "Second Virial Coefficients of Gases Obeying a Modified Buckingham (ExpSix) Potential," *The Journal of Chemical Physics*, vol. 22, no. 2, pp. 187–192, 1954. 49

[84] M. J. Betts and R. B. Russell, "Amino acid properties and consequences of substitutions," *Bioinformatics for Geneticists*, vol. 317, p. 289, 2003. 57

[85] S.-A. Muntean, M. A. Michels, and A. V. Lyulin, "Myoglobin Interactions with Polystyrene Surfaces of Different Hydrophobicity," *Macromolecular Theory and Simulations*, vol. 23, no. 2, pp. 63–75, 2014. 59, 61

[86] R. A. Richardson, K. Papachristos, D. J. Read, O. G. Harlen, M. Harrison, E. Paci, S. P. Muench, and S. A. Harris, "Understanding the apparent stator-rotor connections in the rotary ATPase family using coarse-grained computer modelling," *Proteins: Structure, Function, and Bioinformatics*, 2014. 65, 175

[87] S. P. Muench, J. Trinick, and M. A. Harrison, "Structural divergence of

the rotary ATPases," *Quarterly Reviews of Biophysics*, vol. 44, no. 03, pp. 311–356, 2011. 65

[88] P. D. Boyer, "The ATP synthase-a splendid molecular machine," *Annual Review of Biochemistry*, vol. 66, no. 1, pp. 717–749, 1997. 66

[89] M. Vollmar, D. Schlieper, M. Winn, C. Büchner, and G. Groth, "Structure of the c14 rotor ring of the proton translocating chloroplast ATP synthase," *Journal of Biological Chemistry*, vol. 284, no. 27, pp. 18228–18235, 2009. 68

[90] T. Meier, P. Polzer, K. Diederichs, W. Welte, and P. Dimroth, "Structure of the rotor ring of F-Type Na+-ATPase from Ilyobacter tartaricus," *Science*, vol. 308, no. 5722, pp. 659–662, 2005. 68

[91] D. Pogoryelov, J. Yu, T. Meier, J. Vonck, P. Dimroth, and D. J. Muller, "The c15 ring of the Spirulina platensis F-ATP synthase: F1/F0 symmetry mismatch is not obligatory," *EMBO reports*, vol. 6, no. 11, pp. 1040–1044, 2005. 68

[92] D. A. Cherepanov, A. Y. Mulkidjanian, and W. Junge, "Transient accumulation of elastic energy in proton translocating ATP synthase," *FEBS Letters*, vol. 449, no. 1, pp. 1–6, 1999. 68

[93] M. Grabe, H. Wang, and G. Oster, "The mechanochemistry of V-ATPase proton pumps," *Biophysical*, vol. 78, no. 6, pp. 2798–2813, 2000. 68

[94] W. Junge, H. Sielaff, and S. Engelbrecht, "Torque generation and elas-

tic power transmission in the rotary FOF1-ATPase," *Nature*, vol. 459, no. 7245, pp. 364–370, 2009. 68

[95] H. Sielaff, H. Rennekamp, A. Wächter, H. Xie, F. Hilbers, K. Feldbauer, S. D. Dunn, S. Engelbrecht, and W. Junge, "Domain compliance and elastic power transmission in rotary F_OF_1-ATPase," *Proceedings of the National Academy of Sciences*, vol. 105, no. 46, pp. 17760–17765, 2008. 68

[96] A. Wächter, Y. Bi, S. D. Dunn, B. D. Cain, H. Sielaff, F. Wintermann, S. Engelbrecht, and W. Junge, "Two rotary motors in F-ATP synthase are elastically coupled by a flexible rotor and a stiff stator stalk," *Proceedings of the National Academy of Sciences*, vol. 108, no. 10, pp. 3924–3929, 2011. 68

[97] A. M. Balakrishna, C. Hunke, and G. Grüber, "The Structure of Subunit E of the Pyrococcus horikoshii OT3 A-ATP Synthase Gives Insight into the Elasticity of the Peripheral Stalk," *Journal of Molecular Biology*, vol. 420, no. 3, pp. 155–163, 2012. 68

[98] R. A. Bernal and D. Stock, "Three-Dimensional Structure of the Intact Thermus thermophilus H^+-ATPase/Synthase by Electron Microscopy," *Structure*, vol. 12, no. 10, pp. 1789–1798, 2004. 68

[99] B. Böttcher, I. Bertsche, R. Reuter, and P. Gräber, "Direct visualisation of conformational changes in EF_0 F_1 by electron microscopy," *Journal of Molecular Biology*, vol. 296, no. 2, pp. 449–457, 2000. 68

[100] D. Matthies, S. Haberstock, F. Joos, V. Dötsch, J. Vonck, F. Bernhard, and

T. Meier, "Cell-free expression and assembly of ATP synthase," *Journal of Molecular Biology*, vol. 413, no. 3, pp. 593–603, 2011. 68

[101] A. G. Stewart, L. K. Lee, M. Donohoe, J. J. Chaston, and D. Stock, "The dynamic stator stalk of rotary ATPases," *Nature Communications*, vol. 3, p. 687, 2012. 68

[102] C. F. Song, K. Papachristos, S. Rawson, M. Huss, H. Wieczorek, E. Paci, J. Trinick, M. A. Harrison, and S. P. Muench, "Flexibility within the Rotor and Stators of the Vacuolar H+-ATPase," *PloS one*, vol. 8, no. 12, p. e82207, 2013. 68, 92

[103] C. L. Lawson, M. L. Baker, C. Best, C. Bi, M. Dougherty, P. Feng, G. van Ginkel, B. Devkota, I. Lagerstedt, S. J. Ludtke, *et al.*, "EMDataBank.org: unified data resource for CryoEM," *Nucleic Acids Research*, vol. 39, no. suppl 1, pp. D456–D464, 2011. 69, 175

[104] J. Schöberl, "NETGEN An advancing front 2D/3D-mesh generator based on abstract rules," *Computing and Visualization in Science*, vol. 1, no. 1, pp. 41–52, 1997. 70

[105] W. Wriggers, R. A. Milligan, K. Schulten, and J. A. McCammon, "Self-organizing neural networks bridge the biomolecular resolution gap," *Journal of Molecular Biology*, vol. 284, no. 5, pp. 1247–1254, 1998. 72

[106] J. N. Stember and W. Wriggers, "Bend-twist-stretch model for coarse elastic network simulation of biomolecular motion," *The Journal of Chemical Physics*, vol. 131, no. 7, p. 074112, 2009. 72

[107] T. Meyer, C. Ferrer-Costa, A. Pérez, M. Rueda, A. Bidon-Chanal, F. J. Luque, C. A. Laughton, and M. Orozco, "Essential dynamics: a tool for efficient trajectory compression and management," *Journal of Chemical Theory and Computation*, vol. 2, no. 2, pp. 251–258, 2006. 72

[108] W. C. Lau and J. L. Rubinstein, "Structure of intact Thermus thermophilus V-ATPase by cryo–EM reveals organization of the membrane-bound VO motor," *Proceedings of the National Academy of Sciences*, vol. 107, no. 4, pp. 1367–1372, 2010. 83

[109] S. P. Muench, M. Huss, C. F. Song, C. Phillips, H. Wieczorek, J. Trinick, and M. A. Harrison, "Cryo–electron microscopy of the vacuolar ATPase motor reveals its mechanical and regulatory complexity," *Journal of Molecular Biology*, vol. 386, no. 4, pp. 989–999, 2009. 91

[110] S. Benlekbir, S. A. Bueler, and J. L. Rubinstein, "Structure of the vacuolar-type ATPase from Saccharomyces cerevisiae at 11-å resolution," *Nature Structural & Molecular Biology*, vol. 19, no. 12, pp. 1356–1362, 2012. 91

[111] W. C. Lau and J. L. Rubinstein, "Subnanometre-resolution structure of the intact Thermus thermophilus H+-driven ATP synthase," *Nature*, vol. 481, no. 7380, pp. 214–218, 2012. 91

[112] J.-P. Sumner, J. A. Dow, F. G. Earley, U. Klein, D. Jäger, and H. Wieczorek, "Regulation of plasma membrane V-ATPase activity by dissociation of peripheral subunits," *Journal of Biological Chemistry*, vol. 270, no. 10, pp. 5649–5653, 1995. 92

[113] B. M. Paschal and R. B. Vallee, "Retrograde transport by the microtubule-associated protein MAP 1C," 1987. 98

[114] R. D. Vale, T. S. Reese, and M. P. Sheetz, "Identification of a novel force-generating protein, kinesin, involved in microtubule-based motility," *Cell*, vol. 42, no. 1, pp. 39–50, 1985. 98

[115] A. J. Roberts, T. Kon, P. J. Knight, K. Sutoh, and S. A. Burgess, "Functions and mechanics of dynein motor proteins," *Nature Reviews Molecular Cell Biology*, 2013. 98

[116] R. Mallik and S. P. Gross, "Molecular motors: strategies to get along," *Current Biology*, vol. 14, no. 22, pp. R971–R982, 2004. 98

[117] A. J. Roberts, B. Malkova, M. L. Walker, H. Sakakibara, N. Numata, T. Kon, R. Ohkura, T. A. Edwards, P. J. Knight, K. Sutoh, *et al.*, "ATP-driven remodeling of the linker domain in the dynein motor," *Structure*, vol. 20, no. 10, pp. 1670–1680, 2012. xvii, 98, 100, 101

[118] S. A. Burgess, M. L. Walker, H. Sakakibara, P. J. Knight, and K. Oiwa, "Dynein structure and power stroke," *Nature*, vol. 421, no. 6924, pp. 715–718, 2003. xvii, 98, 102, 103, 104

[119] M. J. Egan, M. A. McClintock, and S. L. Reck-Peterson, "Microtubule-based transport in filamentous fungi," *Current Opinion in Microbiology*, vol. 15, no. 6, pp. 637–645, 2012. 98

[120] M. P. Koonce, "Dictyostelium, a model organism for microtubule-based transport," *Protist*, vol. 151, no. 1, pp. 17–25, 2000. 98

[121] J. K. Moore, M. D. Stuchell-Brereton, and J. A. Cooper, "Function of dynein in budding yeast: mitotic spindle positioning in a polarized cell," *Cell Motility and the Cytoskeleton*, vol. 66, no. 8, pp. 546–555, 2009. 98

[122] F. J. McNally, "Mechanisms of spindle positioning," *The Journal of Cell Biology*, vol. 200, no. 2, pp. 131–140, 2013. 98

[123] J. A. Johnston, M. E. Illing, and R. R. Kopito, "Cytoplasmic dynein/dynactin mediates the assembly of aggresomes," *Cell Motility and the Cytoskeleton*, vol. 53, no. 1, pp. 26–38, 2002. 98

[124] S. Maday, K. E. Wallace, and E. L. Holzbaur, "Autophagosomes initiate distally and mature during transport toward the cell soma in primary neurons," *The Journal of Cell Biology*, vol. 196, no. 4, pp. 407–417, 2012. 98

[125] R. Kamiya, "Exploring the function of inner and outer dynein arms with Chlamydomonas mutants," *Cell Motility and the Cytoskeleton*, vol. 32, no. 2, pp. 98–102, 1995. 101

[126] H. Sakakibara, H. Kojima, Y. Sakai, E. Katayama, and K. Oiwa, "Inner-arm dynein c of Chlamydomonas flagella is a single-headed processive motor," *Nature*, vol. 400, no. 6744, pp. 586–590, 1999. xix, 101, 120

[127] C. B. Lindemann and K. A. Lesich, "Flagellar and ciliary beating: the proven and the possible," *Journal of Cell Science*, vol. 123, no. 4, pp. 519–528, 2010. 101

[128] D. M. Woolley, "Flagellar oscillation: a commentary on proposed mechanisms," *Biological Reviews*, vol. 85, no. 3, pp. 453–470, 2010. 102

[129] L. N. Trefethen and D. Bau III, *Numerical linear algebra*, vol. 50. SIAM, 1997. 104

[130] K. H. Bui, H. Sakakibara, T. Movassagh, K. Oiwa, and T. Ishikawa, "Asymmetry of inner dynein arms and inter-doublet links in Chlamydomonas flagella," *The Journal of Cell Biology*, vol. 186, no. 3, pp. 437–446, 2009. 105

[131] M. Kikkawa, T. Ishikawa, T. Nakata, T. Wakabayashi, and N. Hirokawa, "Direct visualization of the microtubule lattice seam both in vitro and in vivo.," *The Journal of Cell Biology*, vol. 127, no. 6, pp. 1965–1971, 1994. 117

[132] T. Simonson, "Electrostatics and dynamics of proteins," *Reports on Progress in Physics*, vol. 66, p. 737, 2003. 129

[133] B. Honig and A. Nicholls, "Classical electrostatics in biology and chemistry," *Science*, vol. 268, no. 5214, p. 1144, 1995. 129

[134] P. Mark and L. Nilsson, "Structure and dynamics of the TIP3P, SPC, and SPC/E water models at 298 K," *The Journal of Physical Chemistry A*, vol. 105, no. 43, pp. 9954–9960, 2001. 130

[135] D. Case, T. Cheatham III, T. Darden, H. Gohlke, R. Luo, K. Merz Jr, A. Onufriev, C. Simmerling, B. Wang, and R. Woods, "The Amber biomolecular simulation programs," *Journal of Computational Chemistry*, vol. 26, no. 16, p. 1668, 2005. 130, 134

[136] A. R. Leach, *Molecular Modelling: Principles and Applications (2nd Edition)*. Prentice Hall, 2001. 130

[137] R. Luo, L. David, and M. Gilson, "Accelerated Poisson–Boltzmann calculations for static and dynamic systems," *Journal of Computational Chemistry*, vol. 23, no. 13, pp. 1244–1253, 2002. 131

[138] J. Wang and R. Luo, "Assessment of linear finite-difference Poisson–Boltzmann solvers," *Journal of Computational Chemistry*, vol. 31, no. 8, pp. 1689–1698, 2010. 131, 132

[139] R. Yokota, J. Bardhan, M. Knepley, L. Barba, and T. Hamada, "Biomolecular electrostatics using a fast multipole BEM on up to 512 GPUs and a billion unknowns," *Computer Physics Communications*, 2011. 132

[140] L. Greengard and J. Huang, "A new version of the fast multipole method for screened Coulomb interactions in three dimensions," *Journal of Computational Physics*, vol. 180, no. 2, pp. 642–658, 2002. 132

[141] J. Katsikadelis, *Boundary elements: theory and applications*, vol. 1. Elsevier Science, 2002. 132, 153, 154

[142] G. Paulino and L. Gray, "Galerkin residuals for adaptive symmetric-Galerkin boundary element methods," *Journal of Engineering Mechanics*, vol. 125, no. 5, pp. 575–585, 1999. 132

[143] A. Boschitsch and M. Fenley, "Hybrid boundary element and finite difference method for solving the Nonlinear Poisson–Boltzmann equation," *Journal of Computational Chemistry*, vol. 25, no. 7, pp. 935–955, 2004. 132

[144] J. Bardhan, M. Knepley, and M. Anitescu, "Bounding the electrostatic free

energies associated with linear continuum models of molecular solvation," *The Journal of Chemical Physics*, vol. 130, p. 104108, 2009. 132

[145] S. Salon and J. Schneider, "A hybrid finite element-boundary integral formulation of Poisson's equation," *Magnetics, IEEE Transactions on*, vol. 17, no. 6, pp. 2574–2576, 1981. 132

[146] W. Xin and A. Juffer, "A boundary element formulation of protein electrostatics with explicit ions," *Journal of Computational Physics*, vol. 223, no. 1, pp. 416–435, 2007. 132

[147] E. Yap and T. Head-Gordon, "New and Efficient Poisson–Boltzmann Solver for Interaction of Multiple Proteins," *Journal of Chemical Theory and Computation*, 2010. 133

[148] D. Edmonds, N. Rogers, and M. Sternberg, "Regular representation of irregular charge distributions," *Molecular Physics*, vol. 52, no. 6, pp. 1487–1494, 1984. 134

[149] D. Kershaw, "The incomplete Cholesky–conjugate gradient method for the iterative solution of systems of linear equations* 1," *Journal of Computational Physics*, vol. 26, no. 1, pp. 43–65, 1978. 134

[150] M. Holst and F. Saied, "Multigrid solution of the Poisson–Boltzmann equation," *Journal of Computational Chemistry*, vol. 14, no. 1, pp. 105–113, 1993. 134, 136

[151] A. Brandt, *Multi-level Adaptive Techniques (MLAT): I. The Multi-grid Method*. IBM Thomas J. Watson Research Center, 1976. 134, 135

[152] S. McCormick and J. Thomas, "The Fast Adaptive Composite grid (FAC) method for elliptic equations," *Mathematics of Computation*, vol. 46, no. 174, pp. 439–456, 1986. 134, 135

[153] J. E. Bresenham, N. D. Butler, and A. C. Gay, "Line generation in a display system," Feb. 26 1991. US Patent 4,996,653. 135

[154] B. Lu, D. Zhang, and J. McCammon, "Computation of electrostatic forces between solvated molecules determined by the Poisson–Boltzmann equation using a boundary element method," *The Journal of Chemical Physics*, vol. 122, p. 214102, 2005. 141

[155] M. Khayat and D. Wilton, "Numerical evaluation of singular and near-singular potential integrals," *Antennas and Propagation, IEEE Transactions on*, vol. 53, no. 10, pp. 3180–3190, 2005. 149

[156] E. Lutz, "Exact Gaussian quadrature methods for near-singular integrals in the boundary element method," *Engineering Analysis with Boundary Elements*, vol. 9, no. 3, pp. 233–245, 1992. 149

[157] B. Lu, X. Cheng, J. Huang, and J. McCammon, "Order N algorithm for computation of electrostatic interactions in biomolecular systems," *Proceedings of the National Academy of Sciences*, vol. 103, no. 51, p. 19314, 2006. 156

[158] H. Van der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, p. 631, 1992. 158

[159] A. Faruqi and R. Henderson, "Electronic detectors for electron microscopy," *Current Opinion in Structural Biology*, vol. 17, no. 5, pp. 549–555, 2007. 177

[160] W. Kühlbrandt, "The Resolution Revolution," *Science*, vol. 343, no. 6178, pp. 1443–1444, 2014. 177

[161] M. Kikkawa, "Big steps toward understanding dynein," *The Journal of Cell Biology*, vol. 202, no. 1, pp. 15–23, 2013. 177

[162] C. Cibert, J. Toscano, V. Pensée, and G. Bonnet, "Bending of the "9 + 2" axoneme analyzed by the finite element method," *Journal of Theoretical Biology*, vol. 264, no. 4, pp. 1089–1101, 2010. 177