

# Generalised Multi-level Coarse-grained Molecular Simulation and its Application to Myosin-V Movement

William R. Taylor\*, and Zoe Katsimitsoulia

Division of Mathematical Biology,  
MRC National Institute for Medical Research,  
The Ridgeway, Mill Hill, London NW7 1AA, U.K.

\* corresponding author: wtaylor@nimr.mrc.ac.uk

## **Keywords:**

Myosin V, coarse-grained simulation, structural hierarchy

May 26, 2011

# 1 Introduction

Physical methods for the determination of macromolecular structure have improved greatly over the past decade, resulting in a wealth of large molecular structures including both protein and nucleic acid. Most, if not all of these structures are dynamic, either in their assembly (and disassembly) such as cytoskeletal fibrils and viruses or often have large motions associated with their function, such as the F1/F0 ATP synthetase complex. Indeed for the class of molecular motors, motion is the essential aspect of their function [Spudich, 1994, Astumian, 2001].

With the exception of real-time scanning atomic force microscopy [Kodera *et al.*, 2010], which is still not a widely used method, most of the physical techniques, such as electron microscopy or X-ray crystallography, provide only snap-shots of discrete states of a dynamic system [Coureux *et al.*, 2004, Bershitsky *et al.*, 2009]. This can be augmented by data from a wide variety of physical methods, such as spectroscopy or single particle measurements [Forkey *et al.*, 2003, Veigel *et al.*, 2005], that can provide dynamic information against which the different 'snap-shots' can be interpreted. However, given the coordinates for different states, it is still natural to want to visualise how the system progresses from one to the other, or given a set of components, to wonder how they might assemble into a complex.

An impression of how complex systems behave can be gleaned by interpolating between states using a physics-based 'morphing engine' [Krebs & Gerstein, 2000], similar to those used in animating cartoon characters in popular games and movies. Depending on the constraints of the data, these can to some extent be predictive but more typically, simply recapitulate the relationship between the existing structures in a more accessible manner. At the other extreme to this top-down approach, molecular dynamics methods can be used to animate large structures from the bottom up: that is by summing the interactions of individual atoms into increasingly larger motions [Schlick, 2002, Burghardt & Hu, 2007, Kawakubo *et al.*, 2009].

Both the top-down and bottom-up approaches have their problems. The top-down approach may not have a geometry engine of sufficient quality to ensure that the system does not wander into un-natural configurations between known states, which is a problem if there is a large difference between the states (or just one known state). Conversely, with the bottom-up approach, the repeated summation of small atomic forces into large motions is likely to be a divergent process, giving little confidence in the uniqueness of the final configuration.

The ideal solution to both these problems is to combine the two approaches. However, this raises a new set of questions as to the best way that this can be done and there is a variety of approaches depending on whether the aim is mainly to predict or recapitulate. This type of modelling is broadly referred to as coarse-grained (CG) as the constraints from the data are imposed at a level above atomic interaction [Izvekov & Voth, 2005, Zhang *et al.*, 2009, Taylor & Katsimitsoulia, 2010a], typically in the form of distance-based elastic constraints between neighbouring units [Parker *et al.*, 2009, Zheng, 2011] (being two examples from the actin/myosin motor discussed below). The definition of levels is essentially arbitrary but for proteins and RNA there is a good succession in which each level incorporates roughly an order of 10 elements from the lower level giving a progression from atoms to residues/bases to secondary structures to domains to multi-domain/chain macromolecules or assemblies. This follows the conventional structural hierarchy of primary, secondary, tertiary and quaternary with the allowance that in the latter, it is often arbitrary whether the units are linked as domains or exist as distinct chains [Taylor & Aszódi, 2005].

As structural constraints are imposed at levels above atomic, they become less general until the highest level may be specific to a particular known state of a large system. For example: atomic interactions are the same in all molecules, whether nucleic acid or protein. Interactions at the monomer level require knowledge of the chemical structure of four bases for RNA/DNA and twenty amino acids for proteins. At the secondary structure level, both RNA and protein be-

come very different and the interaction (packing) of the secondary structures can only be described because of the large numbers of structures in which they have been observed. The interaction has shifted from being fundamental to being empirical. At the domain level constraints on the fold of the chain must be derived largely from a specific fold and at the top level of a multi-domain chain or multi-chain assembly, constraints derive from a unique structure.

If only one of these levels of constraint is utilised, then the others become poorly determined (or must be kept fixed). As discussed above, using only atomic interaction leads to uncertainty at the higher levels and using just high-level information (say, domain/domain distances) leaves the atomic detail unspecified. Taken by itself, any intermediate level will leave those above and below less well constrained. The ideal solution clearly is not to neglect any level but to impose constraints at every level, ranging from the generic at the lowest level to the specific at the highest. This is sometimes referred to as multi-level modelling. Shifts from one set of high-level constraints to another can then be used to model the transition from one state of the system to another while preserving integrity through the lower levels.

In this chapter, we will give an overview of a new generalised coarse-grained multi-level simulation method that we have developed along the lines described above with a view to simulating the motion of large macromolecular systems and, in particular, molecular motors.

## 2 Implementation

### 2.1 Overview

As outlined in the introduction, the nature of macromolecular structure data is clearly hierarchic so it is natural to turn to a hierarchic data structure to capture it. With a view to generality, it was our aim that the structure of each unit in the hierarchy at any level should be identical with the distinction between levels being made only through externally specified (user) parameters. The most fundamental aspects to be specified include the shape of the unit, its linkage to other units (such as whether it is in a chain or multiply bonded) and how units interact with each other. For ease of description, a unit contained in a higher level unit will be called its "child" with the higher level referred to as the "parent" of the lower level unit. Units with the same parent are therefore "siblings" and if they form a chain, are designated "sister" (preceeding) and "brother" (following). Anthropomorphic terms based on family relationships will be used through.

Interactions can be distinguished as inter- and intra-level. Within a level (intra) these will either be repulsive (active for a pair of units in collision) or attractive (active between a bonded pair). However, there are no bonds between levels (unless individually specified by the user) and no bumps between levels. Inter-level interaction consists only of coordination of motion and containment. Coordinated movement means that when a parent moves, all its children move too, which by implication, propagates down through all successive generations. In the opposite direction, the centroid of the children determines the position of the parent. This relationship also implies indirect interaction between the levels so that when children in different families collide then their parents will also experience a lesser repulsion. Containment was implemented as a simple kick-back to any children that had 'strayed' beyond the shape boundary of their parent.

The motion of each unit is purely random with a fixed step-size defined in the user-specified parameter file. Every move is accepted whether it violates bond geometry or leads to collisions [Katsimitsoulia & Taylor, 2010]. Clearly this would lead to a degradation in the molecular structure so independently of this imposed motion, the bond lengths and local chain geometry (if there is a chain) are continually refined towards their initial configuration. Similarly, units in collision are also corrected. All these processes run concurrently (implemented as separate threads) and in addition a user specified process also runs in parallel in which the directed elements of the model are specified. All of these processes operate on a single representation of the coordinates so care has been taken to ensure that undefined states do not arise in one process that would disrupt another. In general, this can be avoided with each process working on a temporary copy of the coordinates it needs then writing these in one step back into the structure.

The overall structure of the implementation is shown in Figure 1 along with the names of the processes that will be referred to below. The two processes that maintain the integrity of the molecular structure are the collision detection and correction process and the process that maintains the specified links in the chain: respectively, called the **bumper** and the **linker** which will be considered first.

## 2.2 Steric exclusion

It is intended that the implementation should be applied to very large systems so any collision detection based on a full pairwise algorithm would be impractical. This is commonly avoided in molecular (and other) dynamics programs through the use of a neighbour list in which each atom maintains a list of its current neighbours and checks only these for collision. This has the unavoidable problem that the list must be revised periodically. We adopted a similar approach except that we used the hierarchic structure of the data to provide

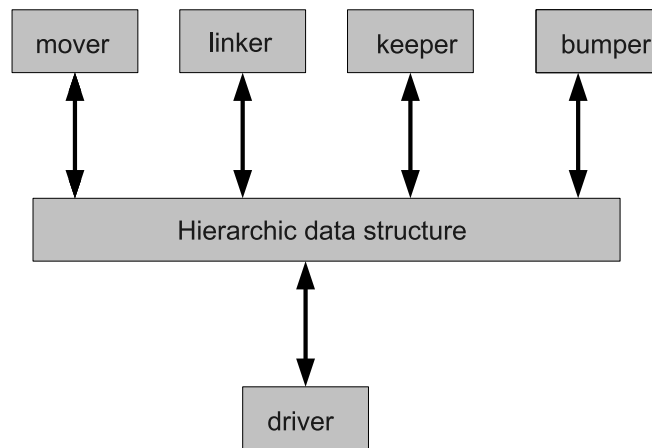


Figure 1: The structure of the program is viewed as independent processes acting on a single common **hierarchic data structure**. The **mover** routine gives every object a random displacement. The **linker** routine maintains bonded distances and cross-links. The **keeper** routine restrains children to stay inside their parents. The **bumper** routine detects collisions and corrects steric violations. The **driver** is a user supplied routine that implements the application specific aspects of the dynamics. Each routine is independent and runs as a parallel process (threads) interacting only with the common data.

built-in neighbour lists where any unit only checks its siblings for collisions. As each family is usually in the order of 10–100 units, this would not be a large task to compute in a pairwise manner, however, we use a faster approach based on ranked lists of units that are maintained for each dimension (X,Y,Z). As sorting avoids quadratic operations in the number of objects, this is much faster for large families [Taylor & Katsimitsoulia, 2010a].

When two units are in collision, they are repelled only if they have no children. Again a single (user specified) kick-back step is applied equally to each unit along the line connecting their centres. If the units have children, then collisions are found between their joint families within a restricted range along the line connecting the two parent centres. If there are no grandchildren, repulsion is again implemented along the line connecting their two centres, otherwise the process is repeated at each lower level until the lowest atomic<sup>1</sup> level is reached.

As mentioned above, parents of different families will automatically adjust their position indirectly to the repulsion between their families as the centroid of the children will move slightly apart. This is often a small effect, and before it becomes significant, the children can become bunched at the collision interface which has the reverse effect of bringing the parents (and hence their children) even closer together. To avoid this, on the return path from the traversal of the family hierarchy (ie: revisiting the parents of the colliding children), the parents themselves are given a small direct displacement proportional to the number of their children that were in collision. If only the positions of the atomic level units were observed, this would have the appearance of a repulsive 'field' as there would seem to be "action at a distance" across a family. Alternatively it can be imagined that the children are embedded in a soft parental jelly-like matrix. Computationally the approach means that in any collision at a high level, there will be fewer low-level collisions generated which saves on computational expense.

---

<sup>1</sup>The term "atomic" only means the lowest level in the hierarchy, which in the protein applications discussed, is the residue level (based on the  $\alpha$ -carbon position).



This approach to the treatment of collisions has an additional effect in that it is relatively insensitive to the shape of the colliding objects, which in our implementation can be spheres, ellipsoids or tubes (discussed in more detail below). If we assume that the atomic level consists of spheres, then different shapes at higher levels are primarily defined by the distribution of their children within them which in turn is determined by the shape within which the family is confined. The only discrepancy comes through the point at which units at the parental level detect collision as, for computational simplicity, this is kept as an isotropic test at the radius of their (user defined) bumping sphere. Objects that are long-thin tubes or extremely prolate ellipsoids which are fully contained in a bumping sphere can therefore be considered in collision before any of their family members come in contact. At worst, however, this is just a slight waste of computer time as the count of colliding children will be zero and the parents will not react.

## 2.3 Polymers and Cross-linking

### 2.3.1 Specifying chain connectivity

Unless liquids are being modelled, the links between units (which are not distinguished from bonds) are the components that impart greatest structure. For biological polymers, links between just adjacent units along a chain, combined with steric exclusion, are sufficient to define a basic model. However, even this, apparently simple, imposition of structure leads to complications in a hierarchic model. If the atomic level is a linear chain, then so too are all higher levels but this is not so if each atomic family forms a separate or a circular chain. Then higher levels can be unlinked (eg; a liquid of cyclic peptides [Taylor *et al.*, 2011]) or otherwise linked in their own way. The linkage polymer state of each level can be specified by the user independently but chain interdependencies are checked internally and imposed by the program.

The structural hierarchy (family structure) is not determined by chain con-

nectivity but should be based on groups of units that will tend to move together as they are all acted upon by their parent's transform operations. At the lower level, such groupings will typically consist of consecutive units along a chain (such as an  $\alpha$ -helix), however, at the domain level and secondary structure level for RNA, families of units will also be composed of sequentially discontinuous segments which are pre-calculated automatically [Taylor, 1999]. Computationally, this requires some book-keeping to keep note of which members are linked across families. To facilitate this, each unit holds a record of its preceding unit, referred to as its "sister", and its following unit, referred to as its "brother"; both of which may specify any unit in any family at the same level. Together, the sisters and brothers constitute two linked-lists, with brothers running from the start of the chain to the end and sister in the opposite direction.

As chain connectivity is not restricted by family groupings, its path at the next higher level is not necessarily linear and can be branched. This means that each unit may have more than one brother or sister which is equivalent to branches in the chain in both directions. In general, this specifies a network and to deal with the associated "book-keeping", each unit holds a stack of its brothers and sisters. Following a chain is therefore not simple and when listing a chain in "sequential order", the lists of brothers and sisters are followed recursively from any given starting unit.

### **2.3.2 Inter-chain cross-links**

Polymer chain links are such a common feature of biological macromolecules that the capacity to encode them was included as a general feature in the data structure of each unit. Inter-chain cross-links, which are less ubiquitous, were allocated only as requested by the user in the data file that specifies the model. For any level, a fixed number of links could be specified, not all of which need necessarily be used. If no linking capacity is specified then computer memory is

not allocated.

Although links can be individually specified, some automated features were incorporated to ease the burden of assigning the local cross-links associated with secondary structure, both in proteins and RNA. For proteins, two types of secondary structure can be defined: the  $\alpha$ -helix and  $\beta$ -sheet. The former is purely local and two links were automatically set to the relative chain positions +4 and -3 of the ideal length found in proteins. Similarly, two local links were made along a  $\beta$ -strand to the +2 and -2 positions. However, each strand in a sheet makes non-local (hydrogen-bonded) links which can be specified by data provided in the coordinate input specification. Again, these links are automatically generated by a separate program that calculates the definition of secondary structures [Taylor, 2001]. Similarly, for RNA, the base-pairings are pre-calculated by a separate program [Hofacker, 2003].

At a higher level, links between secondary structures can be allocated within a given range of interaction. Unlike the secondary structures which are pre-calculated, these links are computed as part of the data input process.

## 2.4 Geometric regularisation

Steric exclusion combined with the range of linkage described above can generate a relatively stable structure. However, given the background of "thermal" noise generated by **mover**, any less constrained parts of the structure will be free to diverge from their starting configuration under the given distance constraints. Typically, this involves twisting and shearing that can generate large motions with little violation of the specified distances, which in principle, cannot constrain chirality. A general mechanism, based only on local angles and distances was provided to reduce these distortions and was applied equally to all levels that form a chain.

In a chain segment of five units (designated: b2,b1,c0,a1,a2), six distances

were recorded from the starting configuration in the upper half of the matrix of pairwise distances excluding adjacent units. Three angles were also recorded as b1-c0-a1 and the torsion angles around b1-c0 and c0-a1. These local distances were continually refined as were the angles. Distances can be regularised with little disruption, however, refining torsion angles can sometimes lead to an error propagation with dramatic effects. To limit the potential for this the torsion angles were dialled-up exactly to generate new positions for b2 (b2') and a2 (a2'). These were used to form a basis-set of unit length vectors along:  $x = a2' - b2'$ ,  $y = c0 - (a2' + b2')/2$ , with  $z$  mutually orthogonal. Starting from the centroid of the five points, the coefficients of an equivalent basis-set defined on the original positions were applied to the new basis-set to generate the new coordinate positions. The result is a compromise between angle and position that remains stable over repeated application.

Although only local information is used, its application over all levels leads to a global effect and indeed is sufficient by itself to recapitulate a large structure. As the procedure was designed to correct defects caused by the addition of random motion (by **mover**), it was not implemented as an independent parallel process but included in **mover** and applied after the coordinates had been displaced, so keeping a balance between disruption and correction. A final feature was included to allow for the necessary requirement that in a dynamic model, the starting configuration of the structure should not be exclusively maintained. This was accommodated by periodically shifting the target distances and angles towards those found in the current configuration. The overall effect of this procedure is to provide a buffering effect against random motion and is similar to giving rigidity to the structure but still allowing movement under a persistent "force". In the current implementation, this shift is by 1% once in roughly every 100 activations of **mover**. This can be adjusted depending on the application.

## 2.5 Shape specification

Three basic shapes were implemented: cylinder, ellipsoid and sphere. Although the sphere is a special instance of an ellipsoid, there are implementation details, described below, that make them distinct. Each shape type by itself has elements of symmetry that can make their orientation arbitrary, however, this symmetry is broken when a unit contains children in an irregular configuration. Thus each unit needs to have an associated reference frame that determines its orientation and is acted on by rotational operations. For a unit in a chain, the current reference frame is based on the direction from its sister to brother (X) with the Y direction as the projection of the unit's position onto this line in an orthogonal direction and Z as their mutual perpendicular. A consequence of this is that flexing of the chain does not preserve the end-point distances between consecutive cylinders or ellipsoids along the chain.

The length of cylinders and ellipsoids is set by reading in two end-points from the coordinate input data which have been pre-calculated from the inertial axes of the point-set that comprises the current unit (say, a secondary structure element or a domain) [Taylor *et al.*, 1983]. As well as the length, the line linking these end-points specifies the axis that corresponds with the X direction in the internal reference frame. The two end-points are then set within the data-structure that defines each unit as two points equidistant from the central point along the X direction. While the length of a unit is determined by these end-points, this is different from the size of each unit which is set generically for every unit on a given level by a value specified in the parameter file that describes the model. For a sphere, this is the only value that is needed and specifies the radius. For a cylinder, it also specifies the radius which is the thickness of the tube. For an ellipsoid, the end-points specify the length along the X axis and the size parameter specifies the other two axes. Therefore all ellipsoids are radially symmetric around X, giving a progression from oblate (disc) through spherical to prolate (cigar).

Ignoring scalene ellipsoids excludes only long flat discs which are not common shapes for secondary structures or domains [Aszódi & Taylor, 1994].

## **2.6 Implementation**

### **2.6.1 Time and memory allocation**

The adoption of a common data structure for each node in the hierarchy can lead to the allocation of memory for variables that are seldom, if ever, used. For example; the data structure allows for a general shape type which includes the coordinates of the end-points for tubes and ellipsoids yet if the object is a sphere, which it commonly is on the most populated atomic level, then space is wasted. Fortunately, with a reasonable workstation or laptop, memory is seldom a limitation for the system and tests have been made using over a million allocated nodes.

With a parallel implementation (using threads), the time allocated to the different processes can present scheduling considerations. A simple solution was adopted in which the call-back loop of each process was interleaved with a sleep call which suspended the process for a fixed period of time (currently 0.1 sec.). Within each process, higher priority was allocated to branches in the hierarchy that were in an active state, such as undergoing collision or close to a component that had been selected as being of special interest (such as the myosin molecule in the example considered below) or close to the user viewpoint if the system is viewed interactively.

### **2.6.2 Visualisation**

Objects were visualised in a simple viewer with all levels except the atomic being rendered as transparent according to the shape they had been given. Objects in a chain were linked by a thin tube which for spheres ran along the centre-centre direction and so was always normal to the spherical surface. For cylinders and

ellipsoids, the linker tube ran from a sphere placed on the end-points, which for cylinders had the same radius as the cylinder (producing sausage-like objects) and for ellipsoids, was only slightly larger than the linker tube. This provides a visual distinction between spheres and spherical ellipsoids.

### 3 Example Application

The program described in the previous section corresponds to a general geometry engine but does not contain any implementation of forces that are more typically found in molecular dynamics (MD) or Monte Carlo (MC) programs. As it stands, the objects simulated will simply shake and diffuse in a random way under the constraints of the imposed steric exclusion, linkage constraints and shape envelope confinement. For anything of biological interest to occur, such as a transition from one state to another, then high level directions must be imposed. These can range from being completely determined, where the starting and final states are fully specified and only the pathway between is of interest or the change can be applied to just a part of the system and the interest is to see how the full system adjusts.

To illustrate how such a system can be implemented in more concrete terms, we will take the example of a molecular motor, specifically myosin-V on an actin filament [Mehta *et al.*, 1999, De La Cruz *et al.*, 1999, Forkey *et al.*, 2003], in which large directed changes are applied to a component of the system (myosin) to drive it from one defined (bound) state to another then back to the original state but bound to a different actin molecule along the actin filament [Kodera *et al.*, 2010]. The system will be introduced in two parts: firstly, by describing how the model was set-up as a hierarchical data structure, then secondly, how the dynamics were introduced. The first part is done through data files that have been alluded to in the previous section, however, the second part requires direct run-time interaction with the simulation and this is implemented as a specific user-defined routine called the **driver** (Figure 1) which interacts only with the common data structure and executes as an independent parallel process.



### 3.1 Model Construction

We will introduce the data-structure from the top down, starting with file (`actmyo.run`) that specifies the two main components: a myosin-V dimer and the actin filament.

---

```

                                actmyo.run
PARAM myosin.model
PARAM actin.model
END
GROUP 2
MODEL 0
INPUT myosin.dimer.dat
MODEL 1
INPUT actin.linear.dat

```

---

The above run file directs the INPUT from two files for the myosin dimer (`myosin.dimer.dat`) and an actin filament (`actin.linear.dat`) that constitute two units (GROUPs) at the highest level. Each group is preceded by a specification of the parameter set (MODEL) that they will use (0 and 1) which corresponds to the PARAMeter files: `myosin.model` and `actin.model`, respectively. These files consist of columns of numbers with each column specifying the values for the different parameters at each level in the hierarchy. The file `myosin.model` consists of seven columns:

---

```

                                myosin.model
0,      0,      0,      3,      1,      2,      1
9999, 1000,  500,  140,   70,   24,   5
0,    100,   100,   50,   20,   20,  11
0,      0,      0,      1,      2,      6,   4
0,      0,      0,      1,      3,      1,   1
0,      0,      1,      1,      1,      1,   1
0,      1,      1,      1,      1,      1,   0
0,      0,      1,      5,      5,      5,   5
0,      0,      0,      1,      0,      0,   3

```

---

where the first is the state of the 'world' and the following six define levels in the hierarchy. For example; the top line specifies the shape associated with

each level with: 1=sphere, 2=cylinder, 3=ellipsoid and 0 being a virtual sphere that is not rendered. The following two lines specify the size of each object and its bumping radius (both in arbitrary units). The equivalent lines for the `actin.model` file show a slightly different structure using different values:

---

actin.model (part)

---

```

0,      0,      1,      3,      1,      2,      1
9999, 3000,  120,  120,   70,   24,   5
0,   400,  100,   0,   70,   20,  11
:
```

---

### 3.1.1 Myosin

Considering firstly the myosin model, the file `myosin.dimer.dat` contains another two GROUPs both specified by the file `myosin.dat`:

---

myosin.dimer.dat

---

```

GROUP 2
SPINY 180.0
TRANS  55.0 -250.0 -450.0
INPUT myosin.dat
TRANS -55.0 -250.0 -450.0
INPUT myosin.dat
```

---

The first INPUT is preceded by two geometrical transforms in which the contents of the file are rotated 180° about the Y axis (SPINY) then translated (TRANS x y z). The second file is only translated but in the opposite direction along X producing two copies related by a twofold axis, as seen in the X-ray crystal structure (PDB code: 2DFS) [Liu *et al.*, 2006a]. (Figure 2).

The file `myosin.dat` contains two components of the myosin molecule which are referred to as the "head" and the "tail" and are set up as two GROUPs in what is now the third level in the hierarch. From the parameter file `myosin.model`

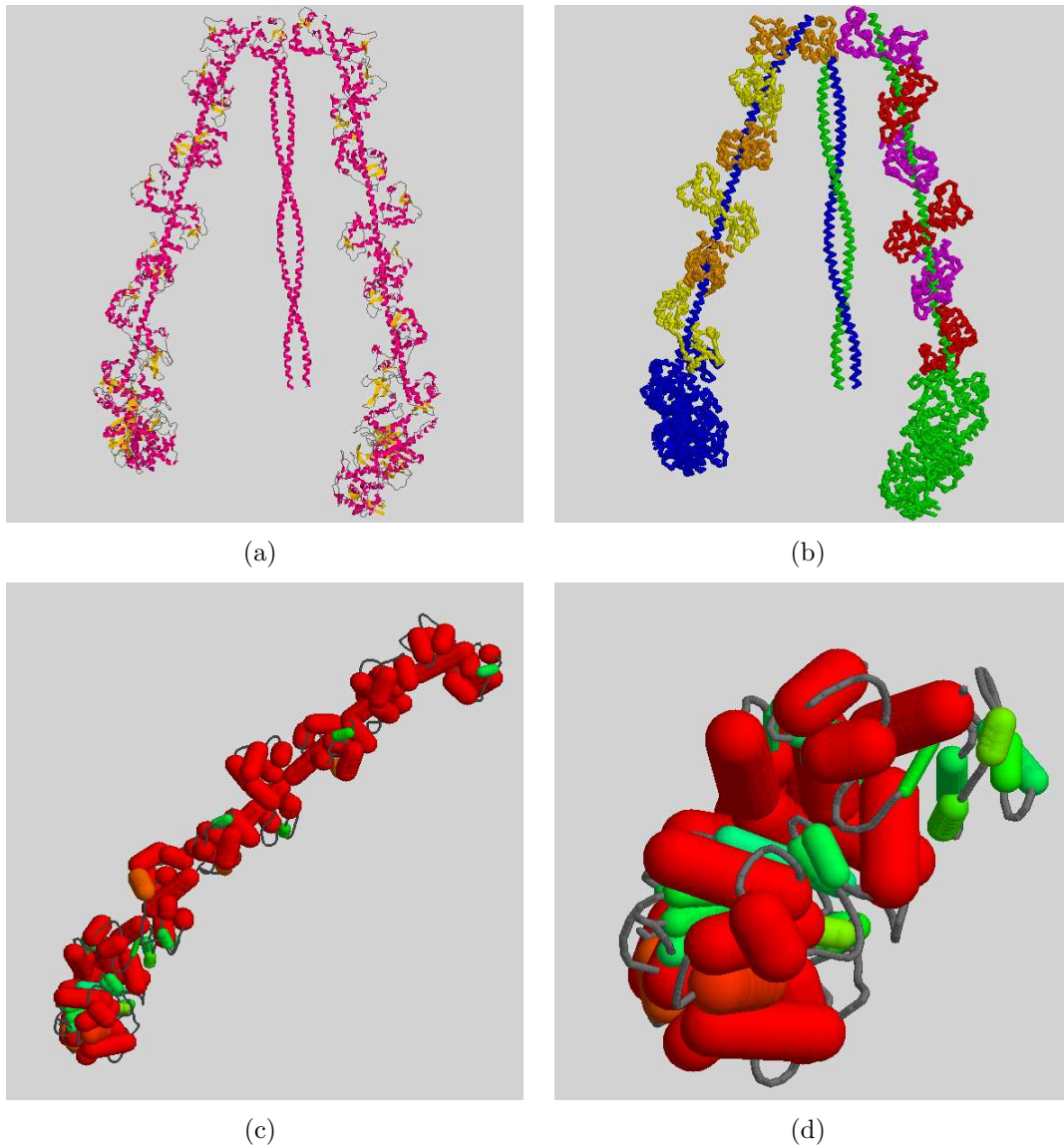


Figure 2: **Myosin-V structure.** The structure of dimeric myosin-V (2DFS), determined by single-particle cryo-electron microscopy is shown (a) with secondary structures represented in cartoon style (using RASMOL) with  $\alpha$ -helix coloured pink and  $\beta$ -strands yellow. (b) The same structure is shown as a virtual  $\alpha$ -carbon backbone with the two heavy chains coloured cyan and green and their associated light-chains in alternating yellow/orange and red/magenta, respectively. Secondary structure line-segments ("sticks") are shown as green tubes for  $\beta$ -strands and thicker red tubes for  $\alpha$ -helices. (c) For the full structure of heavy and light chains (excluding the coiled-coil C-terminus) and (d) for the globular foot domain. (The amino-terminus lies in the all- $\beta$  SH3 domain to the right)

it can be seen (col:4) that these objects are the first to have a defined shape and will behave and be rendered as ellipsoids.

---

```

                                myosin.dat
GROUP 2
TRANS  -31.0 136.5 59.0
INPUT  myosin.head.dat
INPUT  myosin.tail.dat

```

---

The file `myosin.head.dat` specifies the structure of the myosin head-group which is the globular kinase domain of the protein while the file `myosin.tail.dat` specifies the extended alpha-helical tail with its associated calmodulin-like light chains. (Sometimes referred to as the lever-arm or the "leg"). Although these files contain coordinate data from the same protein structure (2DFS), they have been processed separately which put their centroid to the origin and a translation (TRANS) was applied to the head group to restore their proper relative positions. By specifying the head and the tail as separate groups (units), they can be operated on independently by geometric transforms allowing the **driver** routine to recreate a large relative motion between them called the "power-stroke" in which the tail swings through a large angle. (Discussed below).

The myosin head group was split into seven domains, as described previously [Taylor & Katsimitsoulia, 2010b], all of which are contained in distinct files: `head.dom[1-7].dat`. The level-3 unit that contains the domains was defined as an ellipsoid and its end-points are defined on the GROUP definition line (along with the number of children it contains).

---

```

                                myosin.head.dat
GROUP 7  -1.2 -42.6 -37.2    4.32 34.44 39.0
INPUT  head.dom1.dat
INPUT  head.dom2.dat
INPUT  head.dom3.dat
INPUT  head.dom4.dat
INPUT  head.dom5.dat

```

---

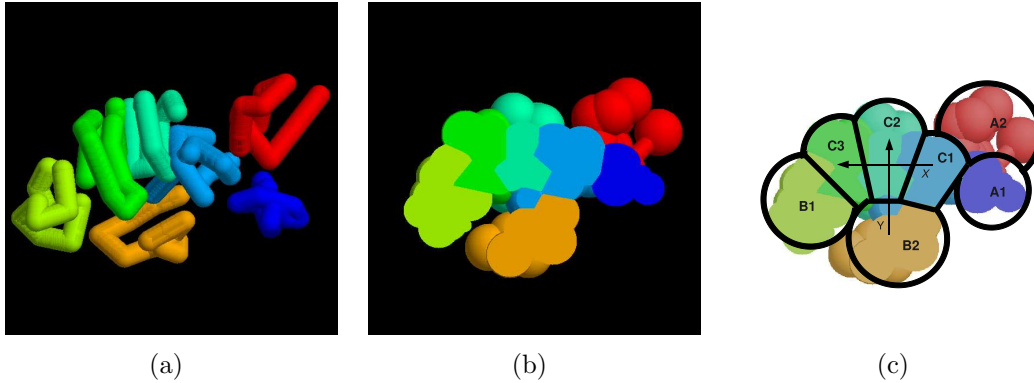


Figure 3: (a) Secondary structure elements (SSEs) allocated to produce seven domains in the myosin head structure. (b) The SSEs are redrawn with space-filling spheres placed on each SSE end-point and sliced through by a plane that would also contain the tail (or leg) structure. (c) As in part b but with domain boundaries sketched and domain codes added: A1-2 (ankle), B1-2 (binding) and C1-3 (core). The direction of the X and Y axes of the internal reference frame are indicated by arrows.

```

INPUT head.dom6.dat
INPUT head.dom7.dat
REBOND 134 169
REBOND 275 320
REBOND 411 429
REBOND 495 412
REBOND 428 276
REBOND 292 541
REBOND 667 496
REBOND 540 293
REBOND 319 135
REBOND 168 668

```

---

The path of the chain through the seven head-group domains does not correspond simply to the domain order. Since each domain group is defined automatically by compactness, (giving sets of units that should move together), it is necessary to specify the chain path through the domains. This is done using the identity of units at the atomic level, sequentially numbered over the scope of the current group. Each rewiring of the units is specified by a REBOND command which states that the unit with the first identity number should be bonded to the

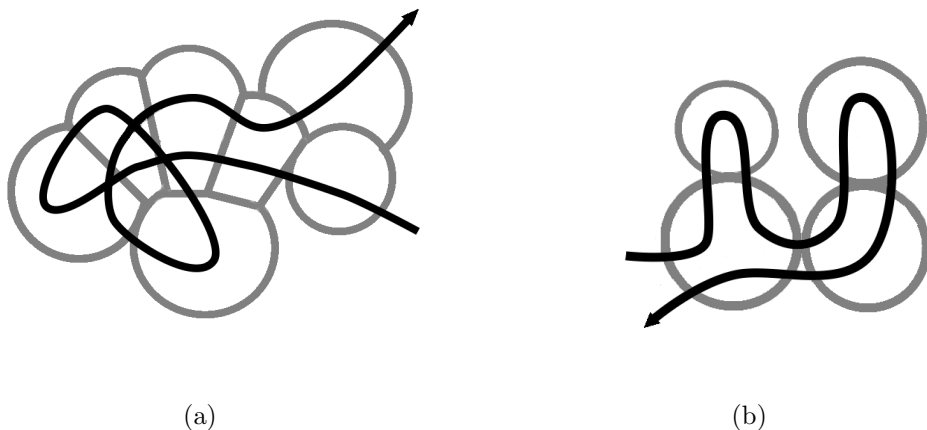


Figure 4: (a) The path of the chain through the seven myosin head domains (grey segments as in Figure 3) is indicated by a black line with an arrow on the carboxy terminus (where the chain enters the tail segment). (b) The path of the chain through the four actin domains is shown in the same way. Although much simpler than the myosin, the chain still creates a branched (Y-shaped) topology at the domain level.

unit with the second identity number. For example; "REBOND 134 169" specifies that residue 134 should now link to residue 169. The resulting loose end at 135 is picked-up by the later command "REBOND 319 135". These connections can be specified "by-hand" but are written automatically by the preprocessing program that defines the domains. (Figure 4).

Each domain file now takes us down to the lowest (atomic) level at which the actual X,Y,Z, coordinates are encountered. To avoid the proliferation of many small files, the two lowest levels (secondary structures and residues) are defined together with the first GROUP command stating that there are nine secondary structures in the group and the second (lower level) GROUP command specifying six ATOMs (atomic-level units) in each subgroup. The coordinate data (from the PDB file) is given at the atom level along with the secondary structure state encoded in the final column as: 1=loop, 2=beta and 3=alpha. As the secondary structures are defined in the parameter file as cylinders, they are given end-point

coordinates on the GROUP command line. If these are omitted, as with the first loop segment, default end-points are generated inside the program.

---

```

head.dom1.dat
GROUP 9
GROUP 6
ATOM      1  CA  GLY A   1      -3.537 -37.198 -20.771  1.00  1.00
ATOM      2  CA  GLY A   2      -1.470 -34.014 -20.628  1.00  1.00
ATOM      3  CA  GLY A   3       1.589 -35.231 -18.657  1.00  1.00
ATOM      4  CA  GLY A   4       3.165 -37.036 -21.663  1.00  1.00
ATOM      5  CA  GLY A   5       6.875 -37.568 -22.449  1.00  1.00
ATOM      6  CA  GLY A   6       8.601 -34.333 -23.626  1.00  1.00
GROUP 7      6.15 -31.57 -21.01  12.09 -24.98 -5.29
ATOM      7  CA  GLY A   7       6.079 -32.117 -21.857  1.00  2.00
ATOM      8  CA  GLY A   8       7.638 -29.779 -19.295  1.00  2.00
ATOM      9  CA  GLY A   9       6.751 -28.831 -15.716  1.00  2.00
ATOM     10  CA  GLY A  10       7.995 -26.626 -12.874  1.00  2.00
:
SHEET
BETA     44 36
BETA     43 37
BETA     37 43
BETA     36 44
BETA     36 23
BETA     23 36

```

---

The links between  $\beta$ -strands in a SHEET are specified as pairings on the BETA records at the end of the file.

The structure of the tail component follows along similar lines and will not be described in detail.

### 3.1.2 Actin

The highest level actin specific file, `actin.linear.dat` describes a segment of an actin filament which consists of repeated actin molecules related by helical symmetry with a rotation of  $-167^\circ$  and  $55\text{\AA}$  translation [Holmes, 2009]. Because of the way they interact with the myosin, actins were taken in pairs (called a dimer)

giving a shift between dimers of  $23^\circ$  and  $110\text{\AA}$  translation<sup>2</sup>. These relationships could be encoded by separate rotate and translate commands but as helical symmetry is common, a combined HELIX command was created specifying the two components together:

---

```

                                actin.linear.dat
GROUP 16
INPUT actin.dimer.dat
HELIX 0.0 0.0 -5.8 26.0
INPUT actin.dimer.dat
HELIX 0.0 0.0 -11.6 52.0
INPUT actin.dimer.dat
HELIX 0.0 0.0 -17.4 78.0
INPUT actin.dimer.dat
HELIX 0.0 0.0 -23.2 104.0
INPUT actin.dimer.dat
:

```

---

Each `actin.dimer.dat` file simply introduces another level in the hierarchy and maintains the same symmetry around the fibre axis but because the dimer centre lies on the axis, only a shift along Z is needed at the higher level.

---

```

                                actin.dimer.dat
GROUP 2
HELIX -3.55 -0.45 0.0 0.0
INPUT actin.one.dat
HELIX 3.55 0.45 -2.9 -167.0
INPUT actin.one.dat

```

---

The actin molecule consists of four domains arranged as shown in Figure 4(b), forming a flat disc which was naturally encoded as an oblate ellipsoid. This was (hand) specified by an axis of length 10 (-5 to 5) which relative to the size of the domain specified in `actin.model` gives an excessive axial ratio which is set to the maximum allowed ratio of 5.

---

<sup>2</sup>NB: the internal coordinates in the data structure are not  $\text{\AA}$ ngstroms



---

```

                                actin.one.dat
GROUP 4      -5.0  0.0  0.0      5.0  0.0  0.0
INPUT actin.dom1.dat
INPUT actin.dom2.dat
INPUT actin.dom3.dat
INPUT actin.dom4.dat
RELINK 32 127
RELINK 172 33
RELINK 90 173
RELINK 216 281
RELINK 372 217
RELINK 280 91
ENDS      372 126

```

---

As with the myosin head domains, the chain path through the actin domains must be relinked. (Figure 4(b)). This also entails the creation of a new terminal position which is specified by the ENDS command. (The equivalent myosin command falls in the tail segment).

The domain files (`actin.dom[1-4].dat`) are similar to the equivalent myosin head group domain files and introduce no novel features.

## 3.2 Driver construction

The `driver` routine encodes the dynamic aspect of the myosin motor. This includes specifying the actin/myosin bound state along with motion at the hinge points between the myosin head and tail and the myosin dimer.

### 3.2.1 Myosin-V processive "walking"

The structure of the actin/myosin bound complex is known from electron microscopy studies [Holmes *et al.*, 2003, Holmes *et al.*, 2004] which have been refined by energy minimisation [Holmes *et al.*, 1990, Chen *et al.*, 2002, Kawakubo *et al.*, 2005, Liu *et al.*, 2006b, Lorenz & Holmes, 2010]. From any given starting configuration, the actin lying closest to the myosin head was identified and the myosin

molecule moved towards it. When the myosin came within a predefined distance, its orientation was also refined. Together these operations "docked" the myosin molecule into a configuration relative to the actin that corresponded to the known structure. When in this position (referred to as "tightly bound"), the orientation of the myosin tail relative to the head was rotated about an axis that corresponded to what can be inferred from the structures of myosin with tails in different positions. This motion, known as the "power-stroke" can only occur if the other half of the myosin dimer is unbound [Reedy, 2000, Dominguez *et al.*, 1998]. In this situation, the free myosin will be carried along towards a new region of the actin filament where it will then search for a new binding site. Cycling through these states leads to a processive motion of myosin along the actin filament<sup>3</sup>.

The mechanics of the myosin walking motion can be decomposed into three distinct components: the myosin can be bound or unbound to actin, the head can be either in a pre- or post- power-stroke position and the two halves of the myosin dimer can swivel around their dimer interface. There is no explicit communication between the binding states of the two myosin molecules, except what can be communicated through their dimer interface. This has the form of an alpha-helical coiled-coil but little is known of how it responds under stress (tension) or how it affects the diffusion search of an unbound head. By contrast, the power-stroke (PS) transition is well characterised by many structural studies and the forward stroke (from pre- to post-PS positions) occurs only when the myosin is bound to actin whereas the reverse stroke (cocking the trigger) occurs in the unbound state. These transitions are all driven by an ATP hydrolysis cycle (summarised in Figure 5), the details of which will not be discussed here.

The deterministic progression of the myosin walking cycle lends itself well to being described as a finite state machine in which a transition to one state opens

---

<sup>3</sup>Note that this processive, or proper walking, motion of myosin-V differs from that of muscle myosin (II) in which the actin-myosin contact is only transient. The latter is more akin to a bank of rowers in which each myosin oar is dipped into the actin river.

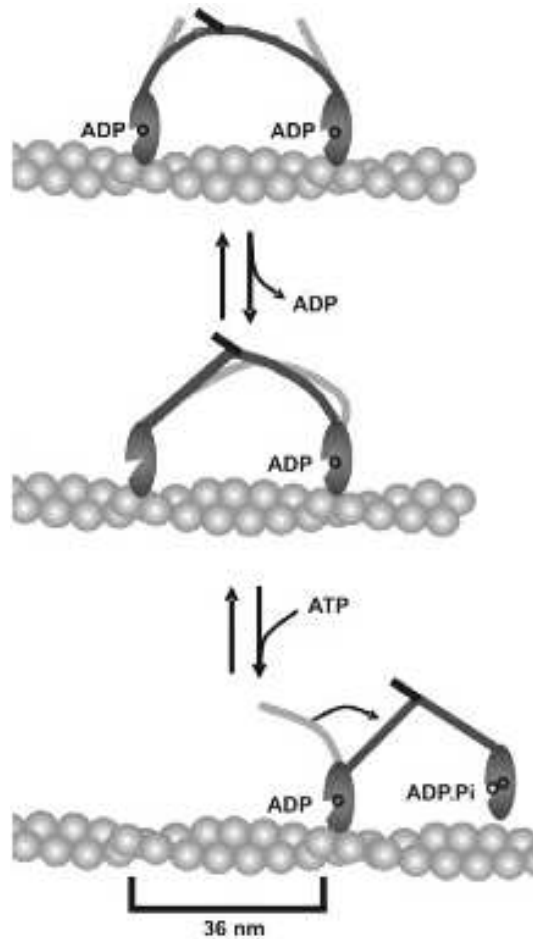


Figure 5: *a*. Both myosins bind ADP and are attached to actin (creating some bending strain from the relaxed ‘leg’ positions shown in light grey). *b*. With loss of ADP in the left leg, the power stroke can progress in the right leg. *c* On ATP binding, the left leg is released from actin and swings to the right and with ATP hydrolysis, it returns to the pre-power-stroke conformation. (Reproduced from [Reedy, 2000], with permission).

the options to progress to others. From a theoretical viewpoint, it is of interest to investigate how simple this machine can be to result in processive motion.

### 3.2.2 The myosin dimer hinge

The most independent component of the myosin machine is the dimeric interface which, by analogy to walking motion, will be referred to as the 'hip' joint. As little is known about its structure or dynamics, it was modelled simply as a constraint to hold the ends of the legs at a fixed distance.

Walking motion requires movement of the legs about the hip-joint and rather than rely on the generic diffusion built into the geometry engine, an additional motion was included in the **driver** routine to give any unbound myosin a rotational displacement about the hip. This provides an example of how the **driver** routine can utilise structural information across levels in the hierarchy as the rotation is applied to the whole myosin molecule (level-2) whereas the axis is determined by the domain positions at the end of the tail in both molecules (level-4 in separate sub-trees).

While a faster rate of driven motion reduces the search time for the free myosin head to find a new binding site without disrupting the position of the bound head, it also gives less time for the generic collision detection algorithm to avert clashes between the myosin molecules and the actin filament. To rectify this, a specific high-level check was made on the moving myosin based on the distance of the head from the actin filament (the actin dimer centre) and the other head group. The tail was also checked but as this is an elongated substructure, the closest approach of the ellipsoid major axes was monitored and if these fell below a fixed cutoff (set at the same distance as the hip joint) then the two myosins were separated.

### 3.2.3 The actin/myosin binding

The distance of the myosin head from each actin dimer centre was monitored and when this fell within a given range, the myosin was moved closer to the selected actin. This choice is made afresh every time the **driver** routine is activated, so the target actin can vary as the configuration of molecules changes. This approach mode, referred to as "loose binding", which includes no orientation component, continues until a shorter threshold is reached at which point the closest actin molecule is selected and the myosin is orientated and translated towards its known binding position. The orientation component is calculated based on the internal reference frames of the actin and the myosin head by applying the rotation matrix and translation vector that reproduces the docked complex. A fully-bound state is declared if this transformation can be applied and the end of the tail is placed within the range of the hip-joint to the other myosin.

### 3.2.4 The power-stroke

The power-stroke (PS) consists of a large swinging motion of the tail relative to the head which is easily encoded in the **driver** routine as a rotation about an axis that lies within the head. Structural studies have associated the swivel point with a particular  $\alpha$ -helix and the coordinates of this helix centre were taken as the hub. The rotation axis is less well defined but can also be inferred from the known structures and this was calculated and set as a fixed axis in the **driver** routine. Similarly, the extent of the left and right swing were preset with reference to the internal reference frame of the myosin head.

Given these constraints, a simple mechanism was encoded that maintained a position at the end of the swing range depending on the bound state of the myosin. If the myosin was unbound the tail angle was incremented until it attained the pre-PS position and only when the myosin became fully bound, was this motion reversed towards the post-PS position. These two states were used to impose

an additional condition on binding: that the myosin can only select an actin for binding when it is in the pre-PS position. This means that the post-PS state must detach from the actin and revert to the pre-PS position before it can rebind. This introduces a refractory period that depends on how fast the tail can swing which has a physical correspondence to the release of ADP and the subsequent binding of ATP.

### 3.3 Simulation

The specification of the **driver** process introduced a number of parameters that will not be considered in detail here. These were tested and roughly optimised to generate processive motion along the actin filament at as high a speed as could be achieved. To avoid end effects, the actin filament was bent into a circular 'race' track and the performance of the system was monitored by the lap-time for the myosin, combined with a penalty for myosin runners that fell off the track (i.e., both heads simultaneously detached).

Although the time scale is arbitrary, this simulation generated behaviour that was remarkably similar to that observed for real myosin-V motion by single particle studies (such as realtime AFM [Kodera *et al.*, 2010]) in which the molecules alternatively swing their legs to new binding positions along the actin filament. If viewed "upside-down", the motion is remarkably similar to the brachiating motion of the great apes (especially the gibbon) as they swing through the trees. The major divergence from reality in our simulations was the nature of the rate-limiting step, which in the real system is the time for release of ADP whereas in our simulation, the search time for the next binding site was rate limiting.

Various tests were performed on how rigid the myosin tail (legs) had to be and how strictly the hip-joint separation had to be maintained. If the legs were too weak or the hip too wide, then a state could be attained in which both myosins were tightly bound in the post-PS position. If this occurred then there was no

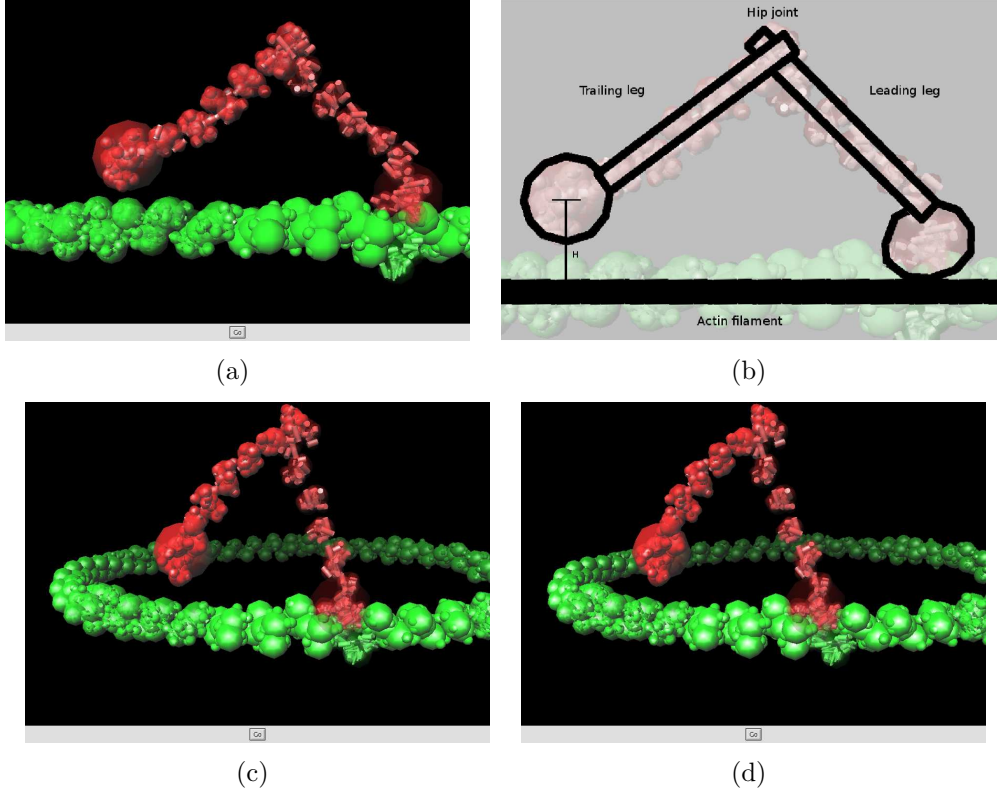


Figure 6: *a.* The myosin-V dimer model (red) is shown bound to the actin filament model (green). Secondary structure elements (SSEs) are depicted as cylinders (with large/small diameters for  $\alpha/\beta$ ) connected by fine lines. The translucent spheres show the higher level groupings of SSEs into domains. *b.* Annotates part *a*, with the myosin **leading leg** bound tightly to the actin in pre-power-stroke conformation. The myosin **trailing leg** has just detached from the **actin filament** (solid line) and is now free to pivot about the **hip-joint** between the two legs. The height (**H**) of the free myosin above the filament determines when it is recaptured, initially into a loose-binding mode. Parts *c* and *d* capture the myosin in a similar conformation showing the circuit of 104 actin molecules (52 dimers with 8 half-repeats) over which the myosin can move.

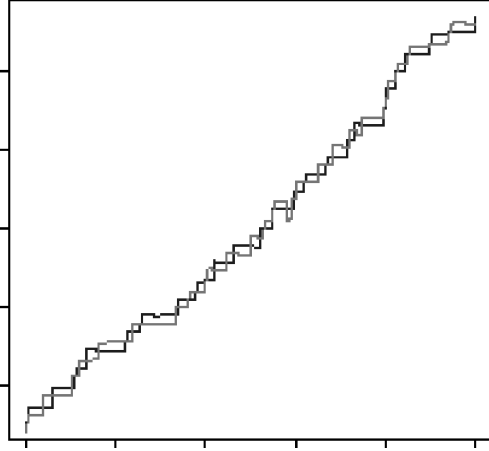
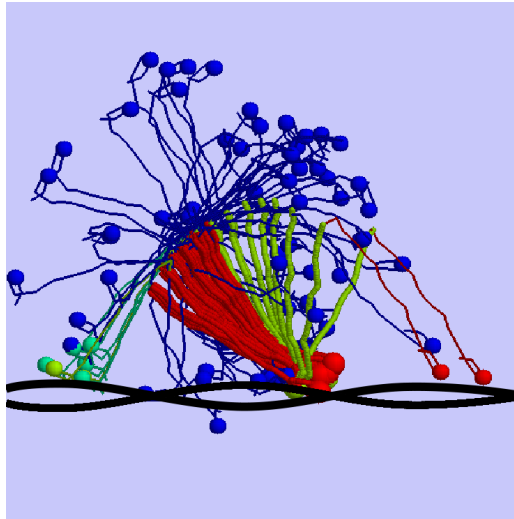


Figure 7: Step traces for myosin simulations showing a staircase of overlapping "steps" which is indicative of a brachiating (ape-like) processive motion. The X-axis is time and the Y-axis actin position in the filament. Trajectories for the two myosin heads are in black and grey. Occasional back-steps can be identified, similar to those seen in biophysical studies.



(a)

Figure 8: Superposed frames of myosin molecules taken from a simulation. Each myosin is represented in simplified form by a line connecting the mid-points of sequential domains in the tail segment which is drawn thicker for a tightly bound myosin. The head groups are represented by a sphere. The colours represent: active power-stroke (green), post-power-stroke conformation (red), unbound (blue) and loosely bound (cyan). The actin filament is shown schematically in black as a double helix tracing the two halves of each dimeric unit. The myosin moves from right to left.



frustration in the system and both myosins could remain stably bound forever, resulting in no motion. At the other extreme, if the legs were too stiff and the hip joint maintained strictly at a short separation, then the myosin pair did not have sufficient flexibility to attain simultaneous binding of the two heads, leading to one head always bound with the other always free or in a loosely bound state to various nearby actins — again resulting in no motion.

## 4 Conclusions

In this chapter we have described a general multi-scale modelling and simulation system that can be easily customised to any large system both in terms of setting-up the model geometry and in directing the behaviour of the simulation.

The use of a multi-level approach to the treatment of collisions means that, at higher levels, the system does not depend strongly on the choice of size or shape assigned to each level. Thus, crude shape approximations do not determine the direct behaviour of the system but just alert the lower level to check for collisions. The propagation of this cascade reaches down to the atomic level where eventually the "hard-sphere" repulsion is implemented giving a more realistic representation of the true excluded volume.

Unlike a conventional MD or MC simulation, the approach we have described cannot be expected to generate predictive behaviour from a summation of atomic interactions. It relies instead on a hierarchy of user or data defined constraints, such as the definition of secondary structures and their packing interactions. However, these constraints can be incomplete and the missing constraints will allow freedom for change in the system that will be stochastically explored through the underlying random motion that can be applied at every level.

The example we provided of the actin/myosin motor is typical of the scale of application that the system can accommodate, both in terms of number of atoms and size of displacement. While there are many aspects of the behaviour of the system that can be adjusted to reproduce realistic behaviour, we see its use mainly as a way of setting up the physical constraints for a complex system that can then be used to test if a hypothesis about the dynamics of the system is consistent with its physical structure.

## References

- [Astumian, 2001] Astumian, R. D. (2001). Making molecules into motors. *Sci. Am.* **285**, 56–64.
- [Aszódi & Taylor, 1994] Aszódi, A. & Taylor, W. R. (1994). Folding polypeptide  $\alpha$ -carbon backbones by distance geometry methods. *Biopolymers*, **34**, 489–506.
- [Bershitsky *et al.*, 2009] Bershitsky, S. Y., Ferenczi, M. A., Koubassova, N. A. & Tsaturyan, A. K. (2009). Insight into the actin-myosin motor from X-ray diffraction on muscle. *Front. Biosci.* **14**, 3188–3213.
- [Burghardt & Hu, 2007] Burghardt, T. P. & Hu, J. Y. and Ajtai, K. (2007). Myosin dynamics on the millisecond time scale. *Biophys. Chem.* **131**, 15–28.
- [Chen *et al.*, 2002] Chen, L. F., Winkler, H., Reedy, M. K., Reedy, M. C. & A., T. K. (2002). Molecular modeling of averaged rigor crossbridges from tomograms of insect flight muscle. *J. Struct. Biol.* **138**, 92–104.
- [Coureux *et al.*, 2004] Coureux, P. D., Sweeney, H. L. & Houdusse, A. (2004). Three myosin v structures delineate essential features of chemo-mechanical transduction. *EMBO J.* **23** (23), 4527–37.
- [De La Cruz *et al.*, 1999] De La Cruz, E. M., Wells, A. L., Rosenfeld, S. S., Ostap, E. M. & Sweeney, H. L. (1999). The kinetic mechanism of myosin v. *Proc Natl Acad Sci U S A*, **96** (24), 13726–31.
- [Dominguez *et al.*, 1998] Dominguez, R., Freyzon, Y., Trybus, K. M. & Cohen, C. (1998). Crystal structure of a vertebrate smooth muscle myosin motor domain and its complex with the essential light chain: visualization of the pre-power stroke state. *Cell*, **94**, 559–571.
- [Forkey *et al.*, 2003] Forkey, J. N., Quinlan, M. E., Shaw, M. A., Corrie, J. E. & Goldman, Y. E. (2003). Three-dimensional structural dynamics of myosin v by single-molecule fluorescence polarization. *Nature*, **422** (6930), 399–404.
- [Hofacker, 2003] Hofacker, I. L. (2003). Vienna RNA secondary structure server. *Nucleic Acids Res.* **31**, 3429–3431.
- [Holmes, 2009] Holmes, K. C. (2009). Actin in a twist. *Nature*, **457**, 389–390.
- [Holmes *et al.*, 2003] Holmes, K. C., Angert, I., Kull, J., Werner, J. & Schröder, R. R. (2003). Electron cryo-microscopy shows how strong binding of myosin to actin releases nucleotide. *Nature*, **425**, 423–427.

- [Holmes *et al.*, 1990] Holmes, K. C., Popp, D., Gebhard, W. & Kabsch, W. (1990). Atomic model of the actin filament. *Nature*, **347**, 44–49.
- [Holmes *et al.*, 2004] Holmes, K. C., Schröder, R. R., Sweeney, H. L. & Houdusse, A. (2004). The structure of the rigor complex and its implications for the power stroke. *Phil. Trans. R. Soc. B*, **359**, 1819–1828.
- [Izvekov & Voth, 2005] Izvekov, S. & Voth, G. A. (2005). A multiscale coarse-graining method for biomolecular systems. *J. Phys. Chem. B*, **109**, 2469–2473.
- [Katsimitsoulia & Taylor, 2010] Katsimitsoulia, Z. & Taylor, W. R. (2010). A hierarchic algorithm for simple Brownian dynamics. *Compu. Biol. Chem.*, **34**, 71–79. doi.10.1016/j.compbiolchem.2010.01.001.
- [Kawakubo *et al.*, 2005] Kawakubo, T., Okada, O., & Minami, T. (2005). Molecular dynamics simulations of evolved collective motions of atoms in the myosin motor domain upon perturbation of the ATPase pocket. *Biophys. Chem.* **115**, 77–85.
- [Kawakubo *et al.*, 2009] Kawakubo, T., Okada, O. & Minami, T. (2009). Dynamic conformational changes due to the atp hydrolysis in the motor domain of myosin: 10-ns molecular dynamics simulations. *Biophys. Chem.* **141**, 75–86.
- [Kodera *et al.*, 2010] Kodera, N., Yamamoto, D., Ishikawa, R. & Ando, T. (2010). Video imaging of walking myosin v by high-speed atomic force microscopy. *Nature*, **468**, 72–76.
- [Krebs & Gerstein, 2000] Krebs, W. G. & Gerstein, M. (2000). The morph server: a standardized system for analysing and visualizing macromolecular motions in a database framework. *Nuc. acids Res.* **28**, 1665–1675.
- [Liu *et al.*, 2006a] Liu, J., Taylor, D. W., Kremetsova, E. B., Trybus, K. M. & Taylor, K. A. (2006a). Three-dimensional structure of the myosin v inhibited state by cryoelectron tomography. *Nature*, **442**, 208–211.
- [Liu *et al.*, 2006b] Liu, Y., Scolari, M., Im, W. & Woo, H.-J. (2006b). Protein-protein interactions in actin-myosin binding and structural effects of R405Q mutation: a molecular dynamics study. *Proteins: Struc. Fun. Bioinfo.* **64**, 156–166.
- [Lorenz & Holmes, 2010] Lorenz, M. & Holmes, K. C. (2010). The actin-myosin interface. *Proc. Natl. Acad. Sci. U S A*, **107**, 12529–12534.
- [Mehta *et al.*, 1999] Mehta, A. D., Rock, R. S., Rief, M., Spudich, J. A., Mooseker, M. S. & Cheney, R. E. (1999). Myosin-v is a processive actin-based motor. *Nature*, **400** (6744), 590–3.

- [Parker *et al.*, 2009] Parker, D., Bryant, Z. & Delp, S. L. (2009). Coarse-grained structural modelling of molecular motors using multibody dynamics. *Cell. Mol. Bioeng.* **2**, 366–374.
- [Reedy, 2000] Reedy, M. C. (2000). Visualizing myosin’s power stroke in muscle contraction. *J. Cell Sci.* **113**, 3551–3562.
- [Schlick, 2002] Schlick, T. (2002). *Molecular modeling and simulation : an interdisciplinary guide*. Springer, New York and London.
- [Spudich, 1994] Spudich, J. A. (1994). How molecular motors work. *Nature*, **372** (6506), 515–8.
- [Taylor, 1999] Taylor, W. R. (1999). Protein structure domain identification. *Prot. Engng.* **12**, 203–216.
- [Taylor, 2001] Taylor, W. R. (2001). Defining linear segments in protein structure. *J. Molec. Biol.* **310**, 1135–1150.
- [Taylor & Aszódi, 2005] Taylor, W. R. & Aszódi, A. (2005). *Protein Geometry, Classification, Topology and Symmetry*. Institute of Physics. Currently published by CRC Press.
- [Taylor & Katsimitsoulia, 2010a] Taylor, W. R. & Katsimitsoulia, Z. (2010a). A soft collision detection algorithm for simple Brownian dynamics. *Compu. Biol. Chem.*, **34**, 1–10. doi.10.1016/j.compbiolchem.2009.11.003.
- [Taylor & Katsimitsoulia, 2010b] Taylor, W. R. & Katsimitsoulia, Z. (2010b). A coarse-grained model for actin myosin simulation. *J. Molec. Graph. Model.* **29**, 266–279.
- [Taylor *et al.*, 2011] Taylor, W. R., Katsimitsoulia, Z. & Poliakov, A. (2011). Simulation of cell movement and interaction. *J. Bioinfo. Compu. Biol.* **9**, 91–110.
- [Taylor *et al.*, 1983] Taylor, W. R., Thornton, J. M. & Turnell, W. G. (1983). A ellipsoidal approximation of protein shape. *J. Molec. Graphics*, **1**, 30–38.
- [Veigel *et al.*, 2005] Veigel, C., Schmitz, S., Wang, F. & Sellers, J. R. (2005). Load-dependent kinetics of myosin-V can explain its high processivity. *Nat. Cell Biol.* **7**, 861–869.
- [Zhang *et al.*, 2009] Zhang, Z., Pfendtner, J., Grafmuller, A. & Voth, G. A. (2009). Defining coarse-grained representations of large biomolecules and biomolecular complexes from elastic network models. *Biophys. J.* **97**, 2327–2337.

[Zheng, 2011] Zheng, W. (2011). Coarse-grained modelling of conformational transitions underlying the processive stepping of myosin V dimer along filamentous actin. *Proteins: Struc. Fun. Bioinfo.* **online**, DOI:10.1002/prot.23055.

**Acknowledgements:** ZK was supported by the Oxford IRC in bionanotechnology and WRT by the Medical Research Council (UK). Claudia Veigel and Justin Molloy are thanked for valuable discussion and comments.