Simulating the Kinesin Walk: Towards a Definitive Theory

by

Richard John Wilson

A dissertation submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Molecular Organisation and Assembly in Cells Doctoral Training Centre, University of Warwick

October 2011

Table of Contents

List of Illustrations ..... **viii**

List of Tables ..... **x**

Acknowledgements ..... **xi**

Declaration ..... **xiii**

Abstract ..... **xiv**

Abbreviations ..... **xv**

Chapter 1 Starting a Fantastic Voyage **1**

1.1 Introduction 1

1.1.1 Nanomachines 2

1.1.2 Dementia 2

1.1.3 The normal brain 4

1.1.4 Axonal transport and dementia 6

1.1.5 This study 7

1.2 Axonal transport system 9

1.2.1 Microtubule track 9

1.2.2 Kinesin motor 11

1.2.2.1 Nucleotide binding site 12

1.2.2.2 Microtubule binding site 12

1.3 Kinesin procession 14

1.3.1 Speed 15

1.3.2 Stall force 16

1.3.3 Run length and detachment 16

1.3.4 Can kinesin walk backwards? 17

1.3.5 Kinesin puppet 17

1.3.6 Impaired transport studies 18

1.4 Kinesin procession mechanism 18

1.4.1 Inching or toeing?.....19
1.4.2 Walking the line.....20
1.4.3 Powering the motor 21
1.4.3.1 Myosin.....21
1.4.3.2 Kinesin 22
1.4.3.3 Zippering 24
1.4.3.4 Power stroke 25
1.4.3.5 Rectified Brownian motion.....27
1.4.4 Wait state configuration.....27
1.4.4.1 Both heads bound 28
1.4.4.2 Free head 28
1.4.4.3 Parked head 28
1.4.5 Head coordination.....29
1.4.5.1 Gated rear head.....29
1.4.5.2 Gated front head.....30
1.4.5.3 ADP release gate.....30
Chapter 2 Modelling kinesin..... 32
2.1 Why model kinesin?.....32
2.2 Previous models of kinesin.....33
2.2.1 Brownian ratchet 33
2.2.2 Chemical-kinetic.....34
2.2.3 Molecular dynamics.....35
2.3 The approach of this study 36
2.3.1 Executable biology and agent-based modelling.....36
2.3.2 Modelling the stepping mechanism.....37
2.3.2.1 Brownian ratchet 37
2.3.2.2 Rectified Brownian motion.....38
2.3.2.3 Power stroke 39

2.4 The simulation 39
2.4.1 Head simulation 39
2.4.2 State transition (event) timings 41
2.4.3 Wait state 42
2.4.3.1 Linker strain and head binding 42
2.4.4 Stepping 43
2.4.5 Load 44
2.4.6 Gating 44
2.4.7 Obstacle 45
2.4.8 Multimotor simulation 45
2.4.9 Visual interface 46
Chapter 3 Results – fixed ATP arrival 47
3.1 Published results 47
3.2 Four part study 48
3.2.1 Part one – PS vs RBM 48
3.2.1.1 Discussion – system phases and timings 51
3.2.1.2 Discussion – procession and detachment 52
3.2.1.3 Discussion – explanation 53
3.2.1.4 Gated PS compared to RBM 54
3.2.2 Part two - gating hypothesis 56
3.2.2.1 Does RBM require a gating mechanism? 56
3.2.2.2 Discussion 57
3.2.3 Part three - behaviour under load 59
3.2.3.1 Discussion 62
3.2.4 Part four - blockage behaviour 64
3.2.4.1 Discussion 65
Chapter 4 Results – random ATP arrival 66
4.1 Single motor investigation 66

4.1.1 Velocity and dwell times at no load 67
4.1.1.1 Simulation results 67
4.1.1.2 Experimental results 69
4.1.1.3 Discussion 70
4.1.2 Load 70
4.1.2.1 High [ATP] 71
4.1.2.2 Medium [ATP] 75
4.1.2.3 Low [ATP] 78
4.1.2.4 Discussion 81
4.2 Two motor investigation 83
4.2.1 Multimotor simulation 83
4.2.2 Simulation results 83
4.2.3 Experimental results 85
4.2.4 Discussion 87
Chapter 5 Evaluation 88
5.1 The study 88
5.2 The simulation 89
5.2.1 Simulation advantages 90
5.2.2 Simulation limitations 90
5.3 The model 90
5.4 Gated RBM model challenged 91
5.5 RBM stepping challenged 92
5.5.1 Power stroke revisited 92
5.5.2 Wait state configuration 93
5.5.2.1 Both heads bound? 93
5.5.2.2 Parked head? 94
5.5.3 ATP-binding gate necessary for procession? 95
5.5.4 Stalling mutants 95

5.6 Model predictions 96
5.6.1 Non-hydrolysable analogue 96
5.6.2 Backsteps at low load 97
5.6.3 Wandering mutants and unfuelled procession 98
5.6.3.1 Extended linkers 98
5.6.3.2 Zero ATP walking 99
5.6.3.3 Walking without zippering 99
5.7 Experiments suggested by the model 100
5.7.1 Gating 100
5.7.2 Non-hydrolysable analogue 100
Chapter 6 Summary and conclusions 101
6.1 Summary 101
6.2 The study 101
6.3 Simulation results 102
6.3.1 Fixed ATP arrival results 102
6.3.2 Random ATP arrival results 102
6.4 The model 103
6.4.1 Explanatory power 103
6.4.2 Challenges 104
6.5 Conclusions 104
References 106
Appendix A Load data 113
Appendix B Program listing 118
B.1 Main listing – program for chapter 3 results 118
B.2 Main listing – single motor program for chapter 4 results 125
B.3 Main listing – multimotor program 133
B.4 Update routines 142
B.5 Brownian motion routines 146

B.6 Analysis routines 148

B.7 Display routines 150

Appendix C Published papers 154

List of Illustrations

Figure 1.1 Brain section comparison © 2007 Alzheimer's Association. All rights reserved. Illustration by Stacy Janis. 3

Figure 1.2 Diagram of a neuron (Mariana Ruiz Villarreal, Wikimedia Commons, public domain image). 5

Figure 1.3 Diagram of kinesin towing a vesicle along a microtubule. From Viel, Lue and Liebler (2006).[13] 8

Figure 1.4 Diagram of a neuron showing microtubule organisation and organelles. From Conde and Cáceres (2009).[20] 11

Figure 1.5 Diagram of kinesin. From Vale (2003).[25].....12

Figure 1.6 PDB 2wbe ribbon diagram of a kinesin head above a tubulin dimer.[29] 14

Figure 1.7 Myosin-II mechanochemical cycle as 4 snapshots. From the top: ATP has been hydrolysed and the head is diffusing; next, the head has bound the filament and phosphate has been released; next, after the head has pulled the filament to the right; lastly, ATP has bound and the head has detached. From Vale and Milligan (2000)[54]. 23

Figure 1.8 Diagram of kinesin processionary cycle. The letter above each head indicates which nucleotide is bound: D for ADP, T for ATP, DP for hydrolysed ATP, 0 for none.....24

Figure 1.9 Myosin V motor domain with both heads bound to actin (left) compared to kinesin motor domain bound to a microtubule (right) at the same scale. (from Vale and Milligan 2000)[54]. 25

Figure 2.1 Brownian ratchet mechanism. Top: particles firstly diffuse at random in a flat potential; middle: the potential changes to a sawtooth when more particles diffuse left than right; bottom: the result is that particles accumulate in potential wells. 34

Figure 2.2 Chemical-kinetic schema. The motor steps to the right by distance d from MT binding site, l, to the next. The N chemical states leading to a step are depicted as circles joined by an arc. At any state, j, the motor moves to the next state or the previous state or detaches as determined by rate constants u_j, w_j and δ_j respectively. Kolomeisky & Fisher (2007)[84] figure 5.....35

Figure 2.3 Head state transition table.....41

Figure 2.4 Program display screen snapshot. The MT is depicted in brown, the motor heads in red and green, and the track of the motor in dark grey..46

Figure 3.1 Relationship between ungated stepping mechanisms and procession under varying linker strain.....50

Figure 3.2 Relationship between stepping mechanisms and detachments under varying linker strain. 51

Figure 3.3 Relationship between stepping mechanisms and procession under varying linker strain. 54

Figure 3.4 Comparison of gated and ungated RBM procession. 59

Figure 3.5 Load characteristics of RBM stepping without gating. 62

Figure 3.6 Load characteristics of RBM stepping with gating. 62

Figure 3.7 Nishiyama et al. (2002)[105] figure 4a plotting the fraction of forward steps (circles), backward steps (triangles) and detachments (squares) for [ATP] of 1 mM (upper plot) and 10 μM (lower plot). The dashed lines sum the backward steps and detachments. 64

Figure 4.1 Relationship between velocity and [ATP]. 68

Figure 4.2 Relationship between dwell time and [ATP]. 69

Figure 4.3 Yajima et al. (2002)[106] figure 2c: plot of run length (central, in red), dwell time (top left to bottom right in blue) and velocity (bottom left to top right in green). 70

Figure 4.4 High [ATP] ungated RBM step ratio to load relationship. 73

Figure 4.5 High [ATP] gated RBM step ratio to load relationship. 74

Figure 4.6 High [ATP] gated PS step ratio to load relationship. 74

Figure 4.7 Mid [ATP] ungated RBM step ratio to load relationship. 77

Figure 4.8 Mid [ATP] gated RBM step ratio to load relationship. 77

Figure 4.9 Mid [ATP] gated PS step ratio to load relationship. 78

Figure 4.10 Low [ATP] ungated RBM step ratio to load relationship. 80

Figure 4.11 Low [ATP] gated RBM step ratio to load relationship. 80

Figure 4.12 Low [ATP] gated PS step ratio to load relationship. 81

Figure 4.13 Simulated multiple motor comparison of velocity. 84

Figure 4.14 Simulated multiple motor comparison of run length. 85

Figure 4.15 Multiple motor comparison of velocity (Beeg *et al.* 2008[107] figure 2b). 86

Figure 4.16 Multiple motor comparison of run length (Seitz and Surrey 2006[39] figure 1d). 86

List of Tables

Table 3.1 Timing combinations yielding interrupted procession – PS and RBM. 49

Table 3.2 Detachments for each stepping mechanism. 50

Table 3.3 Timing combinations yielding interrupted procession. 55

Table 3.4 Gating comparison: procession data. 58

Table 3.5 Gating comparison: average stepping ratios. 60

Table 3.6 Gating comparison: minimum and maximum stepping ratios. 61

Table 4.1 Velocity and dwell time comparison under no load. 68

Table 4.2 High [ATP] ungated RBM step ratios. 72

Table 4.3 High [ATP] gated RBM step ratios. 72

Table 4.4 High [ATP] gated PS step ratios. 73

Table 4.5 Mid [ATP] ungated RBM step ratios. 75

Table 4.6 Mid [ATP] gated RBM step ratios. 76

Table 4.7 Mid [ATP] gated PS step ratios. 76

Table 4.8 Low [ATP] ungated RBM step ratios. 78

Table 4.9 Low [ATP] gated RBM step ratios. 79

Table 4.10 Low [ATP] gated PS step ratios. 79

Table 4.11 Stall force comparison. 82

Table 4.12 Multimotor results. 84

Table A.1 Forward steps 113

Table A.2 Backward steps 114

Table A.3 Detachments 115

Table A.4 Ungated step ratios 116

Table A.5 Gated step ratios 117

Acknowledgements

Before acknowledging the contribution of thoughtful and friendly individuals to my research at Warwick, I would make an environmental observation. Though the university has little in the way of architectural delights, there are natural areas to walk and contemplate or feed the wildfowl. Such activities have helped me to bounce back from the many and varied setbacks to which a student is prone. I wish to thank the powers that be for preserving the campus greenery, and the Woodland Trust for managing the arboreal fragment separating the two University conurbations.

Central to my academic life has been MOAC: one hell of a DTC. The MOAC community is characterised by mutual comradeship, multidisciplinary fun, and bursts of intense debate (thank you Martin and Hugo, the flying Dutchman). Then there are the learned picnic seminars, weird and wonderful training sessions, and the amazing annual jamboree, all of which have contributed to a positive research environment. The person I am fundamentally indebted to, MOAC's stalwart inspiration, head honcho, and prime mover, is the redoubtable Alison Rodger. Having provided the opportunity for me to embark on a momentous career change (I used to have a real job), she has followed through as a true professional, unstinting in her commitment. Her admin team - Dorothea Mengels and Monica Lucena – were beyond compare: the veritable equivalents of Bodie and Doyle. I would like to embarrass Lahari, Gemma and Joob by singling them out of all my MOAC colleagues as being there for me (as I trust I was for them).

I must thank my supervisor, Jianfeng Feng, for the support he has provided me, especially in times when I doubted the course of this project. Thanks are

due to my advisory committee: Sara Kalvala, Matthew Hodgkin, and Matthew Turner, who have helped to keep me on track.

My association with the Computational Biology group has proved beneficial, in particular I want to thank Xuejuan Zhang for theoretical discussions about Brownian motors and Sara Kalvala for reviewing draft papers. In the latter regard, I am also grateful to Brent Kiernan of Systems Biology. A particular vote of thanks is due to external peer reviewers for stimulating an improvement in the quality of my publications and my work in general.

I wish to acknowledge the contribution of Teresa Pinheiro for formulating and supervising a project to investigate α-synuclein neurotoxicity *in vitro*, and Narinder Sanghera for laboratory skills training. Though this early research in the Department of Biological Sciences has proved not to be within the scope of this document, my exposure to the realities of experimental research has lent perspective to my thesis.

The financial support of the Engineering and Physical Sciences Research Council has not only kept the author off the streets, but has also funded the laptop on which I'm writing this and on which I developed the simulation software. My thanks to Jacob Navia for supplying a high quality compiler suite (available at http://www.cs.virginia.edu/~lcc-win32/) which has enabled me to build and run the program.

Finally, a very special mention goes to my dear mother and sister who have plied me with tea, furnished newspaper articles on Alzheimer's, and helped me through Don Quixote moments.

Declaration

The work presented in this document is original (except where otherwise stated). No portion of the work submitted in this thesis has been submitted in support of an application for another degree or qualification at this or any other University or institute of learning.

Abstract

Dementia is a set of incurable, fatal diseases characterised by irreversible degeneration of the brain. One theory of its cause is the failure of intracellular transport in the axons of the neurons that compose the brain. Kinesin is a key motor transporting vital cargo along the axon. We know that this motor is a bipedal engine stepping forward along a polypeptide track but it is too small and fast for this motion to be observed using current experimental techniques. The stepping detail is therefore open to debate. This study firstly addresses the question of how kinesin steps and secondly pilots a possible method for investigating transport disruption *in silico*.

To investigate the detail of stepping, a program has been designed and built to simulate kinesin traversing its track along a section of axon. The motor is modelled as simple, interacting agents obeying rules abstracted from known chemical and binding properties of its components. The agent-based method has proven useful and efficient on the small scale and has potential for simulating the larger and more complex system of axonal transport. This would enable investigation of transport failure in the context of finding a cure for dementia.

A new model of kinesin stepping has been formulated as a consequence of performing virtual experiments using the simulation. Analysis of *in vivo* and *in vitro* experimental studies shows that the model accounts for a wide range of published results, explaining many findings. New experiments are suggested to test the model based on its falsifiable predictions. The principal conclusion of this study is that kinesin stepping is rectified Brownian motion.

Abbreviations

ADP	The nucleotide adenosine diphosphate, a product of ATP hydrolysis
AMP-PNP	Analogue of ATP, non hydrolysing
APP	Amyloid precursor protein, produces amyloidß by proteolytic cleavage
AT	Axonal transport
ATP	The nucleotide adenosine triphosphate, used extensively in organisms to carry energy
AVEC-DIC	Video-enhanced contrast differential interference contrast, an experimental technique for indirectly observing motors in motion
CNB	Cover neck bundle, a structure formed during kinesin stepping
FIONA	Fluorescence imaging one-nanometer accuracy, an experimental technique using averaging to obtain high spatial accuracy
FRET	Fluorescence resonance excitation, an experimental technique for measuring distances of a few nanometres
K0	Kinesin head with no nucleotide bound but bound to the microtubule
KD	Kinesin head bound to ADP and to the microtubule
KDP	Kinesin head bound to hydrolysed ATP and to the microtubule
KDu	Kinesin head bound to ADP but free of the microtubule

KT	Kinesin head bound to ATP and to the microtubule
MD	Molecular dynamics, a computer modelling system simulating molecules at the atomic level
MT	Microtubule, tubulin polypeptide tube along which kinesin moves
Ncd	Non-claret disjunctional kinesin, a type of kinesin that moves towards the minus-end of the microtubule
PDB	Protein Data Bank, a repository of bio-molecular structures (www.rscb.pdb)
PS	Power stroke, a powered stepping mechanism
RBM	Rectified Brownian motion, a diffusional stepping mechanism

Chapter 1 Starting a Fantastic Voyage

> "A biological system can be exceedingly small. Many of the cells are very tiny, but they are very active; they manufacture various substances; they walk around; they wiggle; and they do all kinds of marvelous things, all on a very small scale. Also, they store information. Consider the possibility that we too can make a thing very small which does what we want, that we can manufacture an object that maneuvers at that level!"
>
> **Richard P. Feynman 1959**

1.1 Introduction

The work described here is inspired by the dream of nanomachines for medical treatment and is motivated by the author's desire to find a cure for dementia. We know that dementia destroys the brain but the process is not well understood. One theory of the cause of this destruction is that transport inside brain cells breaks down. Kinesin is a nanomachine that transports vital cargo from the central site of synthesis to the cell's periphery to sustain cellular communication and hence maintains brain functionality. This study is an investigation into the mechanism of motion of kinesin. A better understanding of kinesin function advances our understanding of cargo transport and provides information useful for the design of motors for artificial nanomachines. Both these paths have potential for leading to a cure for dementia.

Material in this chapter has been peer-reviewed and published in the journal Science Progress under the title, *Towards a cure for dementia: the role of axonal transport in Alzheimer's disease* (see appendix C).

1.1.1 Nanomachines

Nanomachines are machines of nanometre dimensions. Eukaryote cells are complex chemical factories dependent on several types of nanomachine from pumps and polymerases to actuators and transporters. Many of these machines are motors powered by the free energy released from the hydrolysis of the nucleotide adenosine triphosphate (ATP). Most ATP is synthesised by rotary motors (ATPsynthases) that reside in the cell's mitochondria and are driven by a proton gradient generated during respiration. Mitochondria are micrometre-sized organelles transported about the cell by linear motors, the type of nanomachine at the heart of this thesis. Powered by ATP hydrolysis, linear motors traverse cytoskeletal protein polymers performing several functions: muscle contraction, cilia movement, chromosome separation during cell division and cargo transport.

The idea of fabricating artificial nanomachines was first put forward by Nobel laureate physicist Richard Feynman in a lecture titled *There's Plenty of Room at the Bottom: An Invitation to Enter a New Field of Physics* given at the California Institute of Technology in 1959. This concept was illustrated in the 1966 film *Fantastic Voyage* which depicts a tiny craft that ferries miniaturised people through a scientist's blood vessels to fix a clot in his brain. Of course miniaturising people is impossible but devising robots small enough to navigate the body and perform medical procedures either autonomously or at the direction of medical staff is one of the goals of nanotechnology. A recent advance in this field is the Proteus piezoelectric nanomotor.[1]

1.1.2 Dementia

Our population is increasing and more of us are suffering disease and infirmity in old age. One particularly distressing set of predominantly late-onset

diseases is dementia. Not only is dementia horrific for the individual, their family and friends, it is also very costly for society. An estimated 20 million people suffer dementia worldwide, a figure expected to double every 20 years. In the UK, the current cost of care alone is calculated at over £17B per annum while the yearly death toll is over 60,000.[2] Alzheimer's disease is the most prevalent type of dementia. It is an incurable, fatal illness characterised by years of progressive mental decline: victims survive an average of 8 years from first diagnosis. Figure 1.1 graphically displays the dire effects of Alzheimer's on the brain.

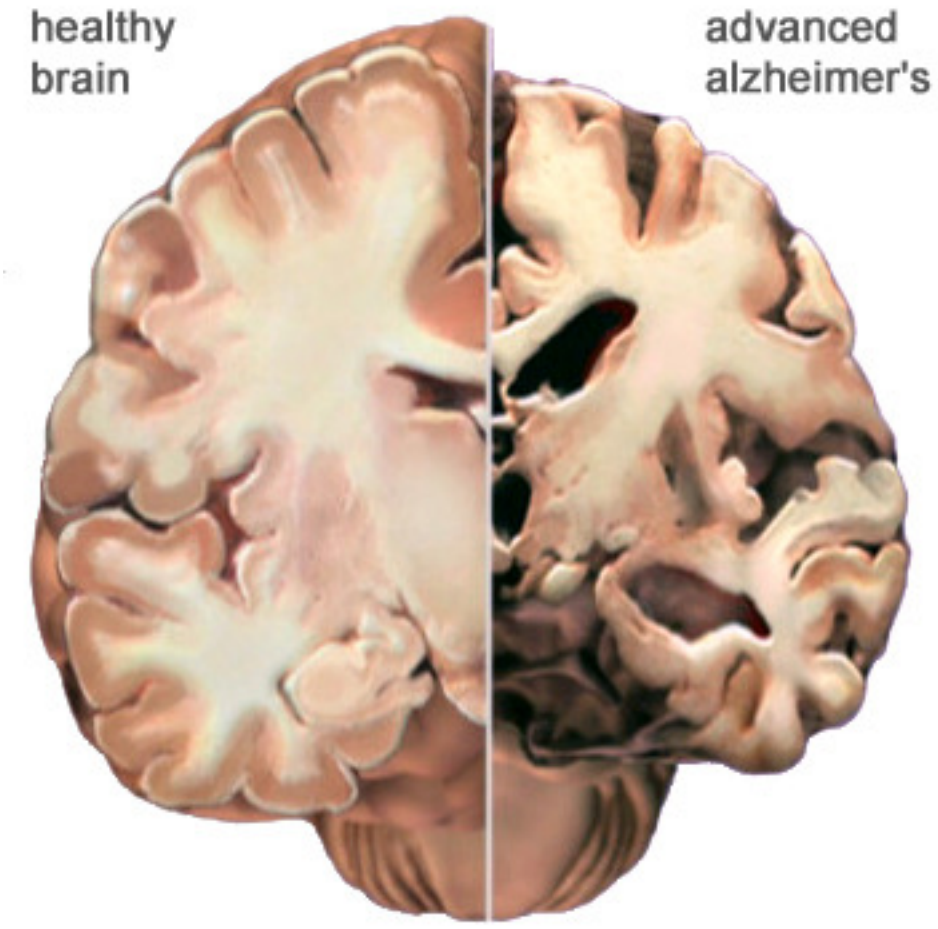

Figure 1.1 Brain section comparison © 2007 Alzheimer's Association. All rights reserved. Illustration by Stacy Janis.

The process of neurodegeneration and subsequent cell death is poorly understood. If we understood how the damage occurs then we would be in a position to develop a cure perhaps by engineering artificial nanomachines to seek out and repair malfunctioning molecular machinery in the brain.

One possible common factor that intensive international research has unearthed is disruption of a system of intracellular transport known as axonal or axoplasmic transport (AT). Defective axonal transport is implicated in several neurodegenerative conditions including Alzheimer's disease, motor neuron disease, amyotrophic lateral sclerosis, Huntington's disease and Parkinson's disease.[3; 4]

1.1.3 The normal brain

The human brain contains some 100 billion neurons (or neurones), cells specialised to process electrical signals, whose normal functioning is responsible for our mental faculties.[5] Figure 1.2 illustrates the main features of a neuron. In common with other cell types, the neuron has a cell body that contains the machinery that synthesises the wide range of components necessary to maintain cellular structure and function. The distinguishing morphological feature of a neuron is several branching projections (neurites) which serve to interconnect neurons into the networks that comprise the brain. The neurites comprise tapering, branching dendrites that receive signals from other neurons and a single axon that sends signals to other neurons. Axons are tubes of uniform diameter (ranging from 0.2 – 20 μm) which can span brain regions, extending centimetres in length (though peripheral neurons can extend over a metre: from the base of the spine to the toes). An axon may be encircled by cylinders of myelin sheath provided by oligodendrocytes (or, for peripheral neurons, Schwann cells as shown in the

figure) which serve to speed up signal conduction. The axon branches out at its terminal to synapse to the cell bodies and dendrites of post-synaptic neurons.

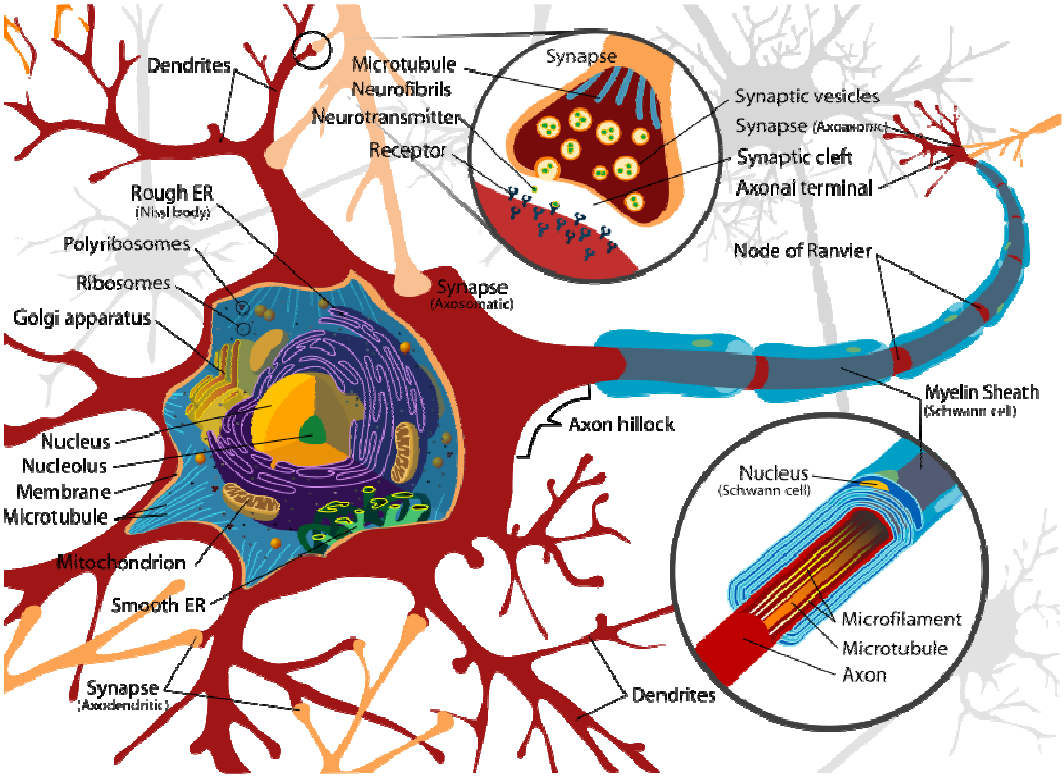

Figure 1.2 Diagram of a neuron (Mariana Ruiz Villarreal, Wikimedia Commons, public domain image).

The typical mode of signal transfer between neurons is chemical diffusion via the synapses. The signal that triggers the chemical release is in the form of an action potential (AP). The AP is generated when the input signal to the neuron, spatially and temporally combined across the plasma membrane, raises the potential at the axon hillock from a resting value of -70 mV to above threshold (about -55 mV). The AP travels down the axon membrane to the synapses as a wave of opening and closing of sodium then potassium voltage-gated ion channels which cause a 1 ms pulse of depolarisation peaking at about 40 mV. At the synapses it triggers calcium ion influx causing

vesicles storing signalling chemicals called neurotransmitters to fuse with the membrane and release their contents into the synaptic cleft. These molecules diffuse across the 10 – 20 nm gap to pass on the signal by activating membrane receptors in the post-synaptic neurons.

1.1.4 Axonal transport and dementia

Each time neurotransmitter is released, some is lost by diffusion from receptors after activation. Without replenishment, the stock of neurotransmitter at the synapse would diminish and, when exhausted, the neuron would cease to communicate. The proteins and membrane components necessary for neurotransmission must be manufactured in the cell body and then actively transported to the synapse.

There is evidence that AT is disrupted by the misfolded proteins characteristic of Alzheimer's disease and it is thought that such disruption could be a common factor in the neurodegeneration process leading to dementia.[6] The atrophied Alzheimer's brain displays myriad characteristic inclusions of two types: senile plaques and neurofibrillary tangles. Senile plaques are extracellular aggregates consisting mainly of amyloid-ß protein while neurofibrillary tangles are intracellular conglomerates composed largely of hyper-phosphorylated tau protein. As we cannot invasively experiment on human subjects, laboratory research is performed either on cells *in vitro* or on transgenic animals engineered to produce neurological symptoms similar to those of human disease. The findings of this research demonstrate a connection between tau, amyloid-ß and AT disruption.

Amyloid-ß results from the sequential cleavage of amyloid precursor protein (APP) by ß-secretase and γ-secretase; studies of murine neurons show that APP binds to the light chain of kinesin[7] and that kinesin transports vesicles

containing APP, ß-secretase and presenilin-1 (a component of γ-secretase).[8] Studies of mice expressing mutant human APP show a correlation between amyloid-ß production and AT disruption, both occurring before the formation of senile plaques.[9; 10] Mice expressing mutant tau protein develop loss of cognitive and motor function concomitant with impairment of AT and axonal swellings which are also seen in the brains of former early-stage Alzheimer's patients.[11] Oligomeric (as opposed to single molecule or fibrillar) forms of amyloid-ß cause cargo to detach from kinesin and thus disrupt AT.[12] Interaction between tau and amyloid was discovered in an *in vitro* study using mouse hippocampal cells which confirmed the inhibition of AT by amyloid-ß oligomers but also found that lowering tau levels prevented this effect.[13]

1.1.5 This study

Given the significance of AT malfunction to the neurodegenerative process, improving our understanding of AT and its modes of failure is an important task in the programme to conquer neurodegenerative disease. This study is largely concerned with understanding the normal functioning of kinesin. A start is made on the study of transport disruption, however, by modelling kinesin's behaviour at a blockage such as may result in axonal swelling.

This document reports on an in-depth theoretical study of the mechanism of motion of the linear motor kinesin-1. Kinesin is vital to AT: the motor carries cargo essential to sustaining neural communication (as described in section 1.1.3) from the body of the neuron to the synapses along cytoskeletal tracks in the axon known as microtubules. Figure 1.3 illustrates the motor carrying a vesicle along a microtubule.

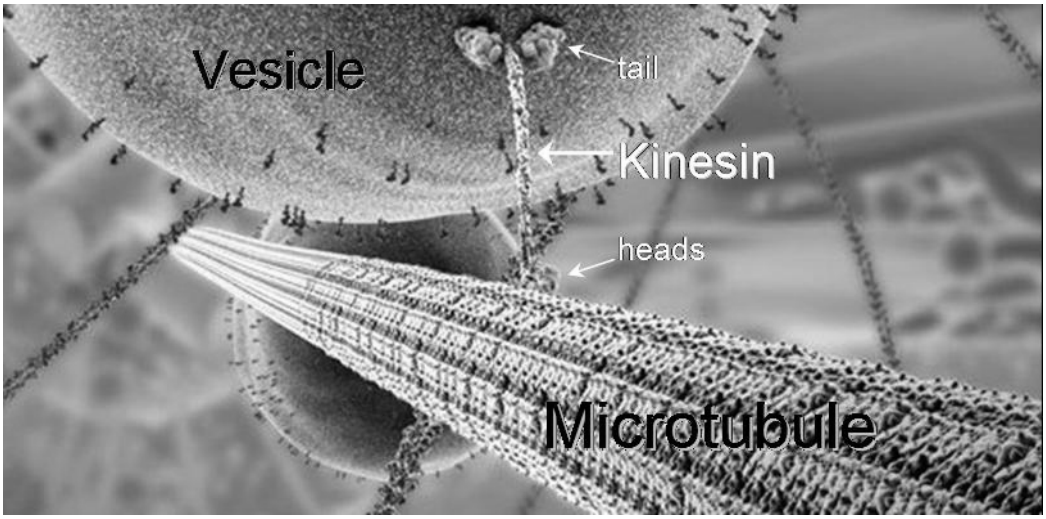

Figure 1.3 Diagram of kinesin towing a vesicle along a microtubule. From Viel, Lue and Liebler (2006).[14]

The mechanism of kinesin's motion is a controversial topic of current research. Many ingenious laboratory experiments have been performed revealing aspects of kinesin dynamics but the motor is too small and fast to be observed directly and so the mechanism remains a matter of debate. The author has devised a new hypothesis of kinesin stepping derived from existing theory and has designed and implemented software to test it. The software is an innovative use of agent-based modelling to simulate the motion of kinesin along a section of microtubule. It is hoped in the future to extend the model to encompass AT with the aim of investigating failure modes and so shed light on the cause of dementia.

The layout of this document is intended to conduct the reader through the research from motivation and background through methodology, results, discussion and analysis to the conclusions. The experimental background to the study is presented below. Chapter 2 motivates the choice of a fresh modelling approach and describes the software. Chapters 3 and 4 present virtual experiments and compare the results to existing experimental findings.

Chapter 5 reviews the research, presents a new model of kinesin stepping and discusses potential problems in relation to existing experimental findings, suggesting experimental tests to falsify the model. Chapter 6 summarises the document and concludes the study.

1.2 Axonal transport system

The molecular motors that transport axonal cargo along microtubules (cytoskeletal tracks) are classified into two families: kinesins and dyneins. Most kinesins perform anterograde transport, travelling towards the plus end of microtubules i.e. toward the synapses; dyneins perform retrograde transport, travelling in the opposite direction: towards the cell body.[15] The maximum velocity of anterograde transport is ~400 mm/d and of retrograde transport is ~250 mm/d. These velocities refer to the transport of membranous cargo; non-membranous cargo travels more slowly, at up to ~8 mm/d. It is thought that the transport mechanism is the same in both cases but that slow transport is fast transport interrupted by prolonged pauses.[16] Most AT is unidirectional but mitochondria, the cell's power plant organelles, move bidirectionally utilising both kinesin and dynein in a cooperative manner.[17] Mitochondrial movement is intermittent, displaying a range of velocities, as the organelle moves in response to energy requirements.[18] This study is focussed on the most well-studied motor, kinesin.

1.2.1 Microtubule track

A microtubule (MT) is a hollow 25 nm diameter protein polymer tube composed of laterally bound filaments. Filaments consist of tubulin heterodimers (~8 nm long) that spontaneously assemble head-to-tail. Each

heterodimer comprises a pair of similar tubulin proteins: α-tubulin and β-tubulin. An MT is a polar polymer capped by a ring of α-tubulins at the minus end and a ring of β-tubulins at the plus end.[19] MTs undergo dynamic instability whereby they cycle between growth and disassembly. This property is used by the cell to alter shape during development and mitosis but would threaten AT in mature cells which stabilise MTs with the microtubule-associated protein tau.[20]

MTs in dendrites and axons do not radiate from the centre, as in other cell types, but are bundled together. Figure 1.4 shows the microtubule structure of a dendrite (stabilised with MAP2) and the axon (stabilised with tau). Dendritic MTs have mixed orientation while axonal MTs are aligned in the same direction with the minus end towards the cell body.[15] Whereas the dendrites contain ribosomes, rough endoplasmic reticulum and Golgi outpost, the axon lacks this synthesis machinery.[21]

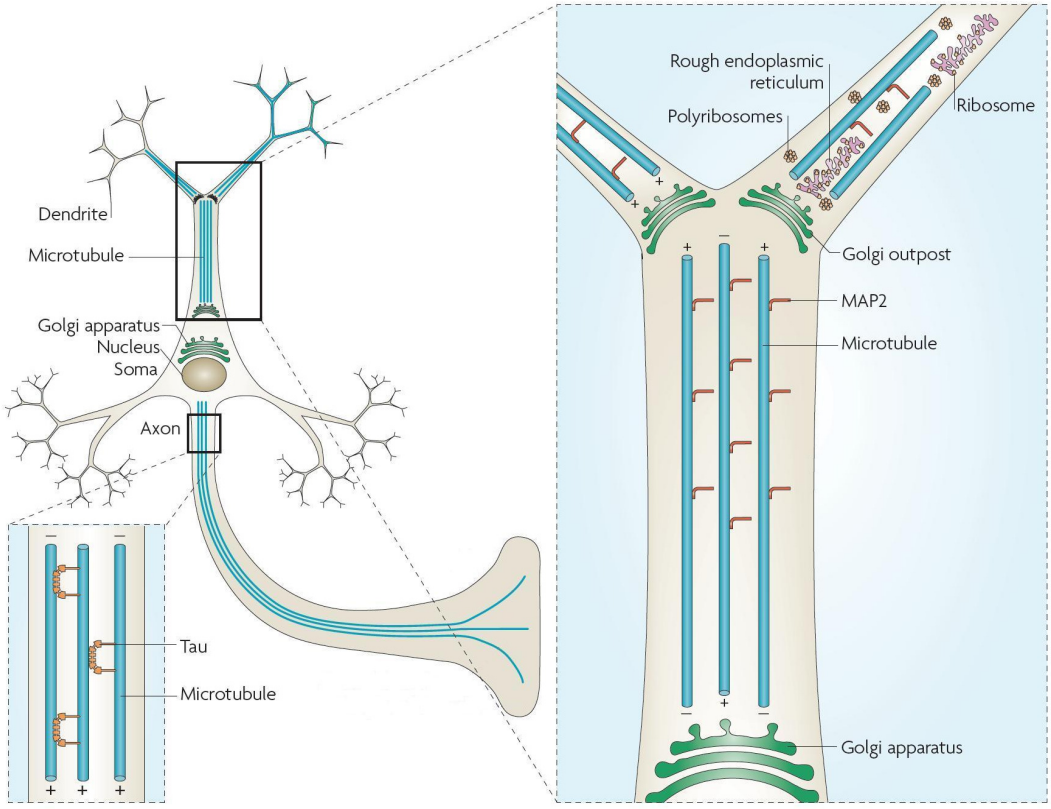

Figure 1.4 Diagram of a neuron showing microtubule organisation and organelles. From Conde and Cáceres (2009).[21]

1.2.2 Kinesin motor

Kinesin-1 (conventional kinesin) was initially identified in motility assays conducted on the axons of chick brain[22] and giant squid neurons.[23] All references to kinesin in this document are to kinesin-1 unless otherwise specified.

Kinesin is a homodimeric protein comprising identical heavy and light chains. Each heavy chain N-terminal region forms a globular, arrowhead-shaped motor domain (head) measuring approximately 4.5 nm by 4.5 nm by 7 nm which has a similar structure to the catalytic domain of myosin though does not share peptide sequence.[24] The head connects to the stalk by a short (~13 residue) single polypeptide neck linker. The stalk is a long (~60 nm), coiled-

coil polypeptide which binds to a light chain in the C-terminal region. The stalks intertwine to form the dimer with a fan-like tail.[25] Figure 1.5 illustrates the structure of the motor: the heads to the left of the picture are connected by blue linkers to the grey stalk which leads to the tail on the right depicted in green (light chains) and purple (C-terminals).

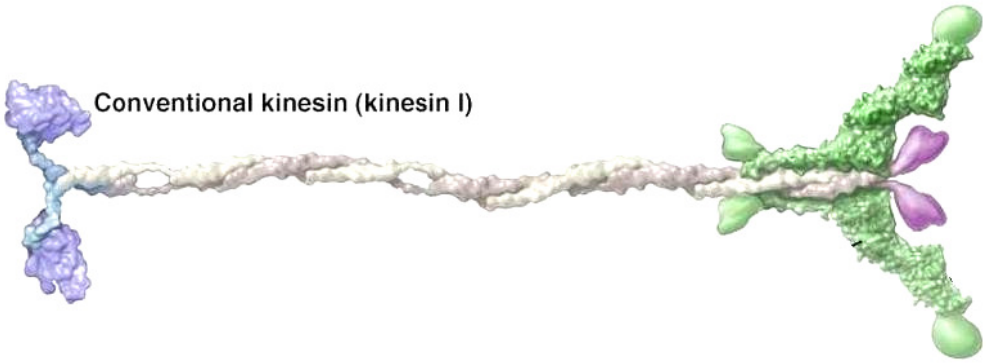

Figure 1.5 Diagram of kinesin. From Vale (2003).[26]

Kinesin's binding sites are at each end of the motor. The tail binds to cargo whereas each head has two interacting binding sites.

1.2.2.1 Nucleotide binding site

Each kinesin head has an active site that binds ADP in solution but releases this nucleotide on binding an MT and then favours ATP binding.[27] Kinesin is an ATPase: the active site hydrolyses ATP to ADP and phosphate, a process that liberates energy to power the motor.[28]

1.2.2.2 Microtubule binding site

The kinesin head binds mainly to ß-tubulin with the head's tapered end facing the plus-end of the MT and with alpha-helix α4 close to the cleft between the monomers composing the tubulin dimer.[29] Figure 1.6 is a ribbon diagram of PDB construct 2wbe: a head of kinesin-5 (similar to kinesin-1) docked to a

tubulin dimer (bottom); the plus-end of the dimer is to the right.[30] The cleft is approximately in the centre of the picture with α4 above it, depicted as a yellow coil facing into the page and angled downwards.

The strength and rigidity of binding are nucleotide dependent. The force required to unbind a head from the MT measured by optical tweezers is ~3 pN for an ADP-bound head but ~6 pN for a nucleotide-free or ATP-bound head. Higher forces (~4 pN and ~9 pN respectively) are required to detach heads when pulling kinesin backwards i.e. towards the MT minus-end, than when pulling forwards (Uemura et al. 2002).[31] Sosa et al. (2001) used a fluorescent probe to determine the rigidity of the binding of a head to a MT.[32] They found that a head is rigidly bound to the MT when nucleotide-free or bound to (analogues of) ATP or hydrolysed ATP, whereas an ADP-bound head is less tightly bound, showing a rocking motion.

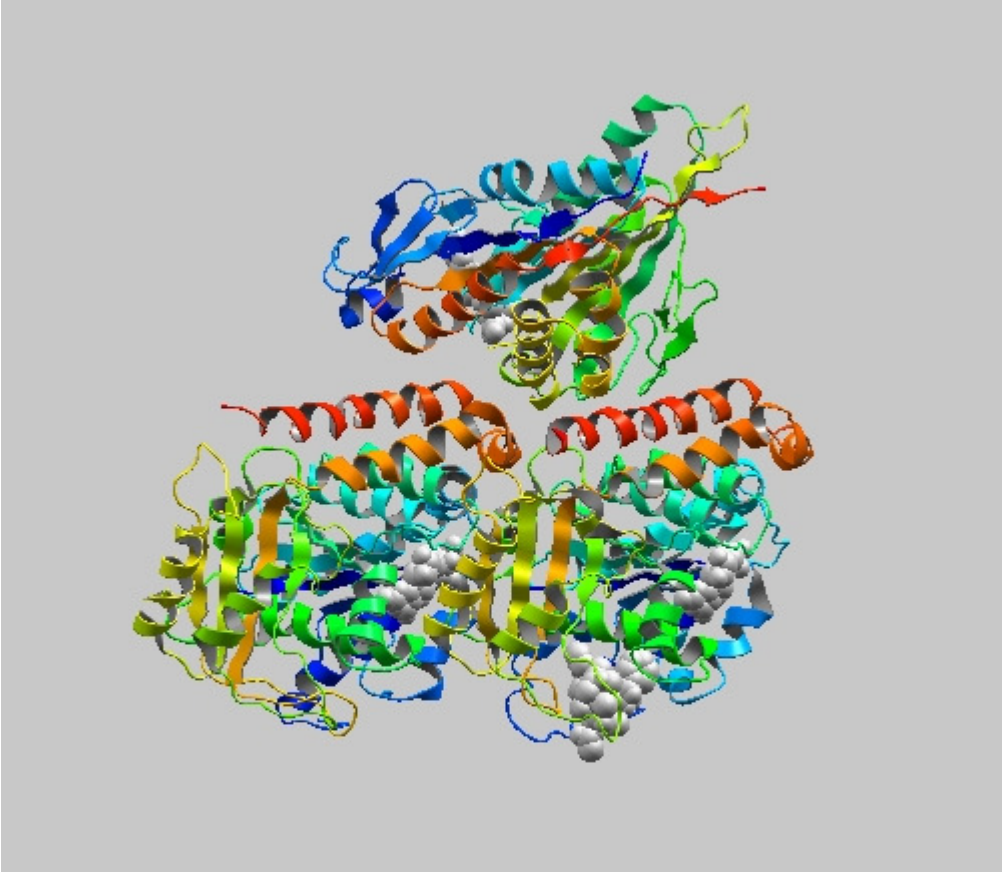

Figure 1.6 PDB 2wbe ribbon diagram of a kinesin head above a tubulin dimer.[30]

1.3 Kinesin procession

Kinesin is a processive motor: it takes successive 8 nm steps along the dimers of the MT towards its plus-end for hundreds of steps and at a speed of up to 1 μms^{-1} that varies with load and ATP concentration.[33]

Three kinds of single-molecule experimental techniques have been developed to investigate kinesin dynamics: the gliding assay, the bead assay and fluorescent tagging. In the gliding assay, an MT glides over a kinesin molecule immobilised heads-up to a cover slip (propelled by the heads). In the bead assay, the MT is secured to the slide while a plastic bead hundreds of nanometres in diameter is attached to the kinesin tail: the motor pulls the

bead along the MT in ATP solution. The third technique is similar to the bead assay but kinesin is tagged with a fluorophore rather than a bead.

Ideally, one would like to observe the heads in motion as kinesin transports cargo. This is not possible because current experimental techniques do not have the necessary resolution. Crystallography and electron microscopy have the spatial resolution but they are static probes whose samples have to be specially prepared and are viewed in artificial conditions. Light microscopy can probe moving samples in solution but has insufficient spatial resolution because of the diffraction limit of ~250 nm. Video-enhanced contrast differential interference contrast (AVEC-DIC) allows observations an order of magnitude smaller than the diffraction limit: transport of vesicles along the giant squid axon was first observed using AVEC-DIC.[34] Kinesin heads remain invisible at this resolution but a refinement of the fluorescent tagging technique has enabled head location to within about a nanometre. Fluorescence imaging one-nanometre accuracy (FIONA) relies on taking thousands of measurements of photons emitted from the fluorophore to build a distribution from which the mean is calculated. Yildiz et al. (2004) used FIONA to locate a tagged head to within 2 nm at a temporal resolution of 0.33 sec.[35]

1.3.1 Speed

The relationship between speed and hindering load has been measured using bead assays where laser tweezers apply force to the bead with a force-feedback system to stabilise this force at a constant value. A range of average speeds is reported in the literature but most studies record a mean of 600–700 nms^{-1} at low load (<1.5 pN) and saturation ATP concentration (1–2 mM).

Above 1.5 pN, the speed reduces almost linearly with load until the motor stalls while reducing the ATP concentration also slows the motor.[25]

1.3.2 Stall force

The load that prevents kinesin from moving forwards, the stall force, has been determined in optical trap experiments. These are bead assays where the bead is held stationary in a laser beam. The bead acts as a spring against which kinesin pulls, slowing the motor progressively until it reaches a stall plateau where it typically remains motionless for a second or more taking an occasional single step forward or back before detaching and returning to the start position. It then resumes its walk. Stall forces of between 5 pN and 7 pN have been measured with most studies showing no relation between stall force and ATP concentration.[36; 37; 38]

1.3.3 Run length and detachment

Measured with bead assays, kinesin's run length (the distance covered without detachment) averaged 1.5 microns with an approximately exponential distribution which implies a constant probability of detachment estimated at 1%.[39; 40; 41] Measured with gliding assays, however, run lengths were ~5 microns[42] or the length of the MT[43] i.e. no detachment was observed.

This discrepancy may be accounted for by the gliding assay being insensitive to detachment events. The MT stays close to the cover slip in a gliding assay and even if it rotates another binding site is close whereas bead rotation takes detached kinesin away from the MT.[39] Alternatively, perhaps optical tweezers used in bead assays tend to pull the motor away from its track.[25]

One study may indicate that the bead assay is more lifelike. The progression of kinesins labelled with quantum dot fluorophores along an MT was observed

to be punctuated by diffusion events *in vivo*.[44] This is not definitive evidence, however, as it is not known whether the detachments were spontaneous or caused by blockages or cargo snagging.

1.3.4 Can kinesin walk backwards?

Isolated backsteps have been noted in bead assays at a level of about two percent[38; 45; 46] but there is conflicting evidence as to whether kinesin can be induced to walk backwards i.e. take successive backsteps.

Coppin et al. (1997) imposed a sudden rearward force of up to 13 pN and observed that kinesin entered a stall plateau before detaching from the MT.[36] Carter and Cross (2005) repeated this experiment and found that kinesin tended to detach when subjected to a super-stall load of >10 pN, as expected.[37] In some cases, however, the motor stepped processively backwards (towards the minus-end of the MT and towards the bead) until the load reached stall force (~7 pN). They also found that reverse walking speed was related to ATP concentration which implies that kinesin was performing the normal hydrolysis cycle and so processing backwards.

1.3.5 Kinesin puppet

If the forward bias of the stepping mechanism is ATP-induced linker-zippering (see section 1.4.3.3) then a suitable external load should substitute for zippering, and so enable procession, in the absence of ATP. Yildiz et al. (2008) applied a constant load in the absence of nucleotide and found that kinesin does indeed process.[38] Forward loads of 3 pN and 6 pN induced kinesin to move processively at ~11 nms^{-1} and ~30 nms^{-1} respectively. Backward loads (applied in the direction of the minus-end of the MT) also caused the motor to process backwards though more slowly. Adding AMP-PNP raised the load

threshold while adding ADP lowered it. These results are consistent with the unbinding forces determined by Uemura et al. (2002) and give support to the zippering model.[31]

1.3.6 Impaired transport studies

Experiments designed to discover what kinesin does when faced with an obstacle have so far yielded inconsistent results.

Crevel et al. (2004) prevented kinesin from stepping by placing a barrier on the MT and found that the motor quickly detached.[47] They partially decorated an MT with wild-type dimeric rat kinesins then saturated the MT with a mutant monomeric kinesin that binds irreversibly to the MT and thus acts as a permanent barrier. When ATP was added, the dimers detached at the rate of 42 s^{-1} i.e. kinesin waits at an obstacle for ~24 ms before detaching, the cycle time for wild kinesin.

Seitz and Surrey (2006) found that the motor waited at a temporary barrier.[40] *Drosophila* kinesins labelled with quantum dots were impeded by mutant dimeric kinesins having a cycle time of ~200 ms i.e. an order of magnitude slower than the wild-type. The effect of the mutants was to slow the wild-type kinesins in proportion to the mutant concentration though there was little effect on wild-type run length. They concluded that kinesin waits in a tightly bound state at an obstacle for at least 200 ms.

1.4 Kinesin procession mechanism

Kinesin steps along the dimers of a MT hydrolysing a single ATP molecule per step though the manner of this movement and the use that kinesin makes of the free energy of ATP hydrolysis are a matter of debate.[48; 49; 50]

1.4.1 Inching or toeing?

One controversy concerning stepping has been settled. Two incompatible ways for kinesin to step have been proposed: inchworm and head-over-head (HoH). Inchworm stepping is like the movement of the eponymous caterpillar: the trailing head steps up to the leading head then the leading head steps forward to the next binding site. Inchworm motion implies that the leading head always stays in front and only one head hydrolyses ATP. The alternative mode, HoH stepping, involves the trailing head passing around the leading head to the next binding site. In HoH motion, the heads change places at each step and both hydrolyse ATP.

Support for the inchworm hypothesis arises from a gliding assay performed by Hua, Chung and Gelles (2002).[51] They argued that, since kinesin is a homodimer, movement would be expected to be symmetric. In the case of HoH, the free head should pass the bound head on the same side at each step. This would mean that HoH stepping would rotate the stalk whereas inchworm would not. They immobilised a truncated *Drosophila* kinesin and measured the orientation of MTs moved along by it. No overall rotation was observed so they ruled out HoH motion in favour of the inchworm mechanism.

Further experiments using a different technique – fluorescent tagging – do not support the inchworm model. Yildiz et al. (2004) tagged one head of a kinesin molecule with a fluorescent probe and observed alternate step lengths of ~16 nm and 0 nm.[35] This result supports HoH, not inchworm as the latter involves equal-length steps. Taken together with the Hua et al. result, an asymmetric HoH model is favoured.

Evidence that supports asymmetric HoH stepping comes from bead assays. Block et al. (2003) showed that imposing sideways loads via a force-clamp

results in asymmetric slowing of kinesin: leftward loading (facing direction of motion) slowed kinesin more than rightward loading.[52] Asbury et al. (2003) engineered a set of mutant kinesins with truncated stalks.[53] They used an optical force-clamp to impose a constant rearward load of 4 pN and measured the dwell time (the time between steps). They found that truncation caused the motor to limp i.e. alternate dwell times increased, the shorter the mutant the more pronounced the limp. These results are only compatible with asymmetric movement since symmetric stepping (whether inchworm or HoH) would show no difference between alternate dwell times.

The generally accepted conclusion is that stepping is asymmetric HoH procession: kinesin walks in a similar manner to toeing a line (though, unlike a human, its "feet" are identical). The movement asymmetry is presumed to result from twisting of the stalk local to the heads. One step twists the base of the stalk biasing the next step to pass the opposite side and thus release the torsion energy. There is then no net twisting of the stalk in conformity with the results of Hua, Chung and Gelles (2002).[51]

1.4.2 Walking the line

To determine the route kinesin takes along the MT, Ray et al. (1993) conducted a gliding assay comparing the movement of MTs composed of different numbers of filaments.[54] A MT comprising 13 filaments is untwisted: the filaments line up in parallel to form the tube. 12 or 14 filaments will form MTs but have to twist to form a tube; a 12 filament MT spirals with opposite handedness to a 14 filament MT. The assay showed that 13-mer MTs glide along the bed of kinesins without rotating but 12-mers and 14-mers rotate as they glide though in opposite directions as expected if the filaments are being

followed by the kinesin heads. The conclusion is that kinesin walks along a filament rather than stepping across filaments.

A recent study adds a caveat this rule: kinesin sometimes steps over to an adjacent filament. Yildiz et al. (2008) labelled one head of a motor with a quantum dot and confirmed the filament-following behaviour recorded above except that 13% of stepping was sideways by ~6 nm with equal preference for right and left side-steps.[38]

1.4.3 Powering the motor

The detail of how kinesin uses the free energy generated by ATP hydrolysis to move forward is as yet unclear and so there are competing theories, one of which proposes a similar mechanism to that of myosin. Though the peptide sequence of myosin is radically different from that of kinesin, they both hydrolyse ATP and share a similarity in structure of the catalytic core.[55]

1.4.3.1 Myosin

Myosin-II is the non-processive molecular motor responsible for muscle contraction whose mechanochemical cycle is well understood. Muscle contraction consists in bundles of myosin-II molecules pulling on bundles of actin filaments in a cyclic, rowing-like motion. Figure 1.7 illustrates the myosin cycle. In the quiescent state the heads are bound to ADP and phosphate but prevented from binding actin as the binding sites are blocked by tropomyosin. Contraction is initiated by calcium ion influx triggered by the firing of the motor neuron that synapses to the muscle cell. Tropomyosin releases from actin to which myosin heads bind and then release their phosphate entering a rigor state: the motors are now tightly bound. The next event is release of ADP when the motors act as lever arms pulling the actin

filament in a power stroke. The cycle continues as ATP binds the empty head causing it to detach from the filament. The free head hydrolyses the ATP molecule causing the head to swing back thus completing the cycle.[56; 57]

1.4.3.2 Kinesin

The kinesin mechanochemical cycle has similarities to that of myosin but differs in detail. Kinesin in solution encounters the MT when one head binds to the MT. This causes the head to release its ADP while the other head remains free.[27] ATP binds the nucleotide-free head causing its neck linker to zipper to the head which results in the ADP-bound head binding to the next MT binding site.[58] This in turn causes ADP release followed by ATP binding and hydrolysis. Meanwhile the other head hydrolyses ATP, releases phosphate and detaches.[59; 60] This sequence repeats so that each head alternately steps forward and hydrolyses ATP: kinesin walks along the MT. Figure 1.8 illustrates the kinesin procession cycle as a series of states or snapshots.

As discussed below, there are two proposals for a stepping mechanism for kinesin. One proposal is that kinesin, though working to a different hydrolysis cycle, also uses a power stroke mechanism to advance along the microtubule. The alternative proposal regards kinesin as a Brownian motor whereby the free head is not pulled forward but follows a forward-biased diffusive path to the next binding site. Fundamental to both proposals is the phenomenon known as linker zippering.

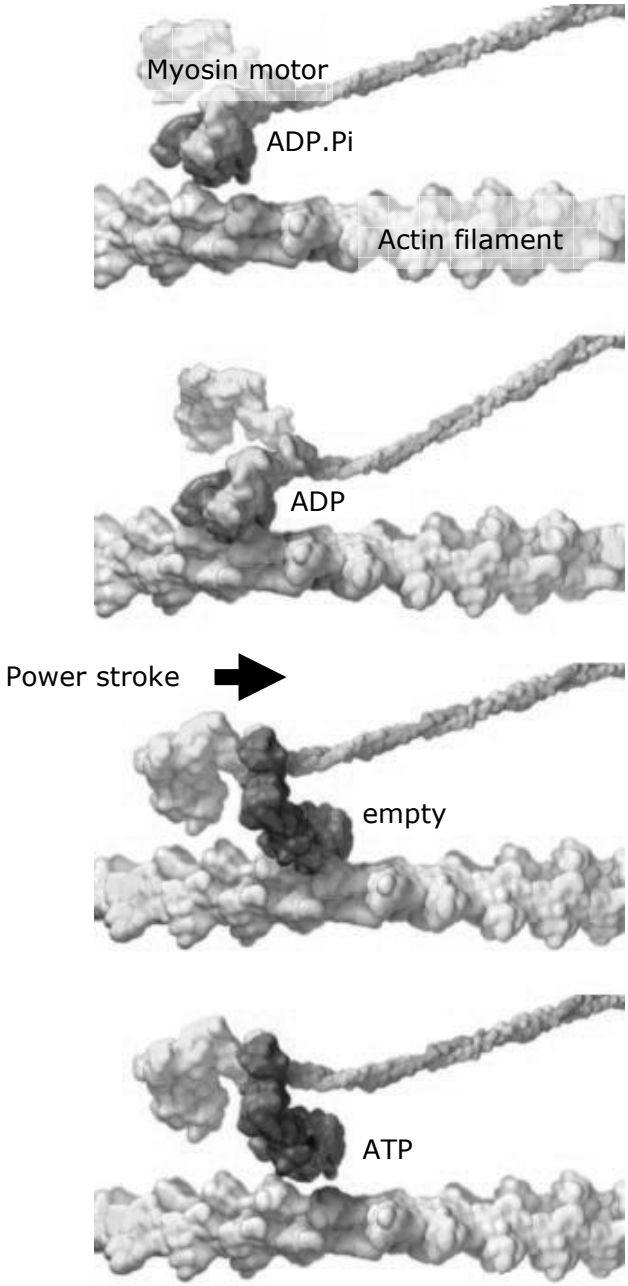

Figure 1.7 Myosin-II mechanochemical cycle as 4 snapshots. From the top: ATP has been hydrolysed and the head is diffusing; next, the head has bound the filament and phosphate has been released; next, after the head has pulled the filament to the right; lastly, ATP has bound and the head has detached. From Vale and Milligan (2000)[55].

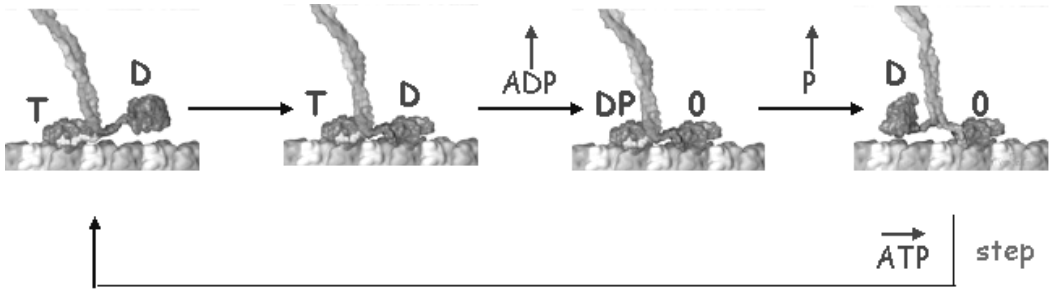

Figure 1.8 Diagram of kinesin processionary cycle. The letter above each head indicates which nucleotide is bound: D for ADP, T for ATP, DP for hydrolysed ATP, 0 for none.

1.4.3.3 Zippering

Rice et al. (1999) observed that ATP binding causes a conformational change in the normally flexible neck linker of monomeric kinesin: it becomes fixed (zippered) to the head and aligned in the direction of motion i.e. pointing towards the plus end of the MT.[58] This change of state of the linker has been confirmed in dimeric kinesin.[61; 62; 63] The importance of zippering for normal motion is demonstrated by a study showing that a non-zippering mutant kinesin failed to move regardless of ATP concentration.[38]

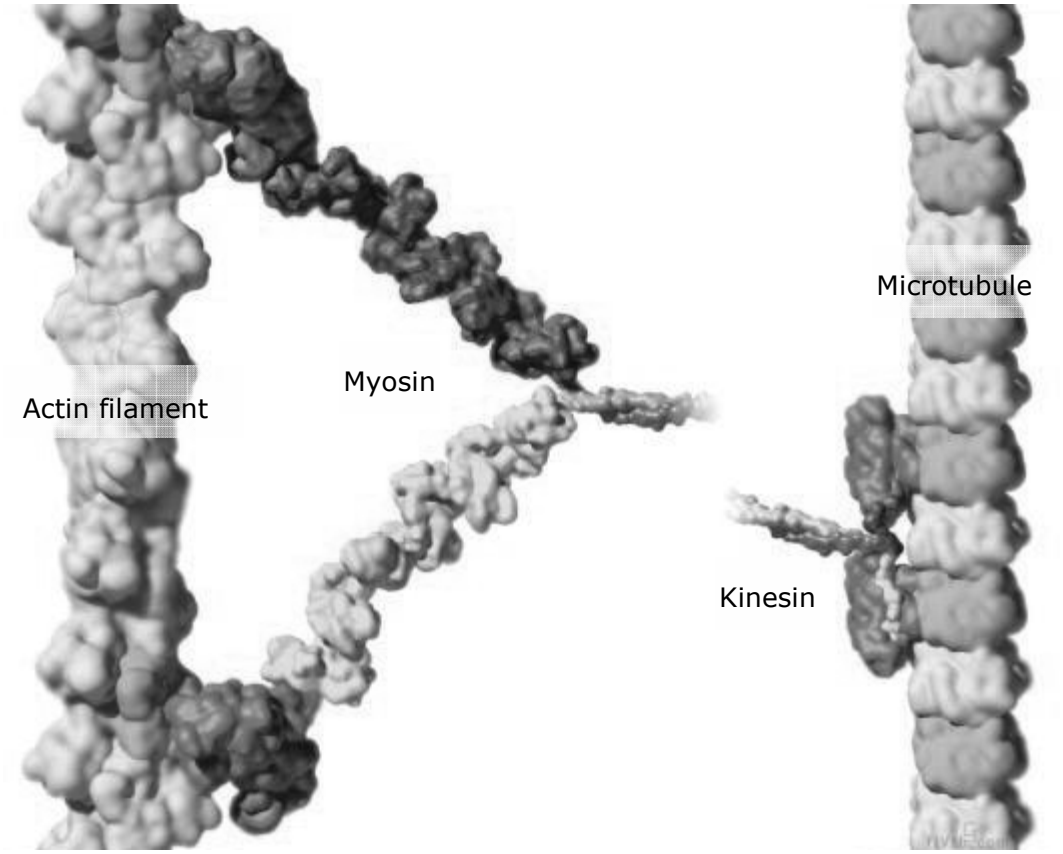

Figure 1.9 Myosin V motor domain with both heads bound to actin (left) compared to kinesin motor domain bound to a microtubule (right) at the same scale. (from Vale and Milligan 2000)[55].

1.4.3.4 Power stroke

Myosin-V, like kinesin, is a processional motor but is much larger, having a step of ~36 nm as opposed to kinesin's 8 nm.[64] Figure 1.9 illustrates the motor domains of myosin-V and kinesin bound to their respective tracks at the same scale. The linkers of myosin-V act as lever arms transmitting the power stroke in a twisting action to traverse the actin filament.[65] Though the flexible linkers of kinesin do not perform a similar function, Vale and Milligan (2000) proposed a power stroke mechanism for kinesin.[55] In their model, ATP binding causes linker zippering which in turn pulls the free head forwards via

its neck linker. The free head is then positioned close to the next binding site to which it diffuses and binds.

Subsequent energy calculations threw doubt on this scenario. Given a stall force of 6 pN together with a step of 8 nm, kinesin develops 48 pN.nm of work per step (29 kJ/mol) but Rice et al. (2003) calculated that the free energy of zippering is about 3 kJ/mol so the energy of zippering is insufficient to power such a step.[66]

A recent molecular dynamics study has revealed an extra component to zippering that may provide the necessary energy for a power stroke.[67] The neck linker is composed of two ß-strands (ß9 and ß10) connected by a hinge region. According to their simulation, ATP-binding causes a cover strand (9 residues long) to form a ß-sheet, the cover neck bundle (CNB), with the linker's ß9 strand. The CNB binds to the head initiating zippering of the remainder of the linker (the hinge region and ß10 strand latch) to the head. Their calculations indicate that CNB formation "may be responsible for generating the force for a walking stroke". Khalil et al. (2008) lend experimental support to this mechanism by showing that a mutant kinesin without a cover strand has drastically impaired procession.[68]

A power stroke would be expected to show up as a sub-step: the free head is first pulled forward by the power stroke and then diffuses to the binding site.[69] Analysis of noisy data from bead assays has proven inconclusive so the existence of a diffusive sub-step remains an open question. Coppin et al. (1996) found a substep of ~5 nm while Nishiyama et al. (2001) revealed 2 substeps of ~4 nm each, the first taking 25 microseconds followed by a slower substep.[70; 71] Carter and Cross (2005) used a more sensitive apparatus and failed to detect any substeps lasting >30 microseconds.[37]

1.4.3.5 Rectified Brownian motion

An alternative role for zippering has been proposed by Fox and Choi (2001).[72] In their model, stepping is achieved by rectified Brownian motion (RBM) as opposed to a power stroke. Rather than being a source of force, zippering provides directionality by forward biasing the otherwise random motion of the free head. The work done by kinesin in transporting its load is then powered by binding of the free head to the next site on the MT: ATP provides the energy to set the latch and not to drive the molecule forward. Rice et al. (1999) discuss a similar model in which "…force generation does not occur by a `power stroke' between two well ordered states, as is generally described for myosin. Instead, movement involves a transition from a disordered to an ordered state, with ATP binding providing the energy source for rectifying this Brownian ratchet."[58]

1.4.4 Wait state configuration

At normal physiological ATP concentration, stepping occurs in microseconds i.e. faster than can be measured by current techniques. Experimenters artificially slow kinesin down by reducing the ATP concentration to investigate the walking process. Kinesin then enters a wait state before each hydrolysis cycle: it has to wait for an ATP molecule to arrive at the empty head before taking the next step. There is controversy about kinesin's configuration in the wait state because experimental measurements point to different configurations. There is agreement that one head is bound to the MT, waiting for an ATP molecule to diffuse to its empty nucleotide binding site. The other head might be in any one of four possible positions: bound at the previous binding site, free to diffuse, parked, or bound at the next binding site. There is no conclusive evidence determining which configuration is correct.[73]

1.4.4.1 Both heads bound

Asenjo et al. (2003) used fluorescence polarisation microscopy to determine the orientation and mobility of labelled kinesin interacting with a MT concluding that both heads are bound in the wait state.[61] The same conclusion was reached by Yildiz et al. (2004, 2008) who observed alternate stepping by a fluorophore-tagged head of ~16 nm (two tubulin dimers) and 0 nm which rules out an intermediate position for a mobile head.[35; 38] These data are compatible with stepping occurring either before or after the wait state but both groups favour the latter whereby stepping forward of the trailing head occurs after ATP binds the leading head thus utilising linker zippering that results from ATP binding.

1.4.4.2 Free head

Bead assays provide evidence for the wait state being one head bound. Kawaguchi and Ishiwata (2001) found that the force required to detach kinesin from an MT in a nucleotide free solution is half that required when the motor is in an ATP analogue solution.[74] Uemura et al. (2002) found no difference between the force required to detach one and two–headed kinesins under no nucleotide conditions.[31] Guydosh and Block (2006) used a variation on the bead assay where the bead was attached to one head instead of the stalk.[75] All three studies indicate that kinesin waits with one head bound to the MT (nucleotide free) while the other head (ADP-bound) is free of the MT. The third study showed this result dynamically: the measurements were made while the motor was walking.

1.4.4.3 Parked head

Two electron microscopy studies show the wait state as one head bound with the other parked close to it.[76; 77] It is possible that these are artefacts

resulting from MT lattice saturation though there appears to be a conformational change in the bound head on ADP release that may provide a parking site.[29]

1.4.5 Head coordination

The question of head coordination was raised by the finding that less than four single headed kinesins bound to a bead fail to process while native kinesin processes yet has but two heads.[42] Tomishige and Vale (2000) chemically cross-linked the neck linkers and found that procession was defeated; severing the link restored normal procession.[78] Yildiz et al. (2008) engineered a series of mutant kinesins with lengthened neck linkers.[38] They found that speed reduced in proportion to the linker length and concluded that linker tension is an important factor in head coordination.

A factor generally regarded as necessary for coordination is a gating mechanism. In order for kinesin to process along the MT, the heads must step forward alternately with at least one head bound to the MT at all times. As the heads are identical, procession entails that the hydrolysis cycles of the heads are out of phase. Initial contact with the MT causes one head to release ADP which means that it starts its hydrolysis cycle ahead of its partner.[27] This phase difference must be maintained for the long run lengths observed (see section 1.3.3). Procession would be terminated if both heads detached at the same time or remained bound, unable to step. Several gating mechanisms have been proposed to account for head coordination.

1.4.5.1 Gated rear head

Hancock and Howard (1999) engineered a mutant kinesin lacking a head and measured the rate of detachment from a MT.[43] They found this was an order

of magnitude slower than wild-type kinesin. A similar result was obtained using fluorescence techniques by Crevel et al. (1999).[79] Both groups infer that linker strain generated by the leading head binding the MT accelerates the release of the trailing head from the MT in wild-type kinesin: binding of the leading head effectively gates release of the trailing head.

1.4.5.2 Gated front head

Rosenfeld et al. (2002, 2003) make an alternative proposal for coordination: ATP is prevented from binding the leading head by the strain in the linkers when the trailing head is bound.[59; 80] Studies using mutant kinesins lend support for this proposal. Farrell et al. (2002) engineered a mutant with a defective head unable to hydrolyse ATP and Klumpp et al. (2004) conducted experiments with a mutant that released phosphate without the head detaching after ATP hydrolysis.[60; 81] Both teams found that their motors failed to move after one hydrolysis cycle, attributing this behaviour to the leading head being unable to bind ATP. The implication is that, for wild-type kinesin, detachment of the trailing head is necessary to enable ATP binding to the lead head.

1.4.5.3 ADP release gate

Hackney (1994) found that kinesin releases only one ADP in contact with MT.[27] This implies gating of ADP release from one of the heads. The mechanism for this gate is controversial and has a bearing on the wait state discussion.

Asenjo et al. (2003) and Mori et al. (2007) propose that linker orientation gates ADP release: only when the linker is bent backwards is ADP release enabled.[61; 82] This is the case when both heads are bound to the MT with the ADP-bound head in the lead. This mechanism would explain Hackney's finding

as the trailing head would retain its ADP because its linker would be bent forward. It could also serve to coordinate the heads during procession by stopping the trailing head from prematurely releasing its ADP which would initiate a futile hydrolysis cycle.

Alonso et al. (2007) argue that the ADP gate operates by way of a parking mechanism and not as a result of linker orientation.[83] They found that kinesin released only one ADP even when unpolymerised tubulin dimers were substituted for the MT; the second ADP was retained until ATP was introduced. The linker is not bent backward in this situation because this only happens when both heads bind the MT. They propose that, in the wait state, the ADP-bound head is parked such that its MT-binding site is blocked and cite electron microscopy evidence for this configuration. The gate is opened by the arrival of ATP which binds to the partner head causing the release of the parked head and allowing it to bind tubulin.

Chapter 2 Modelling kinesin

> "… a good computational model–if one can be found–may explain the mechanisms behind a biological system in more intuitive and more easily analyzable terms than a mathematical model."
>
> **Fisher and Henzinger 2007**[84]

This chapter motivates and describes the computational simulation developed for the study. Laboratory experiments have revealed many aspects of kinesin but not the detailed mechanism of motion. Mathematical modelling has been employed in an attempt to improve our understanding of the motor: it has been explored in terms of its chemical kinetics or as a Brownian ratchet. While both approaches can generate realistic behaviour, the details of kinesin's walk remain moot. The methodology used here draws on aspects of this previous research but takes a computational approach. A program has been designed and built to simulate a section of microtubule (MT) traversed by linked kinesin heads modelled as simple agents. The aim is not just to understand kinesin's walk but also to pilot a computational framework for modelling AT.

2.1 Why model kinesin?

As described in the previous chapter, a variety of experiments has revealed aspects of how kinesin transports cargo along the axon but the movement has yet to be observed in detail. Analysis of single-molecule experimental data has provided the overall picture of kinesin motor domains (heads) stepping in a head-over-head manner but there remain competing theories of how this is achieved. Models of kinesin have been devised to formally investigate theory and to complement experimental work in order to improve our understanding of how the motor works.

2.2 Previous models of kinesin

The two main approaches to modelling the dynamics of linear molecular motors such as kinesin are Brownian (continuum or thermal) ratchet and chemical-kinetic (discrete stochastic) theory.[85] These are simplified mathematical models consisting of systems of differential equations relating measurable quantities and describing how they change over time. Parameter fitting enables these models to generate results, such as load-velocity curves, that are a good fit with those of experiment. A third approach, molecular dynamics, could provide a much more detailed picture of kinesin but has had limited application so far because of computational power limitations.

2.2.1 Brownian ratchet

Kinesin has been modelled as a twin-head chemically-driven Brownian ratchet.[86; 87; 88; 89] The basic principle of a Brownian ratchet is that a particle diffusing according to a potential that changes from a symmetric to an asymmetric form results in net directional movement of the particle towards the base of the nearest potential well. Figure 2.1 illustrates how particles are influenced by switching the potential from flat to sawtooth. In a chemically-driven ratchet, the selection of the potential depends on the chemical changes that occur during ATP hydrolysis; for kinesin, the binding of ATP causes the potential to switch.

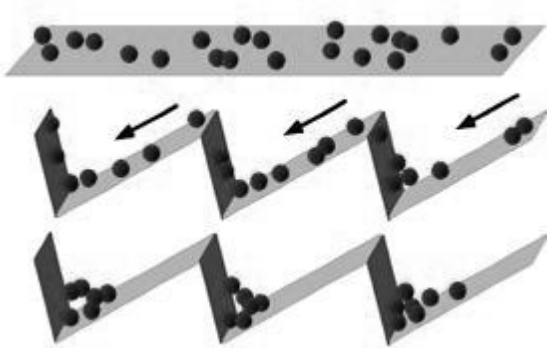

Figure 2.1 Brownian ratchet mechanism. Top: particles firstly diffuse at random in a flat potential; middle: the potential changes to a sawtooth when more particles diffuse left than right; bottom: the result is that particles accumulate in potential wells.

2.2.2 Chemical-kinetic

Kinesin steps along the dimers of the MT hydrolysing a single ATP molecule per step. Discrete stochastic models approximate this processionary cycle by a set of discrete states linked by rate constants. Each state represents a point in the hydrolysis cycle and the corresponding position of kinesin. Figure 2.2 illustrates the basic scheme. Kinesin has been modelled as a whole[90; 91] or as separate heads[92; 93; 94; 95] incorporating the influence of neck linkers or head position on the rate constants. Some models incorporate physical constraints on the heads, connecting them via springs[96; 97] or a hinge[98; 99].

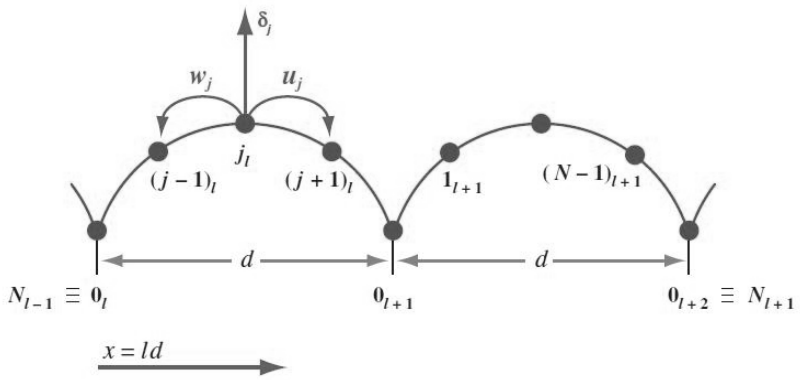

Figure 2.2 Chemical-kinetic schema. The motor steps to the right by distance d from MT binding site, l, to the next. The N chemical states leading to a step are depicted as circles joined by an arc. At any state, j, the motor moves to the next state or the previous state or detaches as determined by rate constants u_j, w_j and δ_j respectively. Kolomeisky & Fisher (2007)[85] figure 5.

2.2.3 Molecular dynamics

Molecular dynamics (MD) is a method that uses Newton's laws to simulate molecules at the atomic level and thus might seem to be a promising tool to investigate kinesin in motion. Though it is possible to construct an MD model of a kinesin motor traversing a section of MT, animating the model is presently an insurmountable problem. This is because current computing power and precision restrict MD simulations to nanoseconds but each step of kinesin takes milliseconds. An additional difficulty is that ATP hydrolysis is a chemical process of bond breaking which is not within the scope of MD. A computationally intensive quantum-mechanical description of the hydrolysis would be required.

MD has been useful on a much more modest scale. A series of simulations of an isolated kinesin head[67] has revealed changes associated with zippering of

the neck linker involving a previously overlooked structure, the cover neck bundle, for which experimental evidence has since been obtained[68].

2.3 The approach of this study

The present approach to modelling kinesin differs from previous attempts which assume procession and a mechanism of stepping. Procession is not built into the program but emerges only if the heads coordinate. The model is designed to compare the behaviour of the system under different stepping mechanisms.

The long-range aim of this research is to investigate the failure modes of the axonal transport system. The present work, in addition to studying kinesin motion, pilots a methodology for a simulation of cargo transport along an axon.

2.3.1 Executable biology and agent-based modelling

The methodology used in this study can be described as executable biology. Fisher and Henzinger (2007) contrast executable biology with mathematical modelling.[84] Mathematical models use equations to describe the relationship between quantities that change in value over time whereas executable biology uses executable computer algorithms to mimic biological phenomena.

To illustrate the rationale for making this distinction, Fisher and Henzinger liken biological systems to complex digital electronic circuits. Differential equations are used to describe the function of the transistors that comprise electronic chips and to build mathematical models of biological systems. Digital system designers do not work at the level of transistors but at the next

level of abstraction – logic gates. Logic gates, consisting of several transistors, are represented not by differential equations but truth tables of inputs and outputs, each with a Boolean value (0 or 1). Just as engineers rely on high-level simulation tools to design sophisticated digital systems so, the authors argue, a similar toolset of biological models would greatly benefit our ability to understand biological systems (see the quotation at the start of this chapter).

It is intended that the work described in this dissertation will form the nucleus for an executable biology of axonal transport. With a view to expansion, an agent-based approach has been taken: further agents may be added to extend the scope of the simulation. The approach is to model kinesin as a system of simple, interacting components detailed enough to capture aspects of the molecule important with respect to walking but simple enough to be readily understandable by biologists and not require large amounts of computer power to simulate.

2.3.2 Modelling the stepping mechanism

Previous models of kinesin assume the stepping mechanism to be a power stroke or a Brownian ratchet or do not address stepping at all. The current study explicitly models kinesin as a pair of coupled heads and enables the comparison of stepping mechanisms.

2.3.2.1 Brownian ratchet

A Brownian ratchet was considered as a stepping mechanism but rejected because there is a fundamental problem applying this mechanism to kinesin. The directionality of a Brownian ratchet motor derives from the asymmetric potential between the motor and the track and so motors with similar heads

would be expected to move in the same direction along the same track. Ncd is a minus-end directed kinesin motor yet shares highly conserved nucleotide- and microtubule-binding motifs with plus-end directed kinesins.[41] Further evidence confounding the ratchet model is that experiments on mutant motors show that the crucial element determining direction is the neck linker and not the MT-binding sites. Ncd motors having a single mutation in the neck linker became bidirectional[100] while Ncd mutants whose heads were replaced with those of kinesin-1 maintained their minus-end directionality[101].

2.3.2.2 Rectified Brownian motion

A different diffusive mechanism, rectified Brownian motion (RBM), does not suffer from the above difficulty as it is the linker that provides directionality and not an asymmetric potential. In a chemically-driven ratchet (section 2.2.1), the energy from ATP hydrolysis is used to switch the potential influencing the head from a symmetric to an asymmetric form such that the head preferentially diffuses forwards. The RBM mechanism does not depend on the shape of potential biasing free head diffusion, instead ATP binding works through the linker to latch the head in the forward position. ATP binding causes a conformational change in the head such that, when the free head diffuses forward, the linker zippers to the head. Thus zippering biases the diffusive motion of the free head towards the plus end of the MT. A second advantage of RBM is that the force of the step does not depend on zippering energy (as discussed in section 1.4.3) or Brownian motion but the binding energy of the free head to the MT.[102]

2.3.2.3 Power stroke

The power stroke theory envisages the motor using the energy of ATP hydrolysis to pull the free head forwards. Though there are difficulties with this model it remains a possible mechanism as discussed in section 1.4.3.

2.4 The simulation

The program is written in the C language and implements a discrete event-driven simulation with a fixed-increment clock. The simulation space represents a two-dimensional section of cytosol containing the motor and a MT filament. The filament is modelled as a one-dimensional lattice of binding points. The motor is modelled as twin kinesin heads. Each head is a finite state machine whose states are the position, nucleotide- and MT- binding possibilities. The state transitions are governed by simple rules. The detailed physical relationships of and between the motor components are not modelled.

A simulation run starts with both heads ADP-bound positioned close to the minus end of the MT. Pseudo-random motion is applied to each head to approximate diffusion until the motor engages with the MT. After the motor engages the MT, the rules come into play and procession may result. If at any point both heads detach, diffusion is resumed. The simulation run is terminated when the motor reaches the plus end of the MT or becomes stuck with both heads permanently bound. Results are output to file for spreadsheet analysis.

2.4.1 Head simulation

The heads are treated as identical simple agents, following the same hydrolysis and binding rules. Each head is modelled as a separate finite state

machine having a position along the MT and one of five possible states of nucleotide and MT binding at any one time. The five states are denoted by KD, K0, KT, KDP and KDu which represent, respectively, a kinesin head bound to ADP, to no nucleotide, to ATP, to hydrolysed ATP, (all bound to the MT) and to the free ADP-bound head (the lower case 'u' is the initial of 'unbound' as the head is not bound to the MT). During procession, each head performs a cycle of transitions:

$$KDu \rightarrow KD \rightarrow K0 \rightarrow KT \rightarrow KDP \rightarrow KDu..$$

These transitions are reversible but since the forward rates are estimated to be at least 2 orders of magnitude greater than backward rates under normal physiological conditions, reverse transitions are ignored in the simulation.[103]

The following series of rules embody the hydrolysis cycle and the interaction between individual head and MT as described in section 1.4.3.2. Figure 2.3 shows the finite state machine corresponding to these rules.

1. If an ADP-bound head encounters the MT, it binds (KDu → KD)
2. Binding to the MT causes ADP release (KD → K0)
3. ATP binds the empty head (K0 → KT)
4. The bound head hydrolyses ATP (KT → KDP)
5. Head detachment occurs with phosphate release (KDP → KDu).

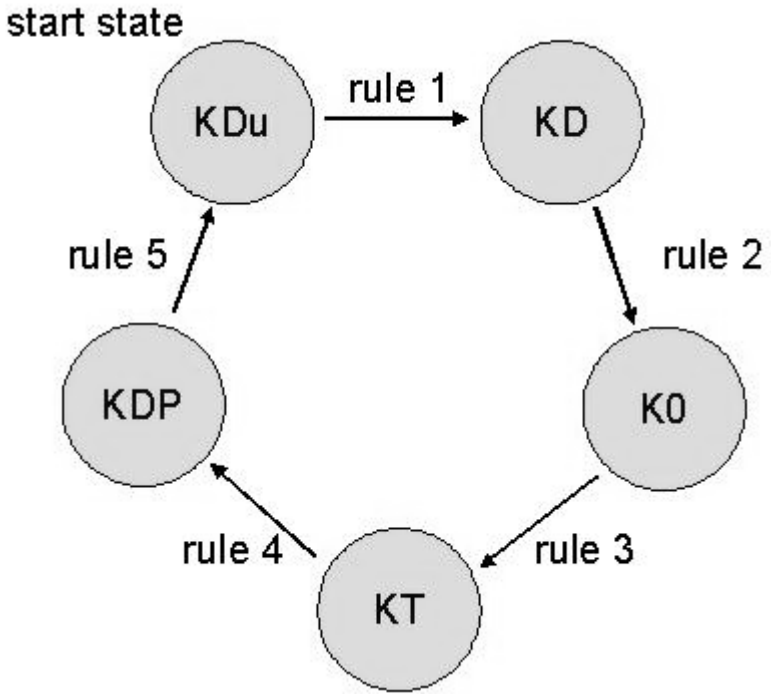

Figure 2.3 Head state transition table.

Procession is not built in to these rules: only when the heads coordinate does procession occur.

2.4.2 State transition (event) timings

In the first part of the study, as described in chapter 3, the simulation event timings are variable rather than fitted to estimated timing data derived from experiments. The relative amount of time a head remains in a particular chemical state is varied and the simulation run to see under what range of timings procession arises. One advantage of this strategy is that different stepping mechanisms can be compared in terms of the range of timings under which procession emerges regardless of whether the timings are realistic.

In the second part of the study, described in chapter 4, a more realistic approach was taken to event timings including taking into account the fact that the arrival of an ATP molecule is a random, diffusive event. The

probability of arrival in a given time interval, which increases with the concentration of ATP molecules, can be modelled by the Poisson distribution:

$$P(k, \lambda) = \lambda^k e^{-\lambda}/k!$$

Where k is the number of times ATP arrives in the time interval when the mean rate of arrival, the expected value, is λ. The ATP binding timings were generated according to a Poisson distribution using Knuth's algorithm.[104] The remaining event timings were fixed and increased from the baseline values applied in previous virtual experiments in order to allow short ATP binding timings i.e. the simulation of high [ATP]. Their ratio approximates those listed in Rosenfeld et al. (2002).[80]

2.4.3 Wait state

There is a point in the processionary cycle when kinesin is said to be in a wait state because the bound head is awaiting ATP binding (as discussed in section 1.4.4). The controversial assumption made here is that the wait state comprises one head bound and one head free. This is the same state as kinesin's first encounter with the MT. During procession, the wait state occurs after rule 2 has been applied to one head and rule 5 has been applied to the other. One head is then nucleotide free and bound to the MT awaiting ATP and the other is ADP bound and diffusing (subject to restraint by the neck linkers).

2.4.3.1 Linker strain and head binding

In the wait state, the movement of the free head is diffusive and simulated by a pseudo-random number function such that there is an equal probability of the head moving forwards or backwards. The neck linkers are assumed to behave as entropic springs. Entropic strain is generated by thermal motion of the single polypeptide strings composing the neck linkers in solution making

them behave like elastic bands which results in the heads being unlikely to reach either binding site when the motor is in the wait state. Rice et al. (2003) suggest that entropic linker strain prevents the free head binding the MT following detachment.[66] Since occasional backstepping has been observed during procession (see section 1.3.4), it is proposed here that linker strain in native kinesin is not strong enough to prevent re-binding but does make it unlikely.

The linkers are implicitly modelled here through their effects on binding. The linker strain parameter is allowed to take unrealistic values in order to explore the relation between strain and kinesin's behaviour. Linker strain is assumed to influence the likelihood of the free head binding the MT in the wait state. The probability of binding varies with the strain according to the formula:

$$P(\text{binding}) = 1 - \text{strain} / \text{maximum strain}$$

Thus, at maximum strain, P(binding) is 0 i.e. the free head cannot reach either binding site. This linear relationship is not to be confused with the physical relationship between head distance and entropic linker strain, which is exponential.[102]

2.4.4 Stepping

Kinesin traverses the MT by "walking" along a filament: one head steps forward while the other is fixed to the MT (the head-over-head mechanism).[105] Since this motion has yet to be observed, there are differing opinions as to how this stepping is accomplished. Two proposed mechanisms are compared in this study: rectified Brownian motion (RBM) and power stroke (PS).

Stepping happens after rule 3 is applied to one head if the partner head is free. The behaviour of the simulation differs depending on the stepping option used. The RBM option is implemented with a zippering switch. The switch is set when rule 3 is applied (ATP binds) and reset when rule 5 is applied (phosphate is released). Thus activation of the switch simulates the setting up of zippering of the neck linker to the bound head and resetting the switch simulates the linker unzippering. If the free head diffuses forwards while the zippering switch is set then a step is taken. The power stroke option forces a step on application of rule 3 unless the partner head is bound. The assumption is that the power stroke is not sufficient to pull forward an MT-bound trailing head.

2.4.5 Load

The effect of hindering load is simulated by altering the operation of the zippering switch. Loads less than 4 pN are assumed to have no effect on zippering, the probability of zippering is progressively reduced as the load is increased from 4 pN to 7 pN, and loads above 7 pN prevent zippering. A small random variation is applied to simulate dynamic load variation expected through stalk springiness. Loading only affects zippering: there is no attempt to simulate any effect load may have on head binding.

2.4.6 Gating

Gating may be applied to investigate the difference between gated and ungated models. It is implemented differently depending on which model is being used.

For the RBM model, the gating changes the action of rule 4 (see section 2.4.1). The gate is implemented by slowing ATP hydrolysis tenfold unless the

partner head is bound to the forward binding site. This mirrors the experimental finding that single-headed kinesin hydrolyses ATP ten times slower than native kinesin.[43] The idea is that linker strain speeds up hydrolysis during normal procession as both heads are bound to the MT when ATP is bound and, in this configuration, the linkers are fully extended. If the free head is prevented from binding then this strain is missing and kinesin behaves as if it were single-headed.

For the PS model, gating affects rule 3 (see section 2.4.1). The gate is implemented by preventing ATP binding the leading empty head while both heads are bound to the MT. This modification implements the proposed mechanism described in section 1.4.5.2 which is also based on linker strain.

2.4.7 Obstacle

In order to investigate the behaviour of the motor on encountering an obstacle, a blockage can be placed on the MT for a given time interval. This obstacle prevents the motor from reaching the next binding site on the MT.

2.4.8 Multimotor simulation

The last part of the study (see section 4.2) concerns linked motors simulating the case where more than one motor is bound to the same cargo. A flexible linkage between two motors was implemented using the existing loading mechanism (section 2.4.5). Loading was applied if the processing motors deviated from the initial separation distance. If the motors moved apart then a load was placed on the leading motor in proportion to the increased distance. If the inter-motor distance decreased below the initial separation, a proportional load was placed on the trailing motor. Thus the two loadings

operated to maintain the initial distance between the motors throughout the run.

2.4.9 Visual interface

The state of the system is displayed in a graphic window so that the experimenter can keep a visual check on the system's behaviour. Figure 2.4 shows a snapshot of the display. The heads are shown as blobs coloured according to nucleotide binding: red represents KT, green represents KD. These are contained within a two dimensional box representing an area of cytosol (pale grey) containing a length of MT filament laid out laterally as alternate α and β tubulins (brown blobs) along the base of the box. Previous positions of the heads are shown in dark grey to give a trace of the path of the motor.

Figure 2.4 Program display screen snapshot. The MT is depicted in brown, the motor heads in red and green, and the track of the motor in dark grey.

Chapter 3 Results – fixed ATP arrival

Virtual experiments were conducted to investigate the mechanism of stepping of kinesin using computational simulation described in the previous chapter. Two stepping theories were compared for robustness in terms of their ability to generate procession (continuous stepping) under differing timing conditions and variable linker strain. Rectified Brownian motion (RBM) was found to be a more robust stepping mechanism than power stroke (PS). Gating of the chemical cycle has hitherto been thought necessary to coordinate the heads and so prevent detachment of kinesin from the MT. The simulated motor achieved procession without gating, however, and showed realistic behaviour in response to load.

3.1 Published results

Results described here have been peer-reviewed and published in the IEEE Proceedings of the 2008 European Modelling and Simulation Symposium and in the journal BioSystems (see appendix C). The first paper introduces the simulation and its use in comparing stepping models and in investigating the behaviour of kinesin at a blockage. An ungated RBM model is proposed but a possible role for an ATP hydrolysis gate is also suggested as serving to make the motor wait at an obstruction. The second paper describes the effect of hindering load on the motor using RBM stepping *in silico* and compares this behaviour to *in vitro* experiments with and without an ATP hydrolysis gate. The argument is made that kinesin both employs ungated RBM stepping and does not wait at an obstacle.

3.2 Four part study

A computer simulation was designed and built to investigate how kinesin walks. The main program is listed in appendix B.1, supporting functions are listed in appendices B.4, B.5, B.6, and B.7. The motor is modelled as a system of two motor domains (heads) restrained by linkers and interacting with the microtubule (MT) and nucleotides according to a set of rules as described in section 2.4. Linker strain is treated as a variable when examining its effect on kinesin's behaviour (parts one and two of this study), but is otherwise fixed at an operational value of 9.5 which gives a level of backstepping matching that observed *in vitro* (see section 1.3.4). The arrival time of ATP is treated simplistically (as a fixed value per run) here but is modelled more realistically in the subsequent work described in the next chapter.

This part of the study may be divided into four parts:

1. Determination of the conditions for procession, comparing stepping models
2. Investigation of the hypothesis that gating is not necessary for procession
3. Exploration of the effect of hindering load on stepping characteristics
4. Investigation of the effect on kinesin of a blockage placed on the MT.

3.2.1 Part one – PS vs RBM

The protocol for the simulation runs for part one is summarised in the following pseudo-code:

Initialise program:

Select power stroke stepping rule

Zero counters.

For linker_strain = 0 to 10 do

For all timing combinations do

If motor fails to process then increment pf_counter

If motor detaches then increment d_counter

Output counter values to file and reset counters to 0.

Repeat above but with RBM stepping rule selected.

The procession results (pf_counter values) from 5 runs for each stepping mechanism over linker strain range are listed in table 3.1. The average values are plotted in figure 3.1 as diamonds for RBM values and squares for PS values. A linear trend line fits the PS data while an exponential trend line fits the RBM data.

Table 3.1 Timing combinations yielding interrupted procession – PS and RBM.

	PS					RBM						
Linker strain	*1*	*2*	*3*	*4*	*5*	*1*	*2*	*3*	*4*	*5*	**PS Av.**	**RBM Av.**
0	15	16	13	14	14	15	15	15	16	15	**14.4**	**15.2**
1	11	12	14	10	13	9	8	9	8	7	**12**	**8.2**
2	13	9	10	10	12	5	2	5	2	2	**10.8**	**3.2**
3	10	8	12	9	5	2	0	0	2	2	**8.8**	**1.2**
4	6	10	10	6	10	0	1	3	2	0	**8.4**	**1.2**
5	8	5	6	9	7	0	0	2	2	0	**7**	**0.8**
6	7	6	6	8	6	0	0	0	0	0	**6.6**	**0**
7	6	5	4	4	2	0	1	0	0	0	**4.2**	**0.2**
8	3	2	5	5	6	0	0	0	0	0	**4.2**	**0**
9	2	3	2	2	1	0	0	0	0	0	**2**	**0**
10	1	1	1	1	1	0	0	0	0	0	**1**	**0**

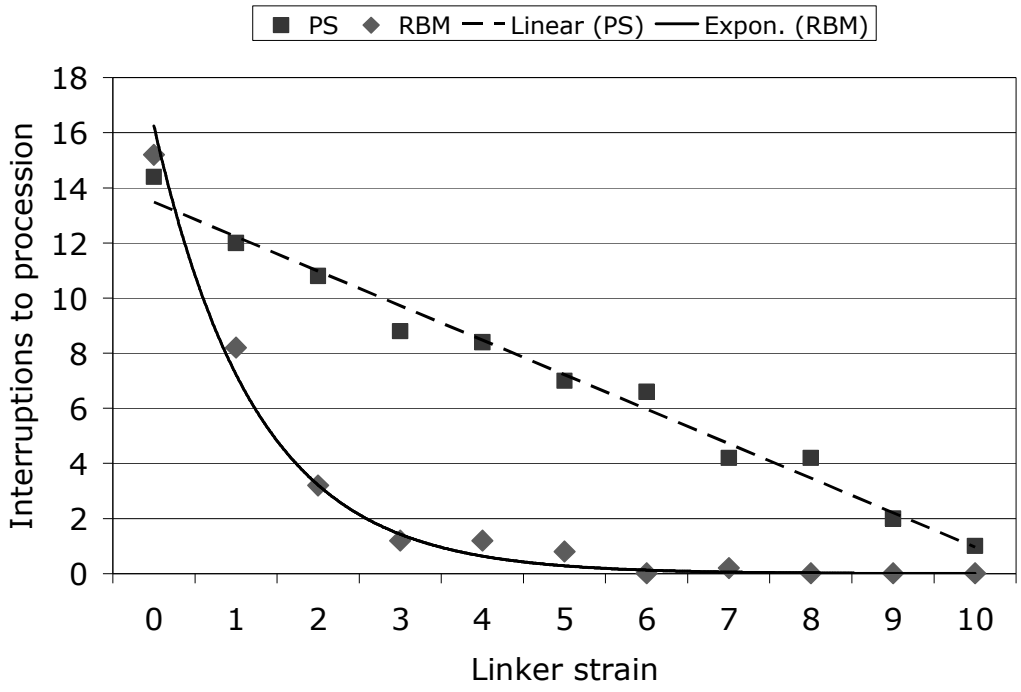

Figure 3.1 Relationship between ungated stepping mechanisms and procession under varying linker strain.

The detachment results (d_counter values) from 5 runs for each linker strain are listed in table 3.2. The average values are plotted in figure 3.2 where diamonds are RBM values and squares are PS values.

Table 3.2 Detachments for each stepping mechanism.

	RBM					PS						
Linker strain	*1*	*2*	*3*	*4*	*5*	*1*	*2*	*3*	*4*	*5*	**RBM Av.**	**PS Av.**
0	7	8	5	6	4	8	8	4	3	5	**6**	**5.6**
1	1	1	1	0	1	2	2	4	2	5	**0.8**	**3**
2	0	0	0	0	0	1	2	2	2	2	**0**	**1.8**
3	0	0	0	0	0	1	1	2	1	1	**0**	**1.2**
4	0	0	0	0	0	1	1	1	0	2	**0**	**1**
5	0	0	0	0	0	1	0	0	2	1	**0**	**0.8**
6	0	0	0	0	0	1	1	1	1	1	**0**	**1**
7	0	0	0	0	0	1	2	1	1	0	**0**	**1**
8	0	0	0	0	0	2	2	2	2	1	**0**	**1.8**
9	0	0	0	0	0	3	2	2	1	1	**0**	**1.8**
10	0	0	0	0	0	1	2	4	3	2	**0**	**2.4**

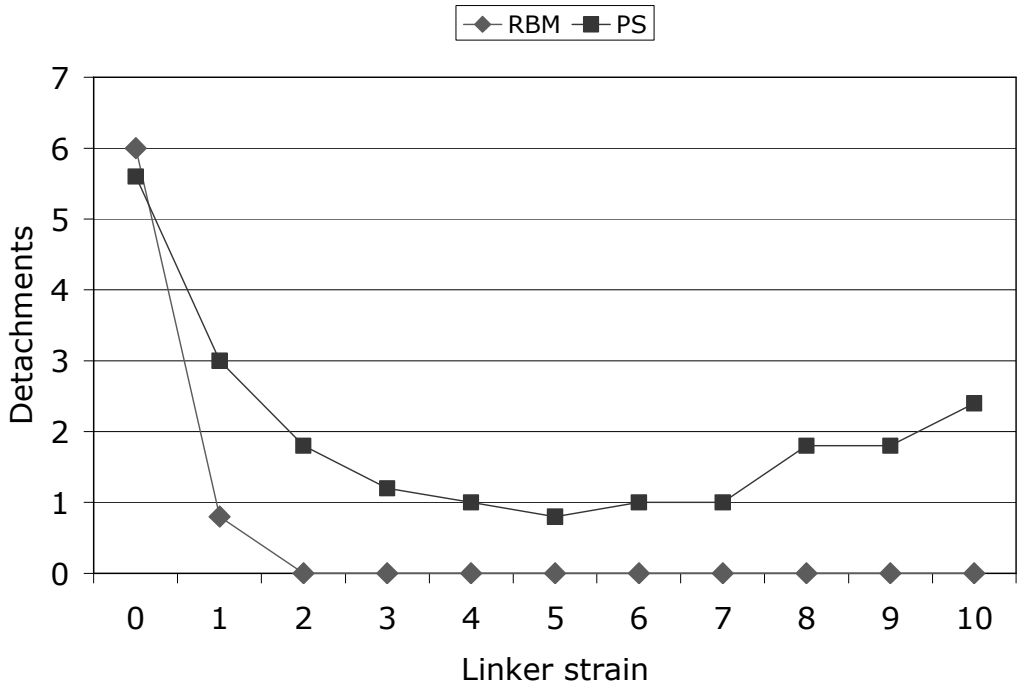

Figure 3.2 Relationship between stepping mechanisms and detachments under varying linker strain.

The overall result was that procession emerged with both PS and RBM stepping mechanisms but RBM gave rise to procession under a wider range of timings and linker strains than PS.

3.2.1.1 Discussion – system phases and timings

Procession occurs when the heads take it in turns to detach and step forward. This requires that at least one head is bound to the microtubule at all times and that the trailing head is free to step forwards when the stepping mechanism comes into force. If the first condition were not met then kinesin would diffuse away from the microtubule. If the second condition were not met then the motor would stall on the MT. Viewed as a system, a motor can have three phases corresponding respectively to the three types of behaviour: procession, diffusion and stalled.

Varying the timings of head binding and hydrolysis events with the PS stepping mechanism resulted in the system displaying all three phases. Procession arose under a specific ratio of timings described by the formula

$T1 + T2 = T3 + T4 + T5.$

Where T1 is the time taken for ATP hydrolysis (KT → KDP), T2 is the head detachment time (KDP → KDu), T3 is the time for head docking (KDu → KD), T4 the time for ADP release (KD → K0) and T5 the time for ATP binding (K0 → KT). Note that the stepping time is not included in this equation as a step is completed in microseconds while dwell time is measured in milliseconds.

The RBM stepping mechanism displayed only two phases: procession and diffusion.

3.2.1.2 Discussion – procession and detachment

The RBM mechanism was found to be more likely to produce procession than the PS at any non-zero linker strain. The results of comparing mechanisms are plotted in terms of the timing conditions under which the motor processed (figure 3.1) and the number of detachments recorded per run (figure 3.2) over a range of linker strain.

Figure 3.1 plots the number of timing combinations that did not result in procession against linker strain. This orientation matches that of the detachment plot as, in both cases, the lower the plotted value the better the motor is performing. With RBM stepping, procession occurred under all timing combinations for high linker strain (values above 7) while with PS, the motor failed to process under some timing combinations at any strain value. The data trends show a linear relationship with linker strain for the PS and an

exponential relation for RBM, reinforcing the marked difference between the two types of stepping. Both sets of data indicate that linker strain assists head coordination (as discussed in the next section).

Figure 3.2 plots the average number of detachments recorded for each timing combination that produced some procession (at least a second step) against linker strain. Again, RBM stepping shows a consistently lower incidence of detachments than PS, none below a strain value of 2. This is despite the plot over-estimating the performance of the PS since some of its timings resulted in a stalled motor.

3.2.1.3 Discussion – explanation

These results can be explained by considering the nature of the mechanisms. In both, ATP binding results in the neck linker zippering to the MT-bound head; the free head is then positioned close to the next MT binding site thus facilitating the next step. PS achieves this in a different way from RBM.

It has been assumed here that the PS is impulsive: when ATP binds the lead head, the rear head is pulled forward as described by Vale and Milligan (2000)[55]. RBM operates differently: ATP binding sets up zippering but the trailing head diffuses forwards rather than being pulled by the linker. Thus PS stepping occurs when ATP binds whereas RBM stepping can occur any time between ATP binding and phosphate release, after which zippering does not occur[58]. The time window for potential stepping is thus much wider with RBM than PS so RBM stepping occurs over a wider range of timing conditions than PS stepping. In view of this relative advantage, further experiments were conducted to compare gated PS with RBM, the results of which are described in the next section.

3.2.1.4 Gated PS compared to RBM

The experiments performed above were re-run to compare both gated and ungated PS with ungated RBM. Figure 3.3 shows the average values of the results, as listed in table 3.3, plotted as squares for PS data, lozenges for RBM data, and circles for gated PS data.

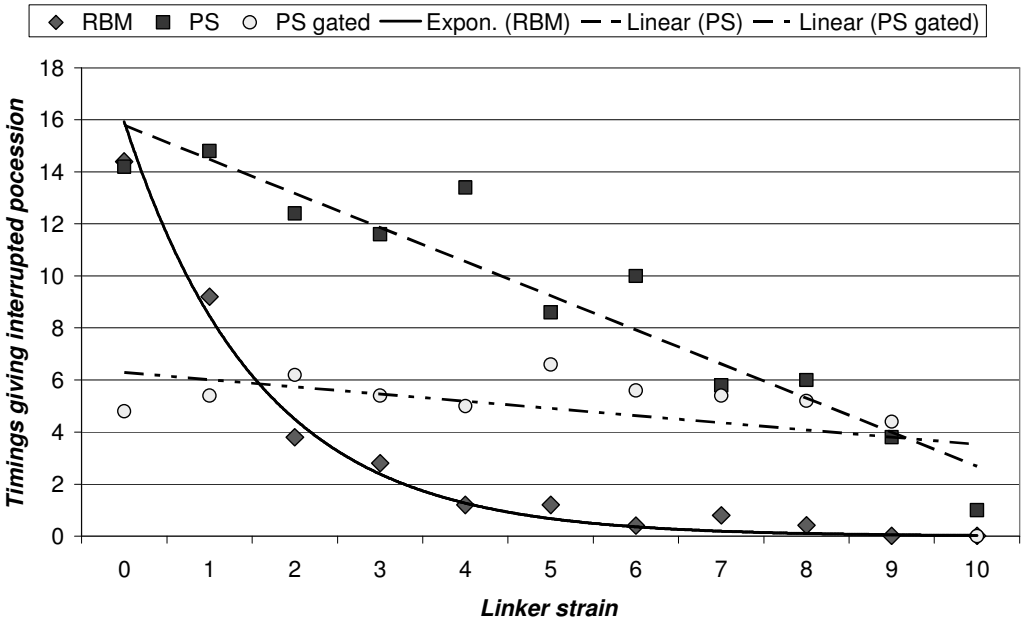

Figure 3.3 Relationship between stepping mechanisms and procession under varying linker strain.

Gating brings the PS model up to the performance of the RBM model at maximum linker strain, and makes for better performance at very low values (under 2), RBM remains on top for most of the range.

The primary reason for the interrupted procession recorded for gated PS is the effect gating has when the motor steps back (which it cannot do at maximum strain): the motor freezes on the MT. As soon as a backstep is taken ATP cannot bind the leading head as this is prevented by the gate. The trailing

head, on engaging the MT, ejects its ADP and both heads remain bound to the MT.

Table 3.3 Timing combinations yielding interrupted procession.

	Run					**Averages**
Linker strain	*1*	*2*	*3*	*4*	*5*	**PS gated**
0	4	4	3	3	4	**3.6**
1	6	5	5	5	6	**5.4**
2	7	5	7	6	6	**6.2**
3	5	5	7	5	5	**5.4**
4	5	6	5	5	4	**5**
5	5	5	5	5	5	**5**
6	5	6	6	6	5	**5.6**
7	6	5	6	5	5	**5.4**
8	5	6	5	5	5	**5.2**
9	4	5	4	4	5	**4.4**
10	0	0	0	0	0	**0**
						PS
0	15	13	13	14	14	**13.8**
1	12	15	15	15	13	**14**
2	14	15	15	14	13	**14.2**
3	12	15	11	12	11	**12.2**
4	13	15	11	12	9	**12**
5	10	14	14	13	11	**12.4**
6	8	7	10	12	11	**9.6**
7	10	10	10	8	6	**8.8**
8	4	7	7	5	5	**5.6**
9	2	6	3	4	2	**3.4**
10	1	1	1	1	1	**1**
						RBM
0	14	16	13	15	14	**14.4**
1	10	7	9	8	12	**9.2**
2	3	3	7	2	4	**3.8**
3	3	2	4	3	2	**2.8**
4	2	1	1	2	0	**1.2**
5	1	0	0	1	4	**1.2**
6	1	0	0	1	0	**0.4**
7	2	1	0	0	1	**0.8**
8	0	0	0	0	0	**0**
9	0	0	0	0	0	**0**
10	0	0	0	0	0	**0**

Since it has been found in the laboratory that kinesin takes an occasional backstep but doesn't then get stuck[38; 45; 46], these data indicate that kinesin does not use a gated PS stepping mechanism.

3.2.2 Part two - gating hypothesis

As described in section 1.4.5, it is thought that gating at one or more points in the hydrolysis cycle is required to achieve head coordination and prevent premature detachment of kinesin from the MT.

3.2.2.1 Does RBM require a gating mechanism?

The hypothesis proposed here is that entropic linker strain (see section 2.4.3.1) is sufficient to coordinate the heads: that gating is not required for procession to occur in the RBM model. This hypothesis is analysed and further discussed below (section 5.5.3) in terms of the evidence for gating.

In order to test the hypothesis, the simulation was used to assess the effect of varying linker strain on the processivity of the RBM model with and without gating. The gating rule delays ATP hydrolysis by a factor of 10 (see section 2.4.6).

The simulation runs were first performed without gating then repeated with gating. Linker strain is modelled as the probability of head binding in the wait state (section 2.4.3.1) where 0 represents no strain while 10 represents enough strain to prevent binding (in the absence of zippering). The protocol is summarised in the following pseudo-code.

Initialise program:

Select RBM stepping rule

Set counter to number of timing combinations.

For linker_strain = 0 to 10 do

For all timing combinations do

If motor processes to end of MT then decrement counter

Output counter value to file and reset counter.

Repeat above but with ATP gating rule selected.

The results are listed in table 3.4. The percentage of timing combinations for which the motor processed with and without gating (columns headed "success") is calculated by subtracting the average procession failures over 5 runs (columns headed "failed") from the total number of timing combinations. These percentages are plotted in figure 3.4 as a histogram where hollow bars show values for the model without the ATP gate, filled bars show values for the model with the ATP gate.

3.2.2.2 Discussion

Without the gate, all the timing combinations applied to the system resulted in procession at high linker strain (values above 7). Reducing linker strain resulted in an increase in the number of timing combinations failing to give continuous procession. With the gate, only maximum strain resulted in all the timing combinations yielding procession. The gate significantly increased the likelihood of procession at no strain.

These results support the hypothesis. High linker strain is sufficient to coordinate the heads regardless of the timing conditions. If linker strain is key to procession then a positive correlation would be expected with the likelihood of procession and gating would be expected to compensate for lack of strain. Both these predictions are born out by the results.

Table 3.4 Gating comparison: procession data.

Linker strain	*Procession failure - no gate* *1*	*2*	*3*	*4*	*5*	Av. - no gate	**Success % no gate**
0	24	26	26	24	25	25	**7.4**
1	9	9	11	7	13	9.8	**63.7**
2	4	6	8	7	6	6.2	**77.0**
3	1	5	3	3	4	3.2	**88.2**
4	1	2	2	2	1	1.6	**94.1**
5	1	2	0	1	0	0.8	**97.1**
6	0	0	1	1	1	0.6	**97.8**
7	0	0	1	1	0	0.4	**98.5**
8	0	0	0	0	0	0	**100**
9	0	0	0	0	0	0	**100**
10	0	0	0	0	0	0	**100**
Linker strain	*Procession failure - ATP gate* *1*	*2*	*3*	*4*	*5*	Av. - ATP gate	**Success % ATP gate**
0	6	6	7	7	6	6.4	**76.3**
1	6	10	8	11	6	8.2	**69.6**
2	7	6	10	6	6	7	**74.1**
3	8	6	5	7	6	6.4	**76.3**
4	5	4	5	5	5	4.8	**82.2**
5	6	2	4	3	4	3.8	**85.9**
6	3	4	4	2	3	3.2	**88.2**
7	3	1	2	2	3	2.2	**91.9**
8	3	1	0	1	1	1.2	**95.6**
9	0	0	1	0	1	0.4	**98.5**
10	0	0	0	0	0	0	**100**

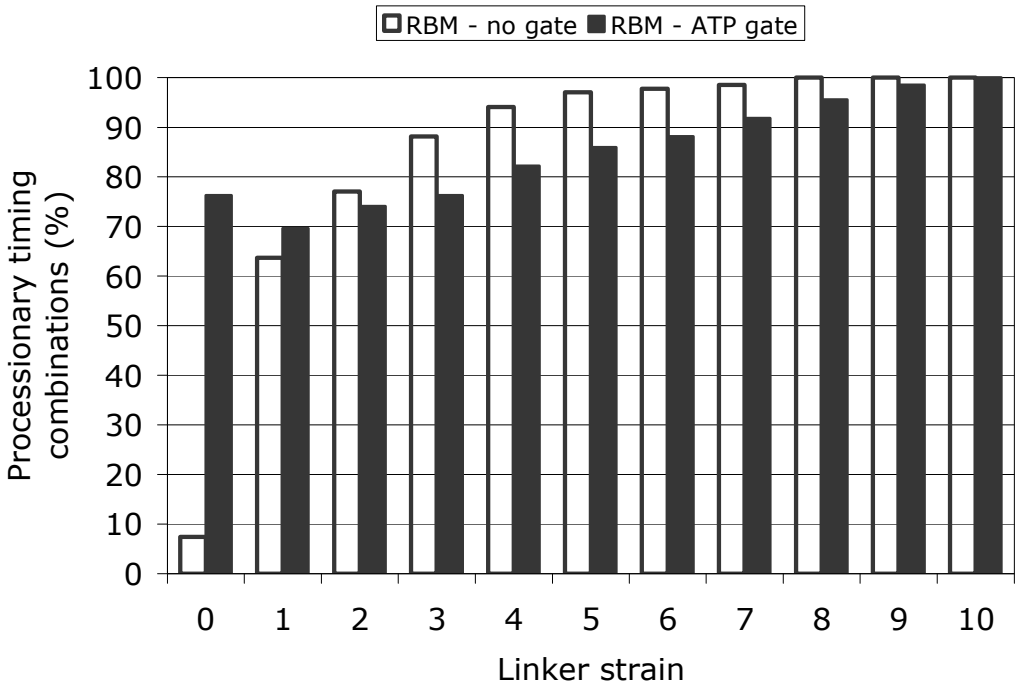

Figure 3.4 Comparison of gated and ungated RBM procession.

3.2.3 Part three - behaviour under load

A hindering load on the motor was implemented as a counterbalance to zippering (see section 2.4.5). The effect of applying increasing load to the motor was measured by counting forward steps, backward steps and detachments for a range of loads. A forward step is a step towards the plus-end of the MT, a backstep is a step towards the minus-end, a detachment is both heads releasing from the MT at the same time. The experiment was repeated under the same conditions except for the addition of the ATP gate. The protocol for the simulation runs is summarised in the following pseudo-code.

Initialise program:

Select RBM stepping rule

Set linker strain to 9.5

Set timing combination to shortest times giving procession.

Zero counters.

For load = 3 to 9 do

Count number of backsteps, forward steps and detachments

Output counter values to file and reset counters.

Repeat above but with ATP gating rule selected.

Appendix A contains the raw data and ratios calculated from the raw data for each run. Tables A.1-3 list the raw data from which the average ratios in table 3.5 have been generated. Each ratio is obtained by dividing the individual average value (either forward steps, backward steps or detachments) by the sum of the average values (forward steps + backward steps + detachments). The ratio or fraction of forward (green circles), backward (blue triangles) steps and detachments (orange squares) to the total stepping is plotted against load in figure 3.5 without gating and in figure 3.6 with gating.

Table 3.5 Gating comparison: average stepping ratios.

	No gate			ATP gate		
Load	**Forward**	**Backward**	**Detach**	**Forward**	**Backward**	**Detach**
3	0.94	0.06	0.00	0.97	0.03	0.00
3.5	0.96	0.04	0.00	0.96	0.04	0.00
4	0.96	0.04	0.00	0.96	0.04	0.00
4.5	0.95	0.05	0.00	0.98	0.02	0.00
5	0.94	0.03	0.02	0.92	0.08	0.00
5.5	0.86	0.06	0.08	0.94	0.06	0.00
6	0.78	0.06	0.16	0.89	0.11	0.00
6.5	0.67	0.07	0.26	0.49	0.46	0.05
7	0.57	0.09	0.34	0.48	0.47	0.05
7.5	0.35	0.34	0.32	0.46	0.49	0.05
8	0.33	0.37	0.30	0.54	0.42	0.04
8.5	0.35	0.34	0.32	0.47	0.50	0.04
9	0.36	0.30	0.34	0.48	0.48	0.04

The variation across runs in some of the raw load data appears large. To confirm that 5 runs per set of parameters is sufficient to ascertain system

behaviour, the published plots have been augmented here with maximum and minimum values for each data point. The maximum values for forward steps (Fmax), backward steps (Bmax), and detachments (Dmax) are plotted as dashes linked by a continuous line. The minimum values for forward steps (Fmin), backward steps (Bmin), and detachments (Dmin) are plotted as dashes linked with a dotted line. Table 3.6 lists these data points for the ungated and gated model; these are derived from the raw data listed in appendix A in tables A.4 and A.5 respectively.

Table 3.6 Gating comparison: minimum and maximum stepping ratios.

	No gate					
Load	***Bmin***	***Bmax***	***Fmin***	***Fmax***	***Dmin***	***Dmax***
3	0.02	0.15	0.85	0.98	0.00	0.00
3.5	0.02	0.08	0.92	0.98	0.00	0.00
4	0.00	0.11	0.89	1.00	0.00	0.00
4.5	0.02	0.07	0.93	0.98	0.00	0.00
5	0.02	0.05	0.92	0.97	0.00	0.05
5.5	0.01	0.14	0.78	0.92	0.05	0.10
6	0.04	0.10	0.70	0.85	0.11	0.19
6.5	0.03	0.10	0.63	0.76	0.20	0.28
7	0.05	0.14	0.53	0.64	0.31	0.38
7.5	0.31	0.35	0.33	0.36	0.30	0.33
8	0.36	0.42	0.30	0.35	0.28	0.31
8.5	0.26	0.40	0.31	0.39	0.30	0.35
9	0.16	0.39	0.31	0.43	0.29	0.41

	ATP gate					
Load	***Bmin***	***Bmax***	***Fmin***	***Fmax***	***Dmin***	***Dmax***
3	0.00	0.04	0.96	1.00	0.00	0.00
3.5	0.00	0.06	0.94	1.00	0.00	0.00
4	0.00	0.06	0.94	1.00	0.00	0.00
4.5	0.00	0.06	0.94	1.00	0.00	0.00
5	0.02	0.16	0.84	0.98	0.00	0.00
5.5	0.02	0.09	0.91	0.98	0.00	0.00
6	0.08	0.16	0.84	0.92	0.00	0.00
6.5	0.36	0.52	0.41	0.61	0.00	0.07
7	0.39	0.50	0.44	0.53	0.02	0.08
7.5	0.45	0.53	0.43	0.52	0.02	0.09
8	0.31	0.49	0.46	0.65	0.00	0.06
8.5	0.44	0.57	0.39	0.53	0.03	0.05
9	0.36	0.56	0.39	0.63	0.01	0.06

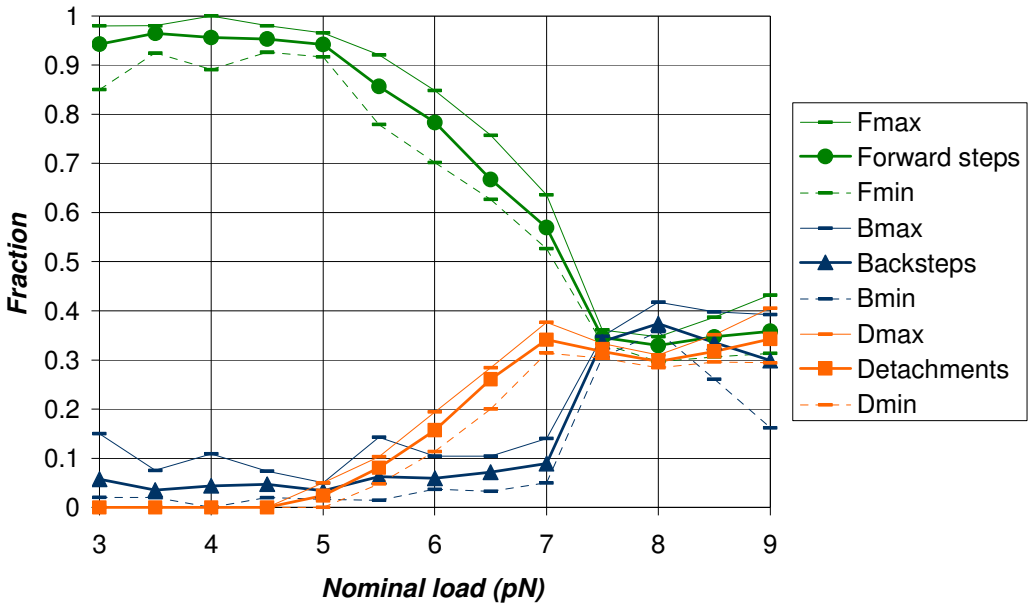

Figure 3.5 Load characteristics of RBM stepping without gating.

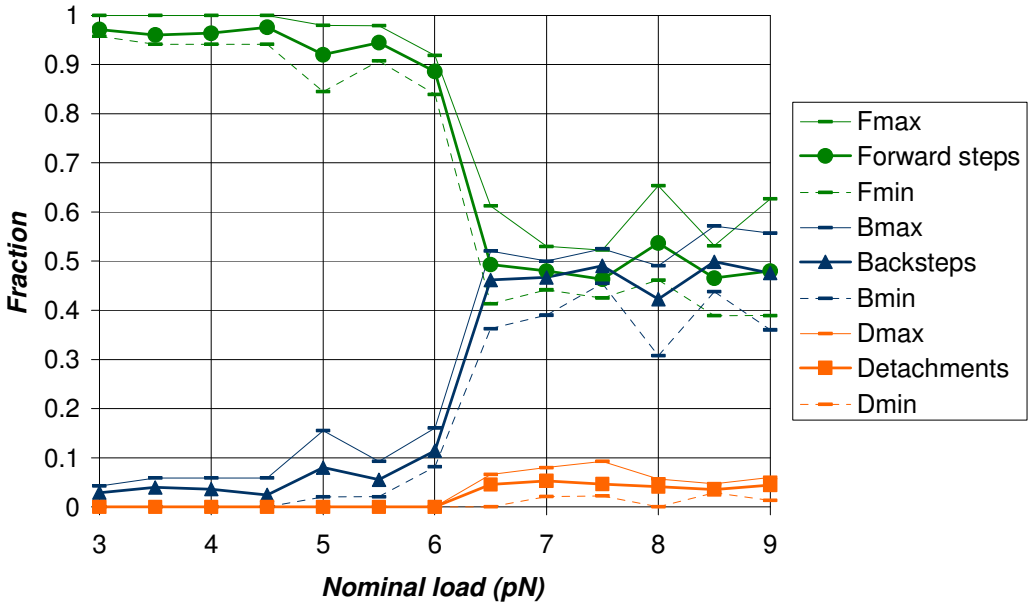

Figure 3.6 Load characteristics of RBM stepping with gating.

3.2.3.1 Discussion

Without the gate (figure 3.5), there is a progressive reduction of forward steps and a rise in detachments above 5 pN and then in backsteps at 7 pN

until equalisation at a load of 7.5 pN, a stall force that is similar to that measured by Carter and Cross (2005).[37] The ATP gate has the effect of aligning and accelerating the change in forward and backward steps which become equal in number at a stall force of 6.5 pN (figure 3.6). The gate almost eliminates detachments.

Gating has a stabilising effect on the motor but reduces its power: the gated motor is less likely to detach but stalls at a lower load than the ungated motor. Figure 3.7 reproduces figure 4a in Nishiyama et al. (2002)[106] which shows the experimental results at two ATP concentrations where the forward step ratio is plotted as circles, backward ratio as triangles and detachment ratio as squares. Comparing these experimental results with those obtained from the simulation, the ungated results match noticeably better than the gated results thus the simulation data favour the ungated model as more realistic.

The mismatch between the simulation results and the experimental results is pronounced at super-stall loads. The experimental curves trend downward for stepping and upward for detachment at super-stall load whereas the simulation curves converge and level off at these loads. The discrepancy may be accounted for by under-recording of detachments by the simulation. In a simulation run, the motor is allowed to re-engage the MT if it detaches so a detachment may not be counted. In the experiment, a run stops if the motor detaches so the detachment is counted. This particularly affects the super-stall data as the number of detachments increases with load. The later version of the program follows the experimental method resulting in a better data match (see section 4.1.2).

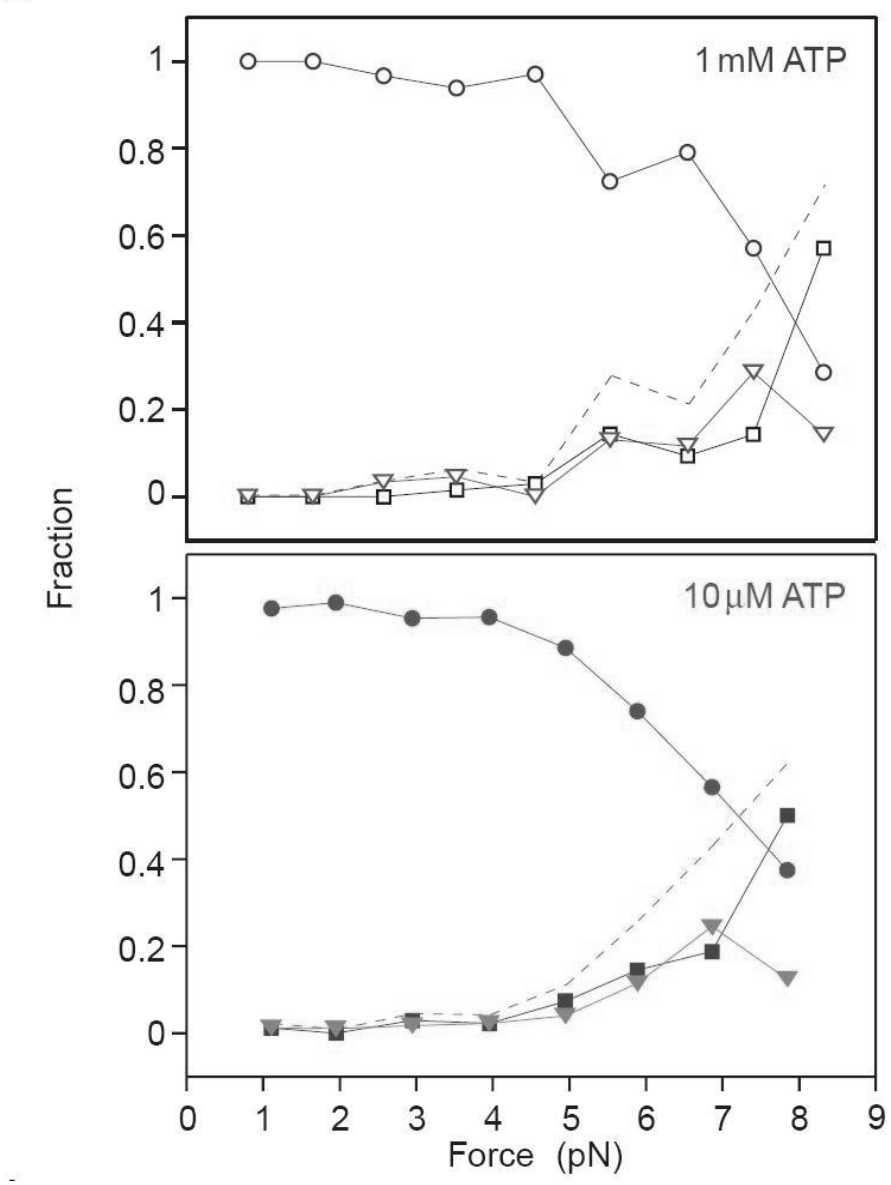

Figure 3.7 Nishiyama et al. (2002)[106] figure 4a plotting the fraction of forward steps (circles), backward steps (triangles) and detachments (squares) for [ATP] of 1 mM (upper plot) and 10 µM (lower plot). The dashed lines sum the backward steps and detachments.

3.2.4 Part four - blockage behaviour

A long-term goal of studying kinesin is to discover more about axonal transport dysfunction since this is implicated in neurodegenerative disease

such as Alzheimer's as discussed in section 1.1.4. An initial step towards this goal is the investigation of the effect of blocking kinesin's path. There is conflicting evidence as to how long kinesin waits before detaching from the MT at an obstruction (section 1.3.6). Here a blockage was placed on the MT for a varying amount of time to determine the waiting period. The simulation runs for part four are summarised in the following pseudo-code.

Initialise program:

Select RBM stepping rule

Set linker strain to 9.5

Set timing combination to shortest times giving procession

Set load to 0

Make successive runs increasing the duration of the blockage.

Repeat above but with ATP gating rule selected.

Without gating, the motor was observed to detach within one hydrolysis cycle. The detachment was delayed tenfold with gating applied.

3.2.4.1 Discussion

The gated result is consistent with the findings of Seitz and Surrey (2006)[40] while the ungated result is consistent with those of Crevel et al. (2004)[47]. Detachment happens because the free head moves forward but is unable to bind the next MT binding site and so is free when its partner head finishes its hydrolysis cycle and detaches. If the blockage is removed before the bound head detaches the free head binds the MT and procession resumes. Gating hydrolysis increases the waiting time since the motor cannot detach before hydrolysis has completed.

Chapter 4 Results – random ATP arrival

The experiments described in the previous chapter used a simplistic model of ATP arrival. In this chapter, the effects of a more realistic model employing a Poisson distribution are described. The behaviour of power stroke and rectified Brownian motion models over a range of loads and ATP concentrations is compared against experimental data. A multiple motor system is also investigated. The main conclusion is that the gated RBM model fits the data best.

4.1 Single motor investigation

The main program used for the virtual experiments described in this section is listed in appendix B.2, supporting functions are listed in appendices B.4-7. In the previous experiments of this study, ATP binding event timings were fixed throughout a run. The arrival times of an ATP molecule, which is a diffusive process, are more realistically modelled here by generating them from a nominal value according to a Poisson distribution using Knuth's algorithm[104] to simulate random arrival of the molecule. The rest of the event timings were increased from the fixed values applied in previous experiments in order to allow for relatively smaller ATP binding timings i.e. the simulation of high [ATP]. The values approximate the estimates of Rosenfeld et al. (2002)[80] in milliseconds: ADP release (KD → K0) = 7, ATP hydrolysis (KDP → KDu) = 8, phosphate release (KDP → KDu) = 13. The number of runs per set of parameters was increased to 100 in order to accommodate the variation introduced by using the Poisson mechanism.

4.1.1 Velocity and dwell times at no load

The simulation was run for each stepping mechanism in turn at zero load for a range of ATP concentrations. The number of steps and time taken were recorded for each run then their average calculated over each set of runs performed at the same nominal ATP arrival time. Velocity was calculated by subtracting the number of backsteps from the number of forward steps with the result divided by the time taken. Dwell time was calculated by dividing the run duration by the number of steps taken during that time. Both sets of values were scaled to compare with experiment.

4.1.1.1 Simulation results

The results averaged over 100 runs are listed in table 4.1. Figure 4.1 plots the velocity data and figure 4.2 plots the dwell times for each stepping mechanism against ATP concentration which is plotted as nominal arrival time of the molecule. Results for gated RBM and gated PS are labelled RBMg and PSg respectively.

Table 4.1 Velocity and dwell time comparison under no load.

	Velocity			
ATP arrival time	***PS***	***RBM***	***PSg***	***RBMg***
1	215.81	800.00	480.74	768.74
2	178.60	775.44	457.67	736.74
4	183.07	715.91	482.98	689.12
8	101.21	620.65	472.56	610.23
16	148.84	453.21	444.28	443.53
32	107.16	192.00	260.47	258.23
64	40.93	70.70	156.28	160.74
	Dwell time			
	PS	***RBM***	***PSg***	***RBMg***
1	1.85	1.06	1.21	0.83
2	2.12	1.21	1.20	0.87
4	1.95	1.64	1.21	0.92
8	2.40	1.99	1.23	1.04
16	2.12	2.58	1.39	1.43
32	2.83	3.69	2.47	2.44
64	4.91	6.00	4.02	3.87

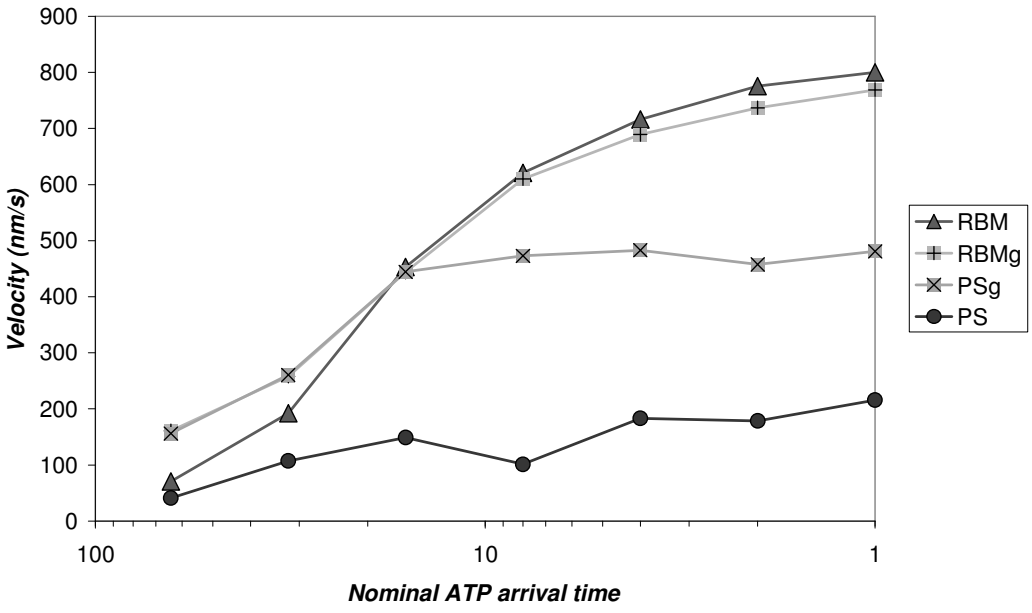

Figure 4.1 Relationship between velocity and [ATP].

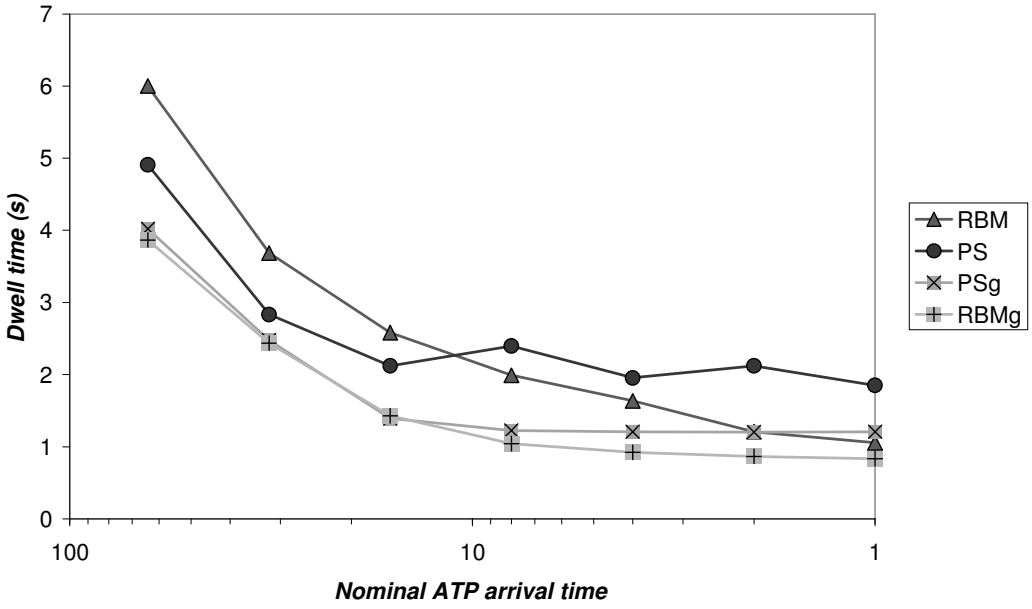

Figure 4.2 Relationship between dwell time and [ATP].

4.1.1.2 Experimental results

Figure 4.3 is taken from an *in vitro* fluorescence study of kinesin at varying [ATP] by Yajima et al. (2002).[107] A similar velocity curve was obtained by bead assay in a study by Rosenfeld et al. (2003).[59]

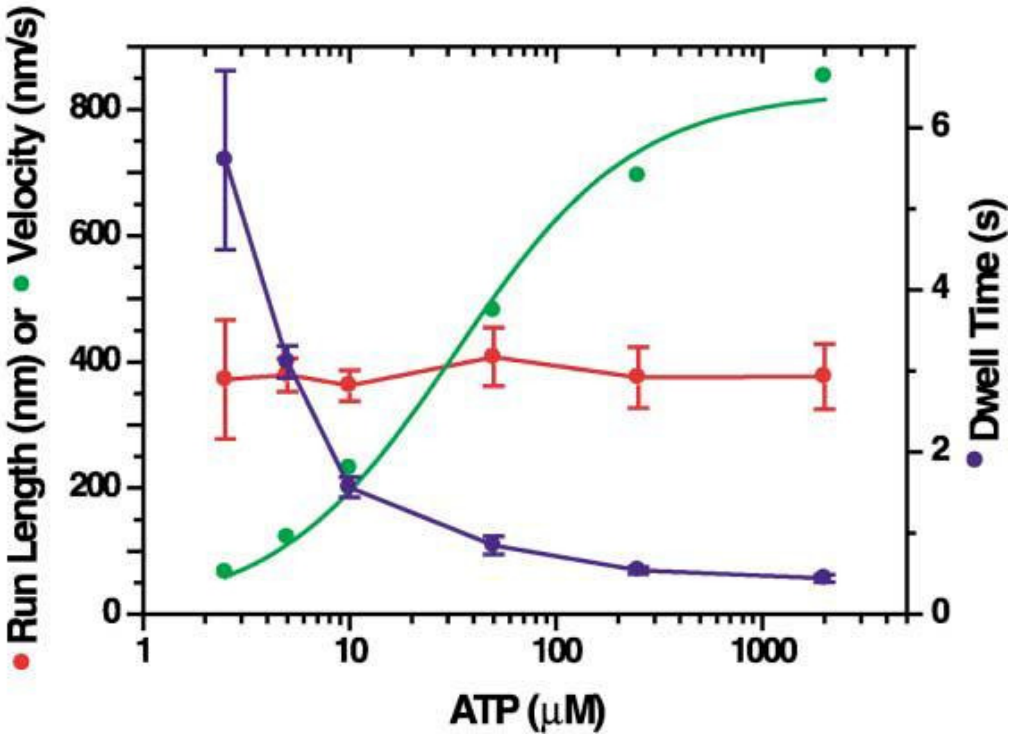

Figure 4.3 Yajima et al. (2002)[107] figure 2c: plot of run length (central, in red), dwell time (top left to bottom right in blue) and velocity (bottom left to top right in green).

4.1.1.3 Discussion

The simulation results confirm those of the previous chapter in supporting the hypothesis that gating is not required for RBM procession and favouring the RBM model. By visual inspection, the ungated RBM model provides as good a fit to both the velocity and dwell time experimental data as the gated RBM model. The velocity data of both PS models show striking variance with experiment.

4.1.2 Load

The simulation was run for each stepping mechanism in turn for a range of ATP arrival times but also for a range of loads. As above, the number of steps

and time taken were recorded for each run then their average calculated over each set of runs performed with the same nominal ATP arrival time. The number of runs in which the motor detached from the track were also recorded. Step ratios (or fractions) were then calculated in the same manner as section 3.2.3: the steps, backsteps and detachments were summed and each result in turn divided by this sum to give a data point.

The results for the ungated PS model are not recorded here as the data proved radically different from experiment, having over 90% detachments regardless of load.

4.1.2.1 High [ATP]

The load results for nominal ATP arrival time of 1 illustrate the motor behaviour at high [ATP] as this is significantly quicker than the hydrolysis cycle time of 28 (the sum of the fixed timings). The simulation results are listed in tables 4.2-4 and plotted in figure 4.4, figure 4.5, and figure 4.6 for ungated RBM, gated RBM and gated PS models respectively.

Table 4.2 High [ATP] ungated RBM step ratios.

Load	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
				Ratios					
Step	0.990	0.987	0.989	0.979	0.979	0.912	0.943	0.356	0.000
Backstep	0.010	0.013	0.011	0.021	0.021	0.016	0.027	0.051	0.019
Detachment	0.000	0.000	0.000	0.000	0.000	0.071	0.030	0.593	0.981
Backstep + Detachment	0.010	0.013	0.011	0.021	0.021	0.088	0.057	0.644	1.000
				Raw data					
Steps	94.4	95.6	96.1	94.1	94.1	89.5	93.8	12.6	0
Backsteps	1	1.3	1.1	2	2	1.6	2.7	1.8	1.9
Detachments	0	0	0	0	0	7	3	21	100

Table 4.3 High [ATP] gated RBM step ratios.

Load	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
				Ratios					
Step	0.975	0.968	0.961	0.963	0.958	0.952	0.952	0.943	0.000
Backstep	0.025	0.032	0.039	0.037	0.042	0.048	0.048	0.034	0.016
Detachment	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.023	0.984
Backstep + Detachment	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.023	0.984
				Raw data					
Steps	98.5	99.6	100.4	99.9	100.7	101.2	98.2	82.2	0
Backsteps	2.5	3.3	4.1	3.8	4.4	5.1	4.9	3	1.6
Detachments	0	0	0	0	0	0	0	2	100

Table 4.4 High [ATP] gated PS step ratios.

Load	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
				Ratios					
Step	0.836	0.712	0.895	0.654	0.561	0.178	0.018	0.003	0.000
Backstep	0.009	0.018	0.024	0.031	0.025	0.044	0.023	0.022	0.023
Detachment	0.154	0.270	0.081	0.315	0.414	0.779	0.959	0.975	0.977
Backstep + Detachment	0.164	0.288	0.105	0.346	0.439	0.822	0.982	0.997	1.000
				Raw data					
Steps	27.1	23.7	22.1	18.7	17.6	5.7	1.3	0.3	0
Backsteps	0.3	0.6	0.6	0.9	0.8	1.4	1.7	2.2	2.4
Detachments	5	9	2	9	13	25	70	97	100

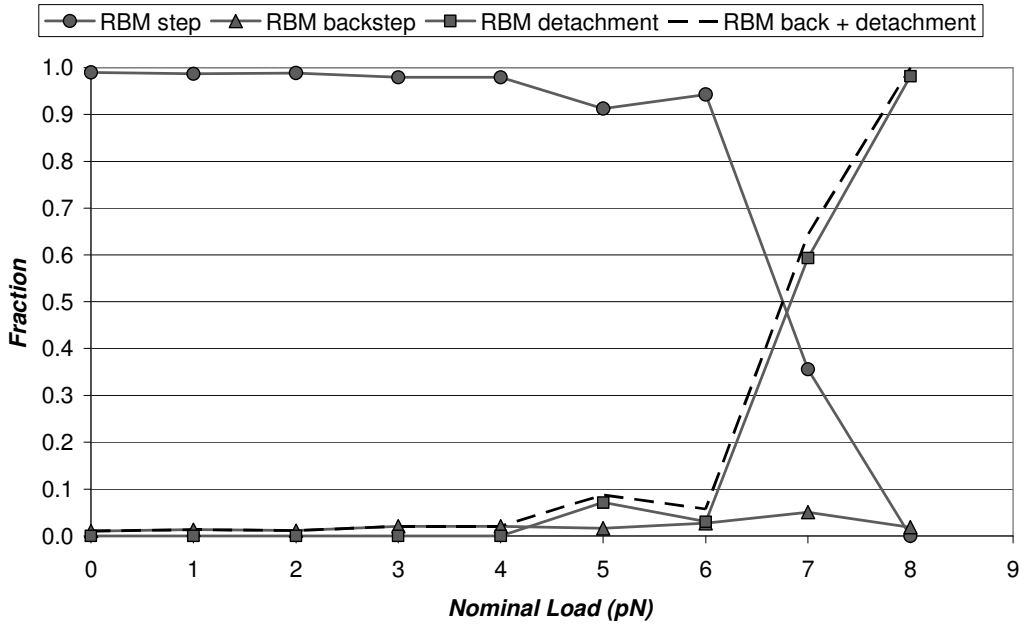

Figure 4.4 High [ATP] ungated RBM step ratio to load relationship.

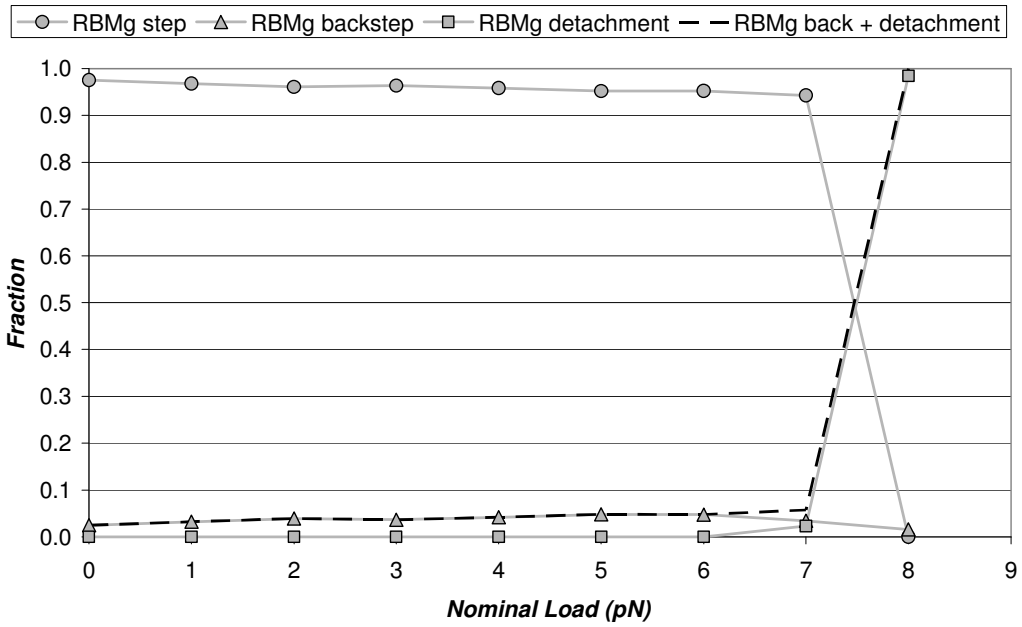

Figure 4.5 High [ATP] gated RBM step ratio to load relationship.

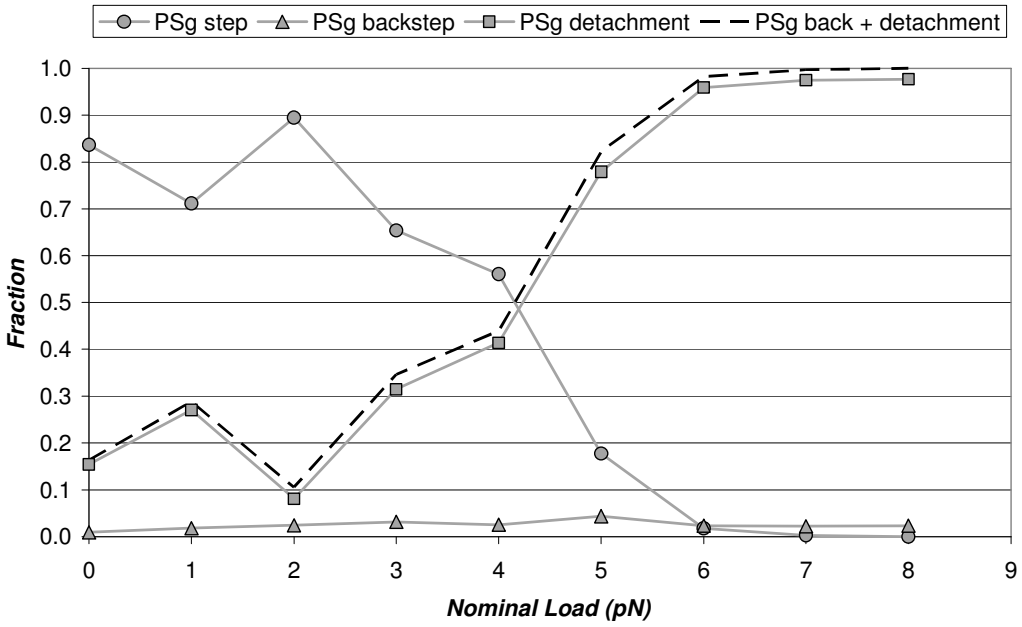

Figure 4.6 High [ATP] gated PS step ratio to load relationship.

4.1.2.2 Medium [ATP]

The load results for nominal ATP arrival time of 16 illustrate the motor behaviour at medium [ATP] giving a range of arrival times comparable to the cycle time. The simulation results are listed in tables 4.5-7 and plotted in figure 4.7, figure 4.8, and figure 4.9 for ungated RBM, gated RBM and gated PS models respectively.

Table 4.5 Mid [ATP] ungated RBM step ratios.

Load	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***
Ratios									
Step	0.971	0.943	0.951	0.945	0.928	0.869	0.807	0.400	0.000
Backstep	0.029	0.057	0.049	0.055	0.072	0.057	0.080	0.186	0.022
Detachment	0.000	0.000	0.000	0.000	0.000	0.074	0.114	0.414	0.978
Backstep + Detachment	0.029	0.057	0.049	0.055	0.072	0.131	0.193	0.600	1.000
Raw data									
Steps	98.6	98.9	98.7	99.9	98.9	93.5	92.2	27.1	0
Backsteps	2.9	6	5.1	5.8	7.7	6.1	9.1	12.6	2.2
Detachments	0	0	0	0	0	8	13	28	100

Table 4.6 Mid [ATP] gated RBM step ratios.

Load	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
				Ratios					
Step	0.963	0.961	0.961	0.950	0.956	0.936	0.943	0.859	0.000
Backstep	0.037	0.039	0.039	0.050	0.044	0.064	0.057	0.046	0.022
Detachment	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.096	0.978
Backstep + Detachment	0.037	0.039	0.039	0.050	0.044	0.064	0.057	0.141	1.000
				Raw data					
Steps	100	100.5	100.3	101.4	101.1	101.2	96.9	35.8	0
Backsteps	3.8	4.1	4.1	5.3	4.7	6.9	5.9	1.9	2.3
Detachments	0	0	0	0	0	0	0	4	100

Table 4.7 Mid [ATP] gated PS step ratios.

Load	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
				Ratios					
Step	0.942	0.963	0.951	0.916	0.891	0.392	0.038	0.005	0.000
Backstep	0.022	0.025	0.034	0.040	0.050	0.067	0.048	0.032	0.036
Detachment	0.036	0.013	0.015	0.044	0.059	0.541	0.914	0.963	0.964
Backstep + Detachment	0.058	0.037	0.049	0.084	0.109	0.608	0.962	0.995	1.000
				Raw data					
Steps	77.4	74.6	64.3	61.9	60.6	12.3	2.1	0.5	0
Backsteps	1.8	1.9	2.3	2.7	3.4	2.1	2.7	3	3.7
Detachments	3	1	1	3	4	17	51	90	100

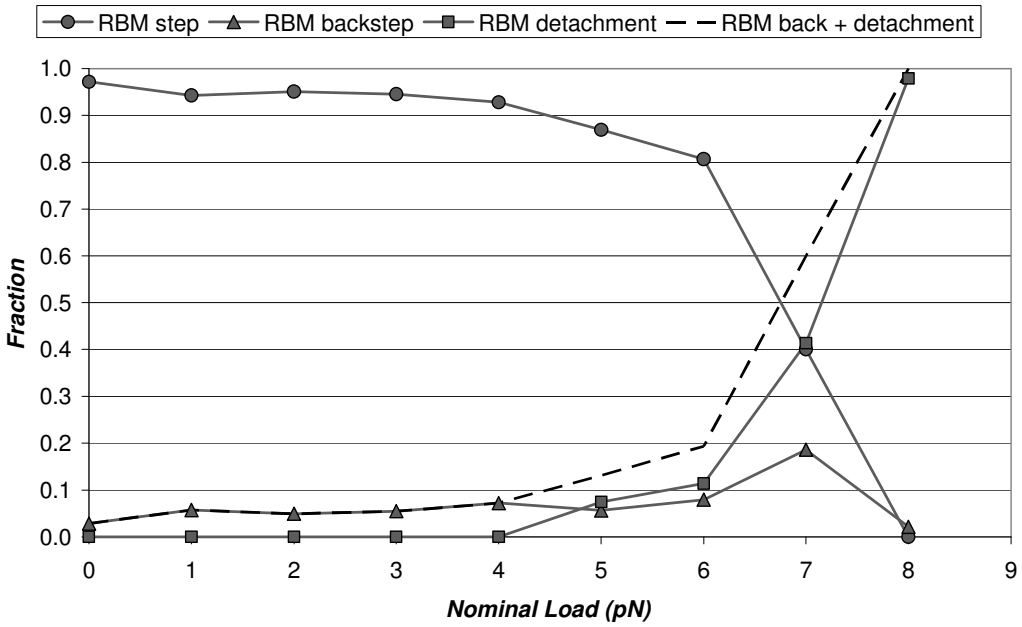

Figure 4.7 Mid [ATP] ungated RBM step ratio to load relationship.

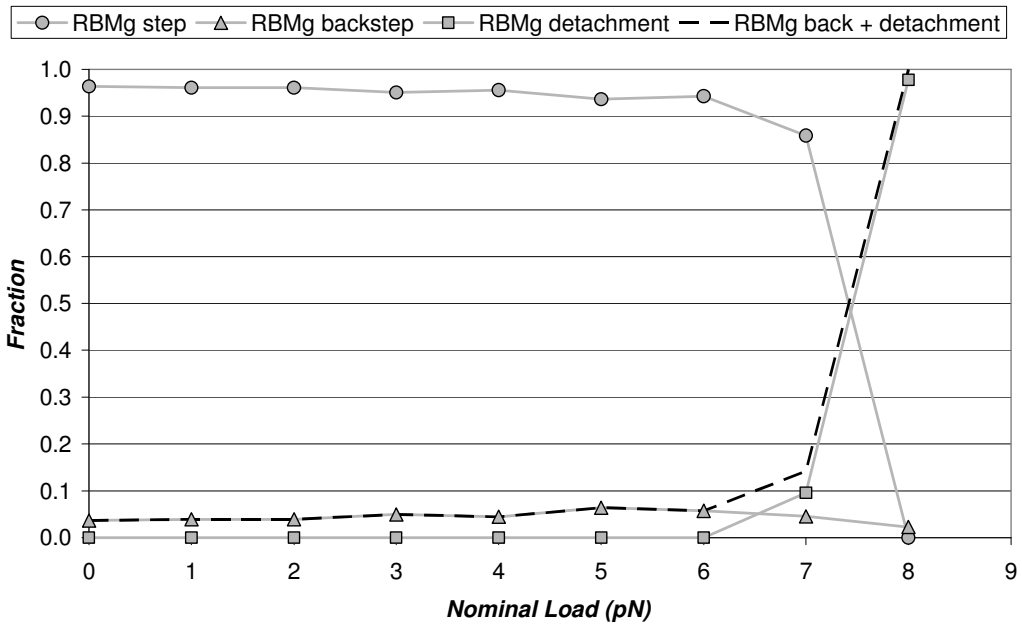

Figure 4.8 Mid [ATP] gated RBM step ratio to load relationship.

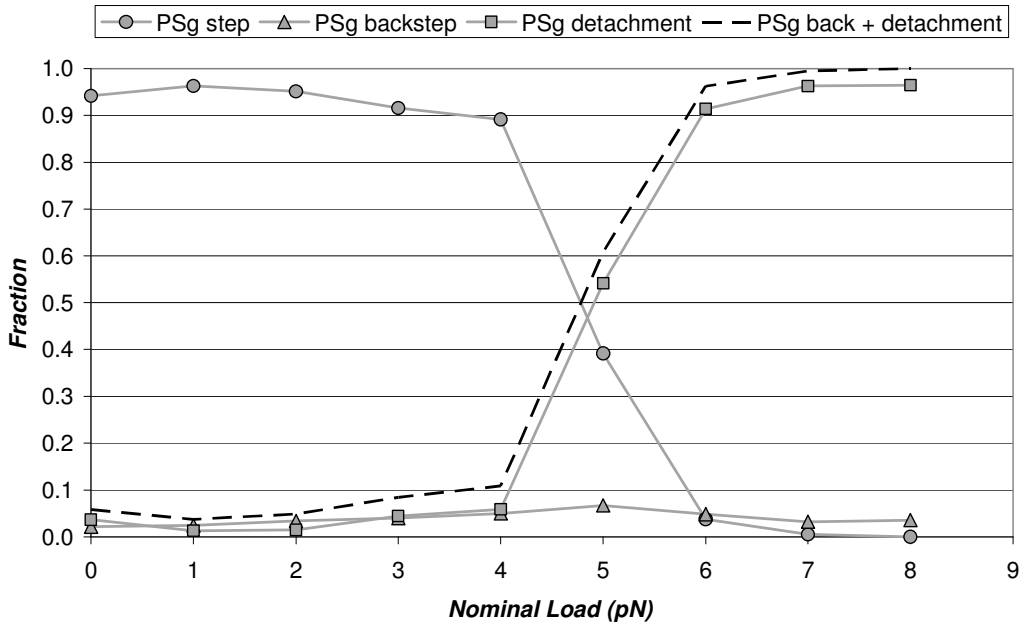

Figure 4.9 Mid [ATP] gated PS step ratio to load relationship.

4.1.2.3 Low [ATP]

The load results for nominal ATP arrival time of 64 illustrate the motor behaviour at low [ATP] as the arrival time is greater than twice the cycle time. The simulation results are listed in tables 4.8-10 and plotted in figure 4.10, figure 4.11, and figure 4.12 for ungated RBM, gated RBM and gated PS models respectively.

Table 4.8 Low [ATP] ungated RBM step ratios.

Load	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***
Ratios									
Step	0.761	0.664	0.549	0.485	0.484	0.479	0.477	0.104	0.000
Backstep	0.239	0.336	0.451	0.515	0.516	0.514	0.513	0.350	0.040
Detachment	0.000	0.000	0.000	0.000	0.000	0.007	0.010	0.547	0.960
Backstep + Detachment	0.239	0.336	0.451	0.515	0.516	0.521	0.523	0.896	1.000
Raw data									
Steps	137.6	186	382.3	392	382.7	392.6	382.9	9.1	0
Backsteps	43.2	94.1	314.2	416.4	408	420.8	411.1	30.7	4.2
Detachments	0	0	0	0	0	6	8	48	100

Table 4.9 Low [ATP] gated RBM step ratios.

Load	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
					Ratios				
Step	0.960	0.947	0.942	0.935	0.906	0.894	0.841	0.728	0.000
Backstep	0.040	0.053	0.058	0.065	0.094	0.106	0.159	0.213	0.046
Detachment	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.058	0.954
Backstep + Detachment	0.040	0.053	0.058	0.065	0.094	0.106	0.159	0.272	1.000
					Raw data				
Steps	96.8	100.3	100.3	99.2	104.7	106.4	111.9	49.8	0
Backsteps	4	5.6	6.2	6.9	10.8	12.6	21.1	14.6	4.8
Detachments	0	0	0	0	0	0	0	4	100

Table 4.10 Low [ATP] gated PS step ratios.

Load	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
					Ratios				
Step	0.975	0.956	0.939	0.932	0.911	0.568	0.095	0.005	0.000
Backstep	0.025	0.044	0.061	0.068	0.089	0.137	0.114	0.030	0.036
Detachment	0.000	0.000	0.000	0.000	0.000	0.295	0.790	0.964	0.964
Backstep + Detachment	0.025	0.044	0.061	0.068	0.089	0.432	0.905	0.995	1.000
					Raw data				
Steps	98.6	99.6	101.6	102.3	104.9	15.4	3.5	0.5	0
Backsteps	2.5	4.6	6.6	7.5	10.2	3.7	4.2	2.8	3.7
Detachments	0	0	0	0	0	8	29	89	100

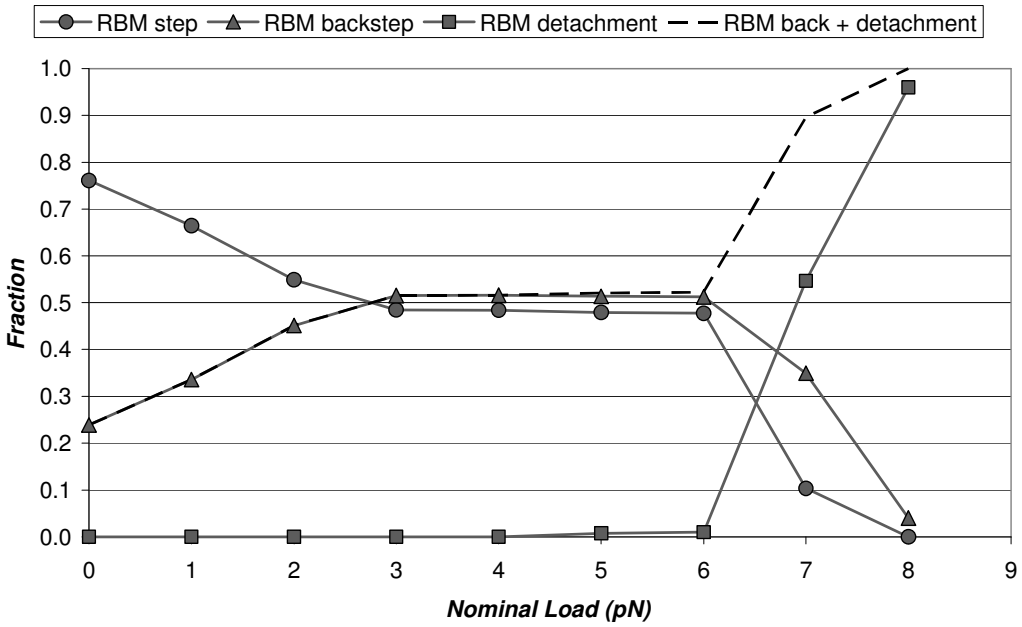

Figure 4.10 Low [ATP] ungated RBM step ratio to load relationship.

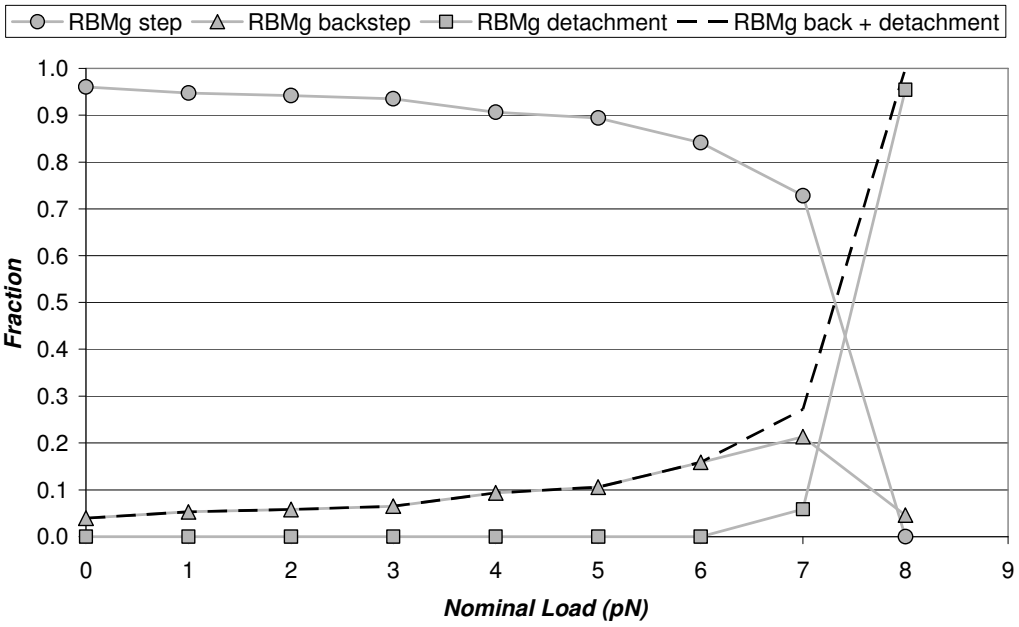

Figure 4.11 Low [ATP] gated RBM step ratio to load relationship.

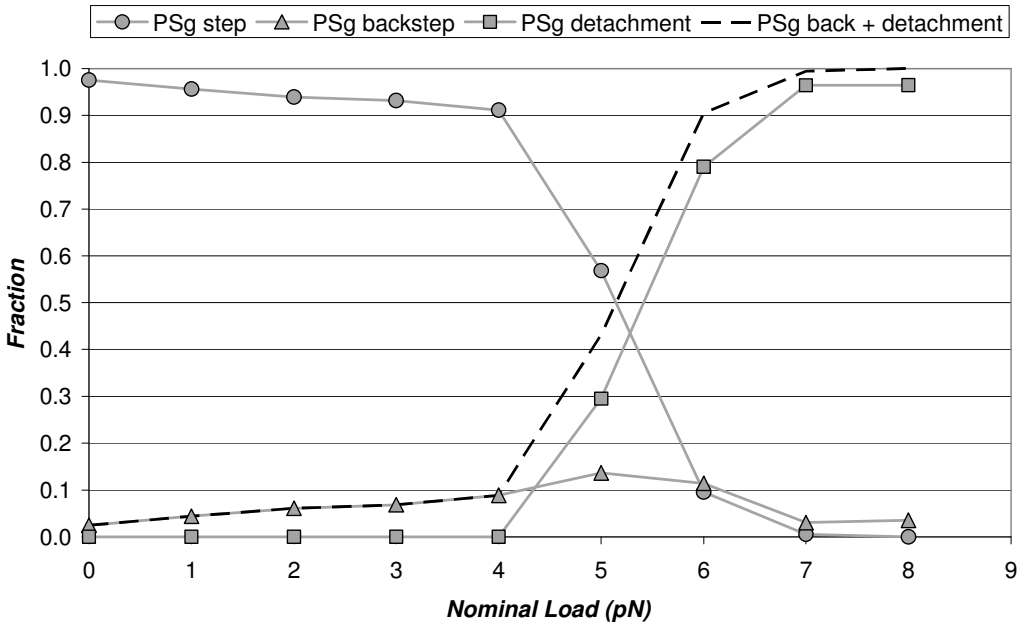

Figure 4.12 Low [ATP] gated PS step ratio to load relationship.

4.1.2.4 Discussion

These results are broadly the same as those of the previous chapter but there are some differences. These differences occur because the earlier version of the program allowed the motor to re-engage the MT after detaching whereas this version of the program ends the run if the motor detaches. The revised program has produced results that are more realistic at high load as the modified program now follows the experimental method. Applying a range of ATP concentrations has revealed unexpected behaviour of the RBM model.

Visual comparison of these results with experiment data (figure 3.7) show general similarities, the best matches being ungated RBM for high and medium [ATP] and gated RBM for low[ATP], the exceptions being the high [ATP] gated PS plot (Figure 4.6) and the ungated RBM model at low [ATP] (figure 4.10). The most surprising result is the behaviour of the ungated RBM model at low [ATP], radically different from that displayed at higher

concentration and completely different from experiment. This is primarily due to an increase in backstepping as ATP concentration is reduced. The backstep is normally a rare occurrence but, as the time interval between ATP arrivals increases, the motor lingers longer in the wait state when the likelihood of a backstep increases. Gating reduces this effect: the gated RBM plot (figure 4.11) shows marked similarity to the experimental data (figure 3.7, lower plot labelled 10 μM ATP). This is because gating increases the time the motor spends hydrolysing ATP (see section 2.4.6) and so it spends less time in the wait state.

The most important feature of the motor under load is the stall force. Table 4.11 compares the stall force of the models to experiment at varying [ATP]. The values shown are estimated from the respective plots to the nearest 0.5 pN as the intersection between the dashed line (backsteps and detachments) with the step data line. Gated RBM closely tracks the experimental values, unlike the other models.

Table 4.11 Stall force comparison.

pN	Experiment	RBM	RBMg	PSg
High [ATP]	7.5	7	7.5	4
Mid [ATP]	-	7	7.5	4.5
Low [ATP]	7	3	7	5

The conclusion is that the gated RBM model behaves the most realistically under the full range of ATP concentration and load conditions.

4.2 Two motor investigation

The power house of the cell, the mitochondrion, is transported along the axon by several motors, as noted in section 1.2. With a view to future modelling of mitochondrial transport, this section reports an initial investigation into the behaviour of two linked motors.

4.2.1 Multimotor simulation

The main program used for the multimotor experiments is listed in appendix B.3, supporting functions are listed in appendices B.4-7. This version of the program includes a linkage between the motors (see section 2.4.8). The run length calculation was changed from that used in the single motor investigation to be the result of subtracting the mid-point between the motors at the finish and start of the run. In the single motor system, the run terminates if the motor detaches so the run length is directly related to the stepping count. In the two motor system the run does not terminate if one motor detaches when its companion continues to process so the stepping count is not a reliable basis for calculating run length.

4.2.2 Simulation results

The run length and time taken were recorded for 100 simulation runs for each stepping model applied to a single motor and then to two linked motors at high [ATP]. The velocity was derived by dividing the run length by the time taken. The average velocities and run lengths are listed in table 4.12. These data are plotted in figure 4.13 and figure 4.14 respectively. Gated RBM and gated PS data are labelled RBMg and PSg respectively.

Table 4.12 Multimotor results.

	Velocity			
	RBM	***RBMg***	***PS***	***PSg***
1 motor	597.89	798.04	183.18	505.66
2 motors	623.77	753.27	96.45	458.70
	Run length			
	RBM	***RBMg***	***PS***	***PSg***
1 motor	1.91	2.01	0.01	0.49
2 motors	1.86	1.89	0.44	1.87

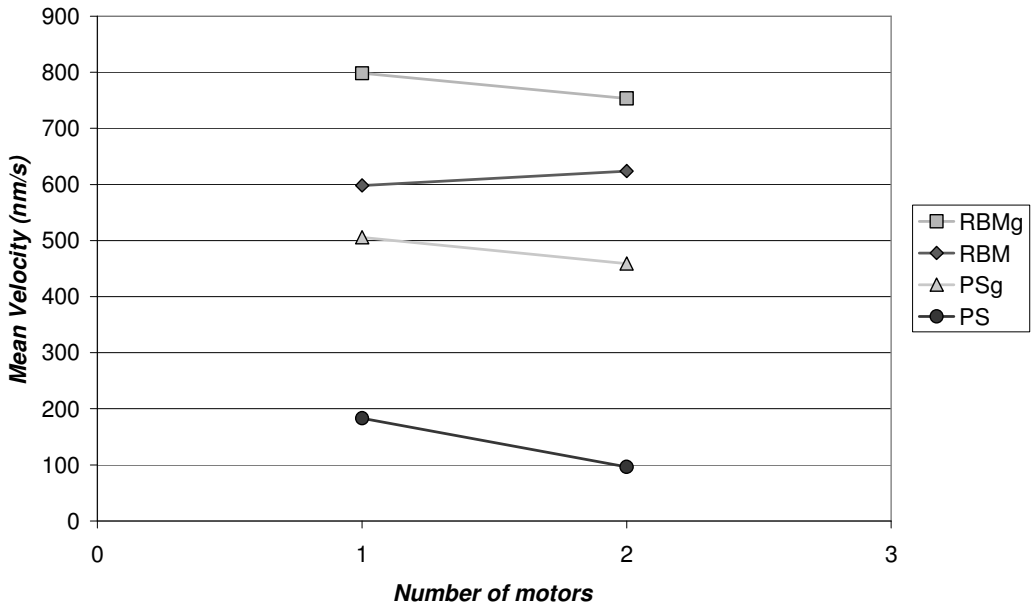

Figure 4.13 Simulated multiple motor comparison of velocity.

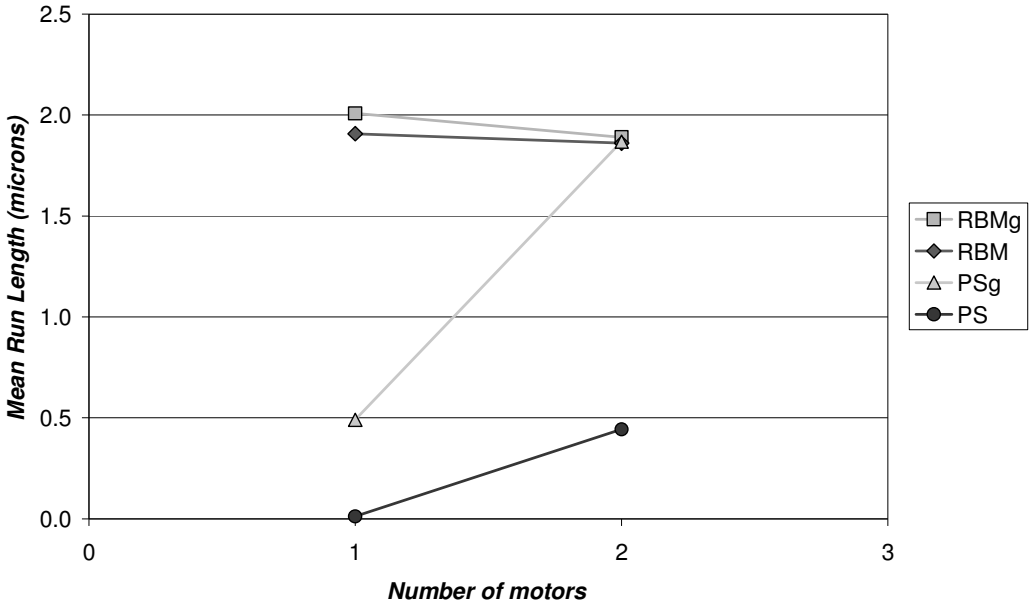

Figure 4.14 Simulated multiple motor comparison of run length.

4.2.3 Experimental results

Bead assays conducted by Seitz & Surrey (2006) and Beeg *et al.* (2008) at saturation [ATP] (1-5 mM) show that the velocity of the bead is hardly affected by the number of motors bound to it (figure 4.15) whereas the run length increases with the number of motors (figure 4.16).[40; 108]

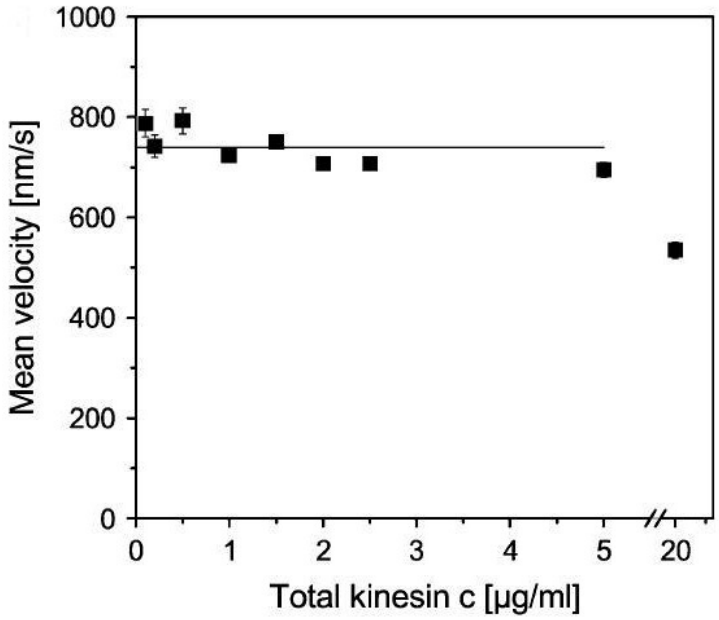

Figure 4.15 Multiple motor comparison of velocity (Beeg *et al.* 2008[108] figure 2b).

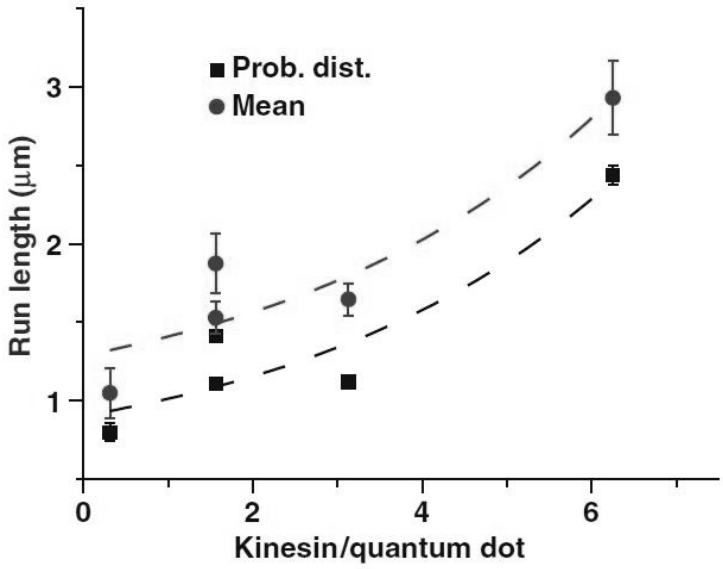

Figure 4.16 Multiple motor comparison of run length (Seitz and Surrey 2006[40] figure 1d).

4.2.4 Discussion

Only gated RBM stepping velocity shows a good fit to experiment (Figure 4.15); the worst fit is the ungated PS which not only runs at a quarter of the assay velocity as a single motor but adding a motor halves the velocity again. The latter effect occurred because progress was slowed by detachments whereas for the single motor the run terminated on detachment. This happened to a much lesser extent with the gated motors though not with ungated RBM where no detachment was observed: linking ungated RBM motors improves individual motor head coordination.

The run length data are not a good fit for experimental values (figure 4.16). Gated PS stepping showed the same upward trend but the increase in run length for the two motor case (almost fourfold) is much higher than the fitted experimental curves would indicate (about 10%). In contrast, for most of the RBM runs, both the single and double motors processed the full length of the MT, indicating a very low probability of detachment. This behaviour is not consistent with the bead assay though it is consistent with gliding assays (see section 1.3.3). The convergence of the twin motor run length results for all but ungated PS is intriguing and may indicate a general property of multiple motor behaviour.

Chapter 5 Evaluation

This chapter comprises an evaluation of the study and the stepping model. A coherent argument is made for a new model of kinesin stepping with due consideration of challenging data. The plausibility of the ungated RBM mechanism for kinesin stepping was established in the first part of this study (chapter 3). Further investigation has shown the requirement for gating under conditions of low ATP concentration (chapter 4). The model is discussed here in relation to laboratory evidence and is shown to be capable of explaining a wide range of recent experimental results. Laboratory experiments are suggested based on the model's predictions.

5.1 The study

The biological goal of this study was to improve our understanding of how kinesin walks. A definitive understanding of this phenomenon was not expected as this is a theoretical study relying on existing evidence gained from the study the motor in the laboratory. A plausible new stepping model has been developed following both the extensive consideration of existing experimental results and theory and the testing of hypotheses by means of a computer simulation.

The technical goal of the study was to pilot a method suitable for modelling AT. The simulation was designed and built for this study and is an original application of agent-based modelling to this field of inquiry. It has proved useful in investigating the kinesin walk and could support an AT model through the addition of further agents with properties derived from experimental findings and theory. Though the software has been useful on the small scale, extending it would be inefficient and produce an unwieldy tool

since it is written in C, a general programming language, as opposed to a tool designed for agent-based modelling.

5.2 The simulation

At first glance, the simulation may be viewed as little more than a toy. There is a striking contrast with most of the existing models of kinesin which use sets of equations to model varying quantities including those measured in experiments. A different methodology is used here: that of executable biology where algorithms are used to mimic biological entites.[84]

The simulation is a necessarily simplified representation of the motor moving along a section of MT. All models are a compromise between simplicity and realism, and must have regard for the computer power and programming effort available. It is believed that a reasonable balance has been struck here because, though accuracy and realism are limited, the simulation has proved useful in the development of the thesis which has due regard for experimental findings. It has enabled virtual experiments exploring the behaviour of different models of the motor under widely varying conditions and comparison of model to real motor behaviour. Specific advantages and limitations of the simulation are listed below.

The deterministic nature of the finite state machine might be thought to impose coordination on the heads and thus be an implicit form of gating. The simulation results show, however, that procession only emerges under some timing combinations and is influenced by stepping mechanism, linker strain, load and ATP concentration. Thus, the fact that event timings for each head are identical is not sufficient for procession; rather, this fixture enables direct comparison of the behaviour of the alternative models with and without gating. In any case, the argument for the necessity of gating put forward by

Rosenfeld et al. (2002) applies whether or not timings are fixed (see section 1.4.5.2).

5.2.1 Simulation advantages

- Enables exploration of the mechanism of kinesin motion by variation of component parameters in virtual experiments
- Visualises the behaviour of the motor during operation of the simulation
- Formal, consistent, executable model of head binding and hydrolysis
- Relatively transparent – straightforward rules as opposed to an intricate web of equations
- Small number of variables with few free parameters
- Potential for increasing scope by adding new agents
- Potential for development into a teaching tool
- Modest computational requirements
- Portable – written in the C programming language.

5.2.2 Simulation limitations

- Written in C programming language which is general purpose as opposed to being a specific agent-based modelling tool
- Requires C programming skills to alter and maintain
- Research software – not user friendly in its present form
- Highly abstract model of kinesin - lacks physical detail and mathematical rigour
- Not amenable to analytical treatment – simulation has to be run in order to get data.

5.3 The model

The rectified Brownian motion model of Mather and Fox (2006)[102] proposes that stepping is a diffusive motion rectified by neck linker zippering. Their model incorporates a gating mechanism that controls when ATP can bind in order to prevent loss of head coordination that would terminate procession.

The new model retains the notion that stepping is a diffusive motion rectified by zippering but assumes that no gating is necessary for procession. Kinesin's heads are connected by single polypeptide linkers that naturally act like springs as a result of thermal motion. This entropic linker strain is hypothesised as sufficient to coordinate the heads and thus facilitate procession. Theoretical support for this hypothesis comes from the results of computer simulation devised and implemented to investigate the kinesin walk as described in chapter 3. Further support is gained by considering the extent to which the model explains experimental findings including challenging data.

The initial argument put forward here is that the model is a parsimonious mechanism (being ungated) that provides wide explanatory power and is therefore a good candidate for explaining the kinesin walk. The ungated RBM stepping mechanism produces robust processive behaviour that is consistent with many experimental results obtained in the laboratory but fails at low [ATP] as we saw in section 4.1.2. Introducing gating, as described in section 2.4.6, rectifies the problem by slowing ATP hydrolysis while the partner head is not bound, thus providing coordination between the heads in addition to entropic linker strain. In conclusion, this study favours the thesis that the stepping mechanism of kinesin is gated RBM.

5.4 Gated RBM model challenged

The model predicts that kinesin should wait at a blockage since the free head is unable to reach the forward binding site and so the hydrolysis gate is invoked. This prediction is not compatible with the results of an *in vitro* study by Crevel et al. (2004) but tallies with the finding of Seitz and Surrey (2006) that the motor waits at an obstacle for an order of magnitude longer than the normal hydrolysis cycle.[40; 47]

The conflict between these studies stimulated the author to add the gating rule to the simulation. This achieved the required waiting time (see section 3.2.4) but it also rendered load behaviour less realistic (section 3.2.3), which would favour the ungated model. Further investigation, using an improved simulation, showed more realistic load behaviour for the gated model over a range of [ATP], as described in chapter 4.

5.5 RBM stepping challenged

While gating has proven necessary to faithfully simulate kinesin, RBM stepping remains the basis of the model. Experimental results that constitute challenges to RBM stepping are addressed in this section.

5.5.1 Power stroke revisited

If PS is the stepping mechanism of kinesin then, clearly, RBM cannot be. As discussed in section 1.4.3, there is evidence for a mechanism that overcomes the energy deficit problem with PS. Khalil et al. (2008) performed optical trap experiments on mutant kinesins with the cover strand removed.[68] The mutants showed a large reduction in stall force compared to wild-type. The reduction of stall force could be compensated for by a constant assisting load. They concluded that the cover strand is required for normal motor operation which involves PS stepping energised by cover neck bundle formation.

An alternative interpretation of these data is that efficient zippering requires formation of the cover neck bundle: without the cover strand, kinesin still walks but is crippled because the zippering process is impaired. The RBM model assumes that hindering loads act against zippering and zippering provides the forward bias of the motor. The model predicts, therefore, that the weaker zippering is, the smaller the force required to prevent zippering

and so stall the motor. The model also predicts that an assisting load will compensate for the impaired zippering by restoring forward bias. Both predictions are in accord with the above findings and so it is argued here that the data do not distinguish between RBM and PS.

The PS model lacks credibility, however, in the light of experimental findings that kinesin steps without ATP and without zippering (see section 5.6.3) since the power stroke requires zippering powered by ATP.

5.5.2 Wait state configuration

The model assumes that the detached head is free to diffuse in the wait state (kinesin waiting for ATP to bind). Evidence for and against this assumption comes from studies as described in section 1.4.4: bead studies indicate that one head is bound whereas fluorescence studies indicate that both heads are bound. Possible explanations for data challenging the assumption are discussed below.

5.5.2.1 Both heads bound?

Yildiz et al. (2004, 2008) propose that both heads are bound in the wait state.[35; 38] They labelled one head with a fluorophore and observed kinesin at a low ATP concentration to extend the duration of the wait state such that steps were discernable at the data capture rate of the apparatus. The head showed alternating movement averaging 0 nm and ~17 nm i.e. the head steps forward by the length of 2 tubulin dimers. This behaviour corresponds to alternate stepping of the heads as would be expected with hand-over-hand motion but points to the wait state configuration being both heads bound (or at least within 2 nm of a binding site since that is the calculated maximum

error on position value). This is because any other wait state configuration would introduce further signals into the data.

If the model is correct then the free head is diffusing in the wait state which should produce a signal averaging 8 nm i.e. half way between binding sites. A possible explanation is that the fluorophore tag is interfering with the free head movement. Carter and Cross (2006) propose that the fluorophore causes the motor to limp.[109] Perhaps the tagging introduces electrostatic attraction between the head and the MT that effectively parks the head close to the binding site or significantly increases the probability of the head being close to the binding site. Further experiment is required to test this hypothesis.

5.5.2.2 Parked head?

Alonso et al. (2007) propose that one head is detached from the MT in the wait state but that it is not free to diffuse.[83] They found that mixing kinesin with unpolymerised tubulin dimers caused only one head to bind in the absence of ATP. Their explanation is that the second head is parked, unable to bind, until released by the arrival of ATP.

The model predicts that both heads would bind free tubulin and so appears to be at odds with these data. There is considerable variation in the results depending on the type of kinesin and tubulin used. Neurospora kinesin mixed with yeast tubulin, for example, behaved as the model predicts: the ATPase rates are the same whether the tubulin is polymerised or not (figure 1A, Alonso et al. 2007)[83]. An alternative interpretation of their results derives from the configuration that kinesin takes without cargo: it is folded such that, although it will bind a microtubule, the tail inhibits normal procession.[110] It is possible that the tail is the source of the gating effect in some of these

experiments: it is obscuring the tubulin binding site of one head. Thus the findings may only apply to kinesin in solution and not to kinesin whose tail is bound to cargo.

5.5.3 ATP-binding gate necessary for procession?

Two sets of studies conclude that an ATP-binding gate is required to prevent kinesin from detaching before it takes the second step. Rosenfeld et al. (2002, 2003) conducted FRET studies to estimate biochemical rates.[59; 80] They calculated that by the time the first head has hydrolysed its ATP and released from the MT, the second head would also have hydrolysed its ATP. Thus procession is prevented as the motor detaches from the MT instead of taking the second step.

Since the ungated model displays procession over a wide range of timings including those estimated above, the present study does not support the need for gating.

5.5.4 Stalling mutants

Farrell et al. (2002) studied mutants with a defective head unable to hydrolyse ATP while Klumpp et al. (2004) studied mutants able to hydrolyse ATP but not detach from the MT.[60; 81] They found that the mutants stalled on the MT after one hydrolysis cycle. Their explanation is that the gate (as described in section 1.4.5.2) remains shut, prevents ATP from binding and thus any further stepping.

An alternative explanation for the stalling is that the mutations caused the zippering function to be defective. Without zippering, the second head fails to bind the MT. The mutants are then stalled in a wait state similar to that of wild-type kinesin (though the bound head is not nucleotide free).

5.6 Model predictions

5.6.1 Non-hydrolysable analogue

AMP-PNP, a non-hydrolysable analogue of ATP, produces intriguing behaviour *in vitro*. Guydosh and Block (2006) observed that AMP-PNP caused kinesin to take isolated backsteps during a long pause (up to several seconds) culminating in a final backstep before return to normal procession.[75] They hypothesise that the backward linker strain caused by a backstep increases the probability that the analogue is released from the leading head to be replaced by ATP thus restarting normal procession. Subramanian and Gelles (2007) repeated these experiments, observing the motor pause but with no movement; they suggest that this behaviour is the result of their experiments being conducted at zero applied force.[111] Their analysis also indicated that short processive runs occur with AMP-PNP bound.

The model predicts the long pause because the analogue behaves like ATP in that it causes the linker to zipper but the head remains bound since hydrolysis is necessary for detachment. The free head is thus held close to the forward binding site. This would result in the motor being stuck in place on the microtubule with the leading head futilely hydrolysing ATP until AMP-PNP dissociates. There is no backstepping while the motor is in this state, as found by Subramanian and Gelles *in vitro*. The model conflicts with their analysis, however, as it predicts no procession while AMP-PNP is bound since the analogue-bound head has high affinity for the MT.[31]

Consideration of the effect of load provides an explanation for backstepping. An important feature of the experimental setup of Guydosh and Block was the use of force-feedback to provide hindering loads of 4.5 pN and 5.3 pN in order

to be able to distinguish steps in the data given the data capture rate of their apparatus. It is proposed here that such loads are high enough to reduce the biasing effect of zippering such that the free head is occasionally able to diffuse close enough to the rear binding site to bind. A backstep is therefore possible but unlikely and so the model predicts an infrequent backstep during the pause, as observed *in vitro*.

An alternative hypothesis is suggested here to account for the terminal backstep before resumption of normal procession. The new proposal is that the cause of the terminal backstep is unbinding of AMP-PNP. When AMP-PNP finally dislodges, the linker unzippers since the empty head does not support zippering. The result of unzippering, given the rearward load, is that a backstep is taken. Procession then resumes when ATP binds the leading head.

5.6.2 Backsteps at low load

Backstepping at high load is explained by the result of load counteracting zippering as explained in the previous section but the model can also account for isolated backsteps as observed *in vitro* at low load (section 1.3.4). At low loads, zippering is unaffected so forward stepping would be predicted. Isolated backsteps at low load can be explained by considering the wait state. The wait state is the period after one head has released ADP and is tightly bound to the MT awaiting ATP to bind. The model assumes that its partner head is free to diffuse. Assuming that entropic neck linker tension makes binding and ADP release improbable (rather than impossible) in the wait state, there is a small probability that the free head will bind the rear site and release ADP thus the model predicts that kinesin takes an occasional backstep even at low load.

5.6.3 Wandering mutants and unfuelled procession

Yildiz et al. (2008) engineered mutants with altered neck linkers and compared their behaviour with wild-type kinesin finding relatively compromised procession.[38] They also found that external force can compensate for the absence of ATP, making wild-type kinesin walk despite the lack of hydrolysis. The model can account for most of these findings as explained below.

5.6.3.1 Extended linkers

At low ATP concentration, mutants with extended linkers slowed and showed lower stall force in proportion to the extension. Though they maintained direction on average, they also showed side stepping and more backstepping than wild-type. External assisting force was found to compensate for slowed procession.

The model predicts that lowering linker strain would result in compromised procession with an increase in wandering. The low concentration of ATP increases the time the motor spends in the wait state where entropic strain keeps the free head diffusing by restraining it like an elastic band. Lengthening the linkers brings more binding sites within range of the head while reducing the strain increases the likelihood of binding. Presumably the extended linkers are less efficient at zippering, the expected effect being a reduction in stall force since load is then acting against weaker zippering. While the linkers are still able to zipper the motor will retain an overall forward bias. Applying an assisting force would reinforce zippering and thus be expected to improve procession in these impaired motors.

5.6.3.2 Zero ATP walking

Native kinesin processed slowly in the direction of an externally applied force of 3 or 6 pN in the absence of ATP.

The model can partly explain this phenomenon. Without ATP, kinesin is in a wait state and no zippering occurs. The model predicts a very low probability of the free head binding in this state. Applying external force would increase the likelihood of the free head binding to the next site in the direction of the force but a step in that direction remains unlikely so progress would be slow. The result of the first step is that both heads would be empty and bound to the MT. A forward force of 6 pN would be sufficient to dislodge the trailing head to return the motor to the wait state and so sustain procession but 9 pN is required in the reverse direction.[31] While the linkers would be under strain thus adding to the external force, it is difficult to see how an external force of 3 pN could be sufficient to move the motor to take a second step.

5.6.3.3 Walking without zippering

A non-zippering mutant was immotile in the presence of ATP. This mutant was induced to process slowly in the direction of an applied external force without ATP (as above). When ATP was added, it moved faster.

The model predicts that removing zippering would stall the mutant regardless of ATP concentration since it remains in a wait state whether hydrolysing ATP or not. Without ATP, an external force will act on the mutant in the same way as for wild-type. Adding ATP would increase the mutant's speed under load because, after hydrolysis and phosphate release, the MT-bound head detaches without having to wait for ADP to dislodge it.

5.7 Experiments suggested by the model

A good model makes testable predictions suggesting further experiment. The model has generated alternative explanations of experimental findings generating the following testable predictions.

5.7.1 Gating

The behaviour of mutant motors has been proposed as demonstrating the need for gating of ATP-binding during the chemical cycle (section 1.4.5.2). The model suggests that this behaviour can be explained by the mutation causing a defect in the zippering function. This prediction could be tested by determining the state of the linker and the motor's configuration in the stalled mutant. If gating is the cause then the linker should be zipped and the motor configuration both heads bound. If zippering is faulty then the linker should be mobile and only one head of the motor bound.

5.7.2 Non-hydrolysable analogue

As noted in section 5.6.1, the model predicts that, during the pause, the motor is futilely hydrolysing ATP until AMP-PNP dissociates from the trailing head and that the final backstep occurs after AMP-PNP release (and not before as proposed by Guydosh and Block 2006)[75]. These predictions could be tested by determining the hydrolysis rate of the paused motor and the order of the backstep and analogue release events.

Chapter 6 Summary and conclusions

6.1 Summary

The long-term goal of this research is to contribute to combating dementia by shedding light on the process of its cause: neurodegeneration. The twin aims of this study are motivated by this goal. They were to investigate the detail of the kinesin walk and to pilot a simulation platform for modelling axonal transport. Axonal transport is vital for the normal functioning of neurons and its failure is implicated in neurodegenerative disease such as Alzheimer's. The molecular motor kinesin plays a major role in axonal transport by carrying cargo from the neuron cell body to the synapses.

6.2 The study

This study explores the motor's movement along its microtubule track using a computational simulation designed and built by the author. The simulation is the first implementation of an agent-based model of kinesin. This type of model was chosen because it has potential for modelling axonal transport, a biological system of much more complexity. The simulation was initially used to conduct four virtual experiments: comparison of stepping mechanisms, testing the hypothesis that gating is not required for procession, investigating the effect of load on the motor, and investigating the effect of placing a barrier on the track. These were conducted using a simplistic model of ATP arrival. A more realistic model of ATP arrival was incorporated in the program and a further set of virtual experiments was conducted to compare models under varying load and ATP concentration. An initial investigation was also made into the behaviour of multiple motors using loading to effectively link them as if bound to the same cargo.

6.3 Simulation results

6.3.1 Fixed ATP arrival results

The PS model of stepping described by Vale and Milligan (2000)[55] was compared to ungated RBM stepping over a range of head event timings and linker strains. Both gave rise to procession but RBM proved less sensitive to timing and strain variations.

The effect of gating on the processionary behaviour of the motor with RBM stepping was measured over a range of linker strains. Gating was found to reduce the number of timings yielding procession except at the extremes of linker strain whereas, without gating and at high values of strain, all the combinations led to procession. These results support the hypothesis that gating is not necessary and linker strain is sufficient for procession.

The effect of gating and load on processionary behaviour with RBM stepping was measured in terms of number of forward steps, backsteps and detachments. The motor behaved more realistically without the gate.

A barrier placed on the MT caused the motor to detach within a hydrolysis cycle without gating while the detachment was delayed with gating. The experimental evidence is divided on this issue so no firm conclusion can be made.

6.3.2 Random ATP arrival results

Varying the ATP concentration at no load confirmed the superiority of RBM over PS in terms of matching experimental data. The viability of the ungated RBM model to process was confirmed.

Applying a variable load in addition to varying the concentration yielded behaviour which supported the unloaded results except at low [ATP]. Gating of RBM stepping was required at low concentrations in order to replicate experimental data.

The behaviour of linked motors gave mixed indications: comparing the results with bead assay data, gated RBM matched the velocity data best but gated PS matched the run length data best. While bead and gliding assays agree in terms of velocity, they differ when it comes to run length (section 1.3.3) so no firm conclusions can be made. Linkage was expected to coordinate the motors in a similar manner to linkage between heads enabling procession but this only happened with PS: run length for linked RBM motors did not increase.

6.4 The model

The simulation results indicate that RBM stepping is more realistic than PS. They also support the initial hypothesis that head coordination can be achieved by entropic linker tension without the need for gating. Gating is required, however, to get realistic load results at low ATP concentration.

Further support for the RBM model derives from its ability to explain and predict experimental findings and the lack of definitive experimental challenges.

6.4.1 Explanatory power

The model has broad explanatory power as discussed in section 5.6 and correctly predicts that:

- During normal procession there will be infrequent backsteps

- A non-hydrolysable analogue will cause long pauses and isolated backstepping
- External force can make the motor process in the absence of ATP
- Mutants with extended linkers show weakened and wayward procession
- Mutants lacking zippering can be induced to process by applying external force.

It is difficult to see how the PS model could account for stepping without ATP or without zippering as the power stroke consists of zippering powered by ATP.

6.4.2 Challenges

In section 5.5 challenges to the model in four areas were addressed: the stepping mechanism, the wait state, ATP-binding gating and blockage behaviour. Alternative interpretations disarming the challenges are put forward though should the evidence against the model become definitive then the model would have to be modified or rejected.

6.5 Conclusions

The primary conclusion of this research is that a good model for kinesin stepping is rectified Brownian motion, as described by Mather and Fox (2006).[102] This conclusion is supported by the results of the computational simulation engineered and utilised in this study and by analysis of the results of laboratory experiments conducted by other researchers. The model explains a wide range of findings from *in vivo* and *in vitro* experiment and its predictions are born out by numerous experiments. As with any model, experimental evidence may yet prove it to be in error but it looks promising

and is at least a stimulus for debate and further experiment which will lead to a definitive understanding of the kinesin walk.

A less positive conclusion is proffered with respect to scaling up the simulation to model axonal transport. Though the present software could be expanded to include further agents, there is a limit to the size of a C program that can be easily maintained and modified. A more efficient, more easily manageable implementation system is required. The methodology would seem more promising: agent-based modelling is relatively simple, transparent and computationally efficient (compared to mathematical modelling). It has proved useful in this study and is expected to be able to capture essential features of axonal transport in order to explore failure modes and so pursue the goal of defeating dementia.

References

1. Watson, B., Friend, J. & Yeo, L. (2009). Piezoelectric ultrasonic resonant motor with stator diameter less than 250 μm: the Proteus motor. *Journal of Micromechanics and Microengineering* **19**.

2. Knapp, M. & Prince, M. (2007). Dementia UK. Alzheimer's Society, London.

3. Roy, S., Zhang, B., Lee, V. M. & Trojanowski, J. Q. (2005). Axonal transport defects: a common theme in neurodegenerative diseases. *Acta Neuropathol* **109**, 5-13.

4. De Vos, K. J., Grierson, A. J., Ackerley, S. & Miller, C. C. (2008). Role of axonal transport in neurodegenerative diseases. *Annu Rev Neurosci* **31**, 151-73.

5. Kandel, E. R., Schwartz, J. H. & Jessel, T. M. (2000). *Principles of Neural Science*. 4th edit, McGraw-Hill, New York.

6. Morfini, G. A., Burns, M., Binder, L., Kanaan, N. M., LaPointe, N., Bosco, D. A., Brown Jr., R. H., Brown, H., Tiwari, A., Hayward, L., Edgar, J., Nave, K.-A., Garberrn, J., Atagi, Y., Song, Y., Pigino, G. & Brady, S. T. (2009). Minisymposium: Axonal Transport Defects in Neurodegenerative Diseases. *J Neurosci.* **29**, 12776–86.

7. Kamal, A., Stokin, G. B., Yang, Z., Xia, C. & Goldstein, L. S. (2000). Axonal transport of amyloid precursor protein is mediated by direct binding to the kinesin light chain subunit of kinesin-I. *Neuron* **28**, 449-459.

8. Kamal, A., Almenar-Queralt, A., LeBlanc, J. F., Roberts, E. A. & Goldstein, L. S. B. (2001). Kinesin-mediated axonal transport of a membrane compartment containing beta-secretase and presenilin-1 requires APP. *Nature* **414**, 643–648.

9. Stokin, G. B., Lillo, C., Falzone, T. L., Brusch, R. G., Rockenstein, E., Mount, S. L., Raman, R., Davies, P., Masliah, E., Williams, D. S. & Goldstein, L. S. (2005). Axonopathy and transport deficits early in the pathogenesis of Alzheimer's disease. *Science* **307**, 1282-8.

10. Smith, K. D., Kallhoff, V., Zheng, H. & Pautler, R. G. (2007). In vivo axonal transport rates decrease in a mouse model of Alzheimer's disease. *Neuroimage* **35**, 1401-8.

11. Ittner, L. M., Fath, T., Ke, Y. D., Bi, M., van Eersel, J., Li, K. M., Gunning, P. & Gotz, J. (2008). Parkinsonism and impaired axonal transport in a mouse model of frontotemporal dementia. *Proc Natl Acad Sci U S A* **105**, 15997-6002.

12. Pigino, G., Morfini, G., Atagi, Y., Deshpande, A., Yu, C., Jungbauer, L., LaDu, M., Busciglio, J. & Brady, S. (2009). Disruption of fast axonal transport is a pathogenic mechanism for intraneuronal amyloid beta. *Proc Natl Acad Sci U S A* **106**, 5907-12.

13. Vossel, K. A., Zhang, K., Brodbeck, J., Daub, A. C., Sharma, P., Finkbeiner, S., Cui, B. & Mucke, L. (2010). Tau Reduction Prevents Aß-Induced Defects in Axonal Transport. *Science* **330**, 198.

14. Viel, A., Lue, R. A. & Liebler, J. (2006). From animation at http://multimedia.mcb.harvard.edu/anim_innerlife.html.

15. Hirokawa, N. (1998). Kinesin and dynein superfamily proteins and the mechanism of organelle transport. *Science* **279**, 519-26.

16. Brown, A. (2003). Axonal transport of membranous and nonmembranous cargoes: a unified perspective. *J Cell Biol* **160**, 817–21.

17. Pilling, A. D., Horiuchi, D., Lively, C. M. & Saxton, W. M. (2006). Kinesin-1 and Dynein Are the Primary Motors for Fast Transport of Mitochondria in Drosophila Motor Axons. *Mol Biol Cell* **17**, 2057–68.

18. Hollenbeck, P. J. & Saxton, W. M. (2005). The axonal transport of mitochondria. *J Cell Sci* **118**, 5411-9.

19. Nogales, E., Whittaker, M., Milligan, R. A. & Downing, K. H. (1999). High-resolution model of the microtubule. *Cell* **96**, 79-88.

20. Drubin, D. G. & Kirschner, M. W. (1986). Tau Protein Function in Living Cells. *J Cell Biol* **103**, 2739-46.

21. Conde, C. & Cáceres, A. (2009). Microtubule assembly, organization and dynamics in axons and dendrites. *Nat. Rev. Neurosci.* **10**, 319-332.

22. Brady, S. T. (1985). A novel brain ATPase with properties expected for the fast axonal transport motor. *Nature* **317**, 73-75.

23. Vale, R. D., Reese, T. S. & Sheetz, M. P. (1985). Identification of a novel force-generating protein, kinesin, involved in microtubule-based motility. *Cell* **42**, 39-50.

24. Kull, F. J., Sablin, E. P., Lau, R., Fletterick, R. J. & Vale, R. D. (1996). Crystal structure of the kinesin motor domain reveals a structural similarity to myosin. *Nature* **380**, 550-5.

25. Howard, J. (2001). *Mechanics of Motor Proteins of the Cytoskeleton*, Sinauer Associates, Inc., Sunderland, MA, USA.

26. Vale, R. D. (2003). The molecular motor toolbox for intracellular transport. *Cell* **112**, 467-80.

27. Hackney, D. D. (1994). Evidence for alternating head catalysis by kinesin during microtubule-stimulated ATP hydrolysis. *Proc Natl Acad Sci U S A* **91**, 6865-9.

28. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. & Walter, P. (2002). *Molecular Biology of the Cell*, Garland Science, NY.

29. Amos, L. A. & Hirose, K. (2007). A cool look at the structural changes in kinesin motor domains. *J Cell Sci* **120**, 3919-27.

30. Bodey, A. J., Kikkawa, M. & Moores, C. A. (2009). 9-Angstrom structure of a microtubule-bound mitotic motor. *J Mol Biol* **388**, 218-24.

31. Uemura, S., Kawaguchi, K., Yajima, J., Edamatsu, M., Toyoshima, Y. Y. & Ishiwata, S. (2002). Kinesin-microtubule binding depends on both nucleotide state and loading direction. *Proc Natl Acad Sci U S A* **99**, 5977-81.

32. Sosa, H., Peterman, E. J., Moerner, W. E. & Goldstein, L. S. (2001). ADP-induced rocking of the kinesin motor domain revealed by single-molecule fluorescence polarization microscopy. *Nat Struct Biol* **8**, 540-4.

33. Svoboda, K., Schmidt, C. F., Schnapp, B. J. & Block, S. M. (1993). Direct observation of kinesin stepping by optical trapping interferometry. *Nature* **365**, 721-7.

34. Allen, R. D., Metuzals, J., Tasaki, I., Brady, S. T. & Gilbert, S. P. (1982). Fast Axonal Transport in Squid Giant Axon. *Science* **218**, 1127-1129.

35. Yildiz, A., Tomishige, M., Vale, R. D. & Selvin, P. R. (2004). Kinesin walks hand-over-hand. *Science* **303**, 676-8.

36. Coppin, C. M., Pierce, D. W., Hsu, L. & Vale, R. D. (1997). The load dependence of kinesin's mechanical cycle. *Proc Natl Acad Sci U S A* **94**, 8539-44.

37. Carter, N. J. & Cross, R. A. (2005). Mechanics of the kinesin step. *Nature* **435**, 308-12.

38. Yildiz, A., Tomishige, M., Gennerich, A. & Vale, R. D. (2008). Intramolecular strain coordinates kinesin stepping behavior along microtubules. *Cell* **134**, 1030-41.

39. Block, S. M., Goldstein, L. S. & Schnapp, B. J. (1990). Bead movement by single kinesin molecules studied with optical tweezers. *Nature* **348**, 348-52.

40. Seitz, A. & Surrey, T. (2006). Processive movement of single kinesins on crowded microtubules visualized using quantum dots. *Embo J* **25**, 267-77.

41. Vale, R. D. (1996). Switches, latches, and amplifiers: common themes of G proteins and molecular motors. *J Cell Biol* **135**, 291-302.

42. Hancock, W. O. & Howard, J. (1998). Processivity of the motor protein kinesin requires two heads. *J Cell Biol* **140**, 1395-405.

43. Hancock, W. O. & Howard, J. (1999). Kinesin's processivity results from mechanical and chemical coordination between the ATP hydrolysis cycles of the two motor domains. *Proc Natl Acad Sci U S A* **96**, 13147-52.

44. Courty, S., Luccardini, C., Bellaiche, Y., Cappello, G. & Dahan, M. (2006). Tracking individual kinesin motors in living cells using single quantum-dot imaging. *Nano Lett* **6**, 1491-5.

45. Svoboda, K. & Block, S. M. (1994). Force and velocity measured for single kinesin molecules. *Cell* **77**, 773-84.

46. Visscher, K., Schnitzer, M. J. & Block, S. M. (1999). Single kinesin molecules studied with a molecular force clamp. *Nature* **400**, 184-9.

47. Crevel, I. M., Nyitrai, M., Alonso, M. C., Weiss, S., Geeves, M. A. & Cross, R. A. (2004). What kinesin does at roadblocks: the coordination mechanism for molecular walking. *Embo J* **23**, 23-32.

48. Schnitzer, M. J. & Block, S. M. (1997). Kinesin hydrolyses one ATP per 8-nm step. *Nature* **388**, 386-90.

49. Coy, D. L., Wagenbach, M. & Howard, J. (1999). Kinesin takes one 8-nm step for each ATP that it hydrolyzes. *J Biol Chem* **274**, 3667-71.

50. Hua, W., Young, E. C., Fleming, M. L. & Gelles, J. (1997). Coupling of kinesin steps to ATP hydrolysis. *Nature* **388**, 390-3.

51. Hua, W., Chung, J. & Gelles, J. (2002). Distinguishing inchworm and hand-over-hand processive kinesin movement by neck rotation measurements. *Science* **295**, 844-8.

52. Block, S. M., Asbury, C. L., Shaevitz, J. W. & Lang, M. J. (2003). Probing the kinesin reaction cycle with a 2D optical force clamp. *Proc Natl Acad Sci U S A* **100**, 2351-6.

53. Asbury, C. L., Fehr, A. N. & Block, S. M. (2003). Kinesin moves by an asymmetric hand-over-hand mechanism. *Science* **302**, 2130-4.

54. Ray, S., Meyhofer, E., Milligan, R. A. & Howard, J. (1993). Kinesin follows the microtubule's protofilament axis. *J Cell Biol* **121**, 1083-93.

55. Vale, R. D. & Milligan, R. A. (2000). The way things move: looking under the hood of molecular motor proteins. *Science* **288**, 88-95.

56. Rayment, I., Holden, H. M., Whittaker, M., Yohn, C. B., Lorenz, M., Holmes, K. C. & Milligan, R. A. (1993). Structure of the Actin-Myosin Complex and Its Implications for Muscle Contraction. *Science* **261**, 58-65.

57. Piezzesi, G., Reconditi, M., Linari, M., Lucii, L., Sun, Y.-B., Narayanan, T., Boesecke, P., Lombardi, V. & Irving, M. (2002). Mechanism of force generation by myosin heads in skeletal muscle. *Nature* **415**, 659-62.

58. Rice, S., Lin, A. W., Safer, D., Hart, C. L., Naber, N., Carragher, B. O., Cain, S. M., Pechatnikova, E., Wilson-Kubalek, E. M., Whittaker, M., Pate, E., Cooke, R., Taylor, E. W., Milligan, R. A. & Vale, R. D. (1999). A structural change in the kinesin motor protein that drives motility. *Nature* **402**, 778-84.

59. Rosenfeld, S. S., Fordyce, P. M., Jefferson, G. M., King, P. H. & Block, S. M. (2003). Stepping and stretching. How kinesin uses internal strain to walk processively. *J Biol Chem* **278**, 18550-6.

60. Klumpp, L. M., Hoenger, A. & Gilbert, S. P. (2004). Kinesin's second step. *Proc Natl Acad Sci U S A* **101**, 3444-9.

61. Asenjo, A. B., Krohn, N. & Sosa, H. (2003). Configuration of the two kinesin motor domains during ATP hydrolysis. *Nat Struct Biol* **10**, 836-42.

62. Asenjo, A. B., Weinberg, Y. & Sosa, H. (2006). Nucleotide binding and hydrolysis induces a disorder-order transition in the kinesin neck-linker region. *Nat Struct Mol Biol* **13**, 648-54.

63. Tomishige, M., Stuurman, N. & Vale, R. D. (2006). Single-molecule observations of neck linker conformational changes in the kinesin motor protein. *Nat Struct Mol Biol* **13**, 887-94.

64. 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**, 590-3.

65. 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.

66. Rice, S., Cui, Y., Sindelar, C., Naber, N., Matuska, M., Vale, R. & Cooke, R. (2003). Thermodynamic properties of the kinesin neck-region docking to the catalytic core. *Biophys J* **84**, 1844-54.

67. Hwang, W., Lang, M. J. & Karplus, M. (2008). Force generation in kinesin hinges on cover-neck bundle formation. *Structure* **16**, 62-71.

68. Khalil, A. S., Appleyard, D. C., Labno, A. K., Georges, A., Karplus, M., Belcher, A. M., Hwang, W. & Lang, M. J. (2008). Kinesin's cover-neck bundle folds forward to generate force. *Proc Natl Acad Sci U S A* **105**, 19247-52.

69. Schnitzer, M. J., Visscher, K. & Block, S. M. (2000). Force production by single kinesin motors. *Nat Cell Biol* **2**, 718-23.

70. Coppin, C. M., Finer, J. T., Spudich, J. A. & Vale, R. D. (1996). Detection of sub-8-nm movements of kinesin by high-resolution optical-trap microscopy. *Proc Natl Acad Sci U S A* **93**, 1913-7.

71. Nishiyama, M., Muto, E., Inoue, Y., Yanagida, T. & Higuchi, H. (2001). Substeps within the 8-nm step of the ATPase cycle of single kinesin molecules. *Nat Cell Biol* **3**, 425-8.

72. Fox, R. F. & Choi, M. H. (2001). Rectified Brownian motion and kinesin motion along microtubules. *Phys Rev E Stat Nonlin Soft Matter Phys* **63**, 051901.

73. Hackney, D. D. (2007). Biochemistry. Processive motor movement. *Science* **316**, 58-9.

74. Kawaguchi, K. & Ishiwata, S. (2001). Nucleotide-dependent single- to double-headed binding of kinesin. *Science* **291**, 667-9.

75. Guydosh, N. R. & Block, S. M. (2006). Backsteps induced by nucleotide analogs suggest the front head of kinesin is gated by strain. *Proc Natl Acad Sci U S A* **103**, 8054-9.

76. Hirose, K., Lockhart, A., Cross, R. A. & Amos, L. A. (1996). Three-dimensional cryoelectron microscopy of dimeric kinesin and ncd motor domains on microtubules. *Proc Natl Acad Sci U S A* **93**, 9539-44.

77. Arnal, I. & Wade, R. H. (1998). Nucleotide-dependent conformations of the kinesin dimer interacting with microtubules. *Structure* **6**, 33-8.

78. Tomishige, M. & Vale, R. D. (2000). Controlling kinesin by reversible disulfide cross-linking. Identifying the motility-producing conformational change. *J Cell Biol* **151**, 1081-92.

79. Crevel, I., Carter, N., Schliwa, M. & Cross, R. (1999). Coupled chemical and mechanical reaction steps in a processive Neurospora kinesin. *Embo J* **18**, 5863-72.

80. Rosenfeld, S. S., Xing, J., Jefferson, G. M., Cheung, H. C. & King, P. H. (2002). Measuring kinesin's first step. *J Biol Chem* **277**, 36731-9.

81. Farrell, C. M., Mackey, A. T., Klumpp, L. M. & Gilbert, S. P. (2002). The role of ATP hydrolysis for kinesin processivity. *J Biol Chem* **277**, 17079-87.

82. Mori, T., Vale, R. D. & Tomishige, M. (2007). How kinesin waits between steps. *Nature* **450**, 750-4.

83. Alonso, M. C., Drummond, D. R., Kain, S., Hoeng, J., Amos, L. & Cross, R. A. (2007). An ATP gate controls tubulin binding by the tethered head of kinesin-1. *Science* **316**, 120-3.

84. Fisher, J. & Henzinger, T. A. (2007). Executable cell biology. *Nat Biotechnol* **25**, 1239-49.

85. Kolomeisky, A. B. & Fisher, M. E. (2007). Molecular motors: a theorist's perspective. *Annu Rev Phys Chem* **58**, 675-95.

86. Astumian, R. D. & Derenyi, I. (1999). A chemically reversible Brownian motor: application to kinesin and Ncd. *Biophys J* **77**, 993-1002.

87. Bier, M. (2007). The stepping motor protein as a feedback control ratchet. *Biosystems* **88**, 301-7.

88. Derenyi, I. & Vicsek, T. (1996). The kinesin walk: a dynamic model with elastically coupled heads. *Proc Natl Acad Sci U S A* **93**, 6775-9.

89. Kanada, R. & Sasaki, K. (2003). Theoretical model for motility and processivity of two-headed molecular motors. *Phys Rev E Stat Nonlin Soft Matter Phys* **67**, 061917.

90. Fisher, M. E. & Kolomeisky, A. B. (2001). Simple mechanochemistry describes the dynamics of kinesin molecules. *Proc Natl Acad Sci U S A* **98**, 7748-53.

91. Maes, C. & van Wieren, M. H. (2003). A Markov Model for Kinesin. *Journal of Statistical Physics* **112**, 329-355.

92. Mogilner, A., Fisher, A. J. & Baskin, R. J. (2001). Structural changes in the neck linker of kinesin explain the load dependence of the motor's mechanical cycle. *J Theor Biol* **211**, 143-57.

93. Thomas, N., Imafuku, Y., Kamiya, T. & Tawada, K. (2002). Kinesin: a molecular motor with a spring in its step. *Proc Biol Sci* **269**, 2363-71.

94. Shao, Q. & Gao, Y. Q. (2006). On the hand-over-hand mechanism of kinesin. *Proc Natl Acad Sci U S A* **103**, 8072-7.

95. Xie, P., Dou, S. X. & Wang, P. Y. (2006). Mechanochemical couplings of kinesin motors. *Biophys Chem* **123**, 58-76.

96. Duke, T. & Leibler, S. (1996). Motor Protein Mechanics: A Stochastic Model with Minimal Mechanochemical Coupling. *Biophysical Journal* **71**, 1235-1247.

97. Bolterauer, H., Tuszynski, J. A. & Unger, E. (2005). Directed binding: a novel physical mechanism that describes the directional motion of two-headed kinesin motor proteins. *Cell Biochem Biophys* **42**, 95-119.

98. Peskin, C. S. & Oster, G. (1995). Coordinated hydrolysis explains the mechanical behavior of kinesin. *Biophys J* **68**, 202S-210S; discussion 210S-211S.

99. Atzberger, P. J. & Peskin, C. S. (2006). A Brownian Dynamics model of kinesin in three dimensions incorporating the force-extension profile of the coiled-coil cargo tether. *Bull Math Biol* **68**, 131-60.

100. Endow, S. A. & Higuchi, H. (2000). A mutant of the motor protein kinesin that moves in both directions on microtubules. *Nature* **406**, 913-6.

101. Endow, S. A. & Waligora, K. W. (1998). Determinants of kinesin motor polarity. *Science* **281**, 1200-2.

102. Mather, W. H. & Fox, R. F. (2006). Kinesin's biased stepping mechanism: amplification of neck linker zippering. *Biophys J* **91**, 2416-26.

103. Schief, W. R., Clark, R. H., Crevenna, A. H. & Howard, J. (2004). Inhibition of kinesin motility by ADP and phosphate supports a hand-over-hand mechanism. *Proc Natl Acad Sci U S A* **101**, 1183-8.

104. Knuth, D. E. (1969). *Seminumerical Algorithms*. The Art of Computer Programming, 2, Addison Wesley, Reading, Mass.

105. Asbury, C. L. (2005). Kinesin: world's tiniest biped. *Curr Opin Cell Biol* **17**, 89-97.

106. Nishiyama, M., Higuchi, H. & Yanagida, T. (2002). Chemomechanical coupling of the forward and backward steps of single kinesin molecules. *Nat Cell Biol* **4**, 790-7.

107. Yajima, J., Alonso, M. C., Cross, R. A. & Toyoshima, Y. Y. (2002). Direct Long-Term Observation of Kinesin Processivity at Low Load. *Curr Biol* **12**, 301-6.

108. Beeg, J., Klumpp, S., Dimova, R., Gracia, R. S., Unger, E. & Lipowsky, R. (2008). Transport of Beads by Several Kinesin Motors. *Biophys J* **94**, 532-41.

109. Carter, N. J. & Cross, R. A. (2006). Kinesin's moonwalk. *Curr Opin Cell Biol* **18**, 61-7.

110. Cross, R. & Scholey, J. (1999). Kinesin: the tail unfolds. *Nat Cell Biol* **1**, E119-21.

111. Subramanian, R. & Gelles, J. (2007). Two distinct modes of processive kinesin movement in mixtures of ATP and AMP-PNP. *J Gen Physiol* **130**, 445-55.

Appendix A Load data

The data in tables A.1-3 were generated by the simulation of the motor over five runs at each load value as described in section 3.2.3. Tables A.4-5 record the step ratios calculated from these data which in turn are used to calculate the minimum and maximum data points as described in section 3.2.3.

Table A.1 Forward steps

	No gate – forward steps					
Load	*1*	*2*	*3*	*4*	*5*	Average
3	47	48	51	51	49	**49.2**
3.5	49	49	47	51	47	**48.6**
4	47	48	49	48	46	**47.6**
4.5	49	49	49	50	45	**48.4**
5	51	55	56	56	57	**55**
5.5	58	51	60	60	58	**57.4**
6	67	47	54	55	66	**57.8**
6.5	53	63	78	42	61	**59.4**
7	60	42	113	89	110	**82.8**
7.5	26	39	25	17	13	**24**
8	33	27	15	91	57	**44.6**
8.5	43	60	16	33	84	**47.2**
9	60	20	50	16	16	**32.4**

	ATP gate – forward steps					
Load	*1*	*2*	*3*	*4*	*5*	Average
3	47	45	48	48	48	**47.2**
3.5	48	48	49	48	49	**48.4**
4	47	48	48	48	48	**47.8**
4.5	49	48	48	48	48	**48.2**
5	49	48	49	47	49	**48.4**
5.5	47	49	46	49	48	**47.8**
6	47	47	47	45	39	**45**
6.5	50	48	54	49	49	**50**
7	47	53	53	61	48	**52.4**
7.5	51	51	43	46	48	**47.8**
8	50	48	52	49	51	**50**
8.5	50	52	48	51	49	**50**
9	48	51	47	41	50	**47.4**

Table A.2 Backward steps

	No gate – back steps					
Load	*1*	*2*	*3*	*4*	*5*	Average
3	1	1	2	9	2	**3**
3.5	1	4	1	2	1	**1.8**
4	0	2	6	0	3	**2.2**
4.5	3	1	1	4	3	**2.4**
5	2	2	2	3	1	**2**
5.5	2	5	1	11	2	**4.2**
6	3	7	3	6	3	**4.4**
6.5	3	3	12	7	7	**6.4**
7	16	6	19	7	17	**13**
7.5	27	39	25	15	11	**23.4**
8	35	38	18	98	64	**50.6**
8.5	29	78	15	29	77	**45.6**
9	65	11	33	6	20	**27**

	ATP gate – back steps					
Load	*1*	*2*	*3*	*4*	*5*	Average
3	1	2	0	2	2	**1.4**
3.5	3	1	3	0	3	**2**
4	2	2	3	2	0	**1.8**
4.5	1	0	0	3	2	**1.2**
5	4	1	9	3	4	**4.2**
5.5	1	5	2	4	2	**2.8**
6	6	9	5	4	5	**5.8**
6.5	63	47	63	32	29	**46.8**
7	44	39	60	68	44	**51**
7.5	63	56	44	40	50	**50.6**
8	24	51	48	26	48	**39.4**
8.5	51	49	54	42	72	**53.6**
9	44	73	27	53	38	**47**

Table A.3 Detachments

	No gate – detachments					
Load	*1*	*2*	*3*	*4*	*5*	Average
3	0	0	0	0	0	**0**
3.5	0	0	0	0	0	**0**
4	0	0	0	0	0	**0**
4.5	0	0	0	0	0	**0**
5	0	3	0	1	3	**1.4**
5.5	3	6	7	6	5	**5.4**
6	9	13	11	13	12	**11.6**
6.5	14	26	31	18	27	**23.2**
7	38	29	72	44	65	**49.6**
7.5	25	34	23	16	12	**22**
8	27	26	14	85	49	**40.2**
8.5	39	58	15	30	74	**43.2**
9	58	19	48	15	15	**31**

	ATP gate - detachments					
Load	*1*	*2*	*3*	*4*	*5*	Average
3	0	0	0	0	0	**0**
3.5	0	0	0	0	0	**0**
4	0	0	0	0	0	**0**
4.5	0	0	0	0	0	**0**
5	0	0	0	0	0	**0**
5.5	0	0	0	0	0	**0**
6	0	0	0	0	0	**0**
6.5	8	5	8	0	2	**4.6**
7	5	8	7	7	2	**5.8**
7.5	6	3	3	2	10	**4.8**
8	4	5	6	0	4	**3.8**
8.5	3	3	5	3	5	**3.8**
9	3	7	1	6	5	**4.4**

Table A.4 Ungated step ratios

	Forward step ratios				
Load	*1*	*2*	*3*	*4*	*5*
3	0.979	0.98	0.962	0.85	0.961
3.5	0.98	0.925	0.979	0.962	0.979
4	1	0.96	0.891	1	0.939
4.5	0.942	0.98	0.98	0.926	0.938
5	0.962	0.917	0.966	0.933	0.934
5.5	0.921	0.823	0.882	0.779	0.892
6	0.848	0.701	0.794	0.743	0.815
6.5	0.757	0.685	0.645	0.627	0.642
7	0.526	0.545	0.554	0.636	0.573
7.5	0.333	0.348	0.342	0.354	0.361
8	0.347	0.297	0.319	0.332	0.335
8.5	0.387	0.306	0.348	0.359	0.357
9	0.328	0.4	0.382	0.432	0.314
	Backstep ratios				
Load	*1*	*2*	*3*	*4*	*5*
3	0.021	0.02	0.038	0.15	0.039
3.5	0.02	0.075	0.021	0.038	0.021
4	0	0.04	0.109	0	0.061
4.5	0.058	0.02	0.02	0.074	0.063
5	0.038	0.033	0.034	0.05	0.016
5.5	0.032	0.081	0.015	0.143	0.031
6	0.038	0.104	0.044	0.081	0.037
6.5	0.043	0.033	0.099	0.104	0.074
7	0.14	0.078	0.093	0.05	0.089
7.5	0.346	0.348	0.342	0.313	0.306
8	0.368	0.418	0.383	0.358	0.376
8.5	0.261	0.398	0.326	0.315	0.328
9	0.355	0.22	0.252	0.162	0.392
	Detachment ratios				
Load	*1*	*2*	*3*	*4*	*5*
3	0	0	0	0	0
3.5	0	0	0	0	0
4	0	0	0	0	0
4.5	0	0	0	0	0
5	0	0.05	0	0.017	0.049
5.5	0.048	0.097	0.103	0.078	0.077
6	0.114	0.194	0.162	0.176	0.148
6.5	0.2	0.283	0.256	0.269	0.284
7	0.333	0.377	0.353	0.314	0.339
7.5	0.321	0.304	0.315	0.333	0.333
8	0.284	0.286	0.298	0.31	0.288
8.5	0.351	0.296	0.326	0.326	0.315
9	0.317	0.38	0.366	0.405	0.294

Table A.5 Gated step ratios

	Forward step ratios				
Load	*1*	*2*	*3*	*4*	*5*
3	0.979	0.957	1	0.96	0.96
3.5	0.941	0.98	0.942	1	0.942
4	0.959	0.96	0.941	0.96	1
4.5	0.98	1	1	0.941	0.96
5	0.925	0.98	0.845	0.94	0.925
5.5	0.979	0.907	0.958	0.925	0.96
6	0.887	0.839	0.904	0.918	0.886
6.5	0.413	0.48	0.432	0.605	0.613
7	0.49	0.53	0.442	0.449	0.511
7.5	0.425	0.464	0.478	0.523	0.444
8	0.641	0.462	0.491	0.653	0.495
8.5	0.481	0.5	0.449	0.531	0.389
9	0.505	0.389	0.627	0.41	0.538
	Backstep ratios				
Load	*1*	*2*	*3*	*4*	*5*
3	0.021	0.043	0	0.04	0.04
3.5	0.059	0.02	0.058	0	0.058
4	0.041	0.04	0.059	0.04	0
4.5	0.02	0	0	0.059	0.04
5	0.075	0.02	0.155	0.06	0.075
5.5	0.021	0.093	0.042	0.075	0.04
6	0.113	0.161	0.096	0.082	0.114
6.5	0.521	0.47	0.504	0.395	0.363
7	0.458	0.39	0.5	0.5	0.468
7.5	0.525	0.509	0.489	0.455	0.463
8	0.308	0.49	0.453	0.347	0.466
8.5	0.49	0.471	0.505	0.438	0.571
9	0.463	0.557	0.36	0.53	0.409
	Detachment ratios				
Load	*1*	*2*	*3*	*4*	*5*
3	0	0	0	0	0
3.5	0	0	0	0	0
4	0	0	0	0	0
4.5	0	0	0	0	0
5	0	0	0	0	0
5.5	0	0	0	0	0
6	0	0	0	0	0
6.5	0.066	0.05	0.064	0	0.025
7	0.052	0.08	0.058	0.051	0.021
7.5	0.05	0.027	0.033	0.023	0.093
8	0.051	0.048	0.057	0	0.039
8.5	0.029	0.029	0.047	0.031	0.04
9	0.032	0.053	0.013	0.06	0.054

Appendix B Program listing

B.1 Main listing – program for chapter 3 results

```
/*
  Author: Richard Wilson, MOAC DTC, Coventry House, Warwick University, CV4 7AL, UK
      richard.j.wilson@warwick.ac.uk

  Purpose:
    To investigate the motion of the molecular motor kinesin by changing stepping mechanism,
relative timings of ATP hydrolysis, linker strain and load.

  Research questions:
    Is there a difference in behaviour between power stroke and RBM stepping?
    Does linker tension suffice to coordinate the heads (or is gate necessary)?
    How does motor react to blockage?
    How does motor react to load?

  Latest change to program: prep for multiple motors ***unfinished – this version is single motor
only***
*/

#include <stdio.h>
#include <stdbool.h>

//GLOBAL DECLARATIONS

//simulation run parameters
bool ATP_gate=false;  //ATP_hydrolysis gate switch - see update_heads.h
bool RBM=true;
//true if rectified Brownian model (else power stroke) - see update_heads.h
bool AMP_PNP=false; //indicates whether non-hydrolysable analogue is //bound to a head - see
update_heads.h

float linker_tension; //linker tension variable
float linker_tension_max=10.; //maximum value of linker tension
//at this value of linker_tension, binding of free head prevented when kinesin //in wait state
(K0.KDu)

float back_load; //hindering load applied to motor
const float stall_load=6.;
//load that defeats zippering (see zippering routine, update_heads.h)
const float load_variation=3.;
//random variation of load (see zippering routine in update_heads.h)

const int last_motor=1;//number of motors simulated

int run; //variable for number of duplicate runs at given parameters
result_max=2; //timing parameter loop control, must be positive integer
const int ATP_result_max=2;
//as above but may want to use different range for ATP_binding parameter
int results[result_max][result_max][result_max][ATP_result_max]; //temporary storage for
results to enable organising before output to file

//kinesin head state
struct head_struct
  {
  int MTbinding;   //whether kinesin head bound to MT or free
  int nuc_binding;          //which nucleotide is bound to kinesin, if any

  int ATP_binding_count;  //timing counter for ATP binding (K0 -> KT)
  int hydrolysis_count;     //timing counter for ATP hydrolysis (KT -> KDP)
  int P_release_count;
      //timing counter for phosphate and head release (KDP -> KDu)
```

```
  int MT_binding_count;    //timing counter for MT binding (KDu -> KD)
  int ADP_release_count;  //timing counter for ADP release (KD -> K0)

  int prev_posx; //last x pos
  int prev_posy; //last y pos
  int posx;         //current x position of head
  int posy;         //current y position of head
  };

//kinesin motor has 2 heads
struct motor_struct
  {
  struct head_struct heada;
  struct head_struct headb;
  };

//array of motors for multiple motor experiments
struct motor_struct motor_array[last_motor];

//head state values
const int k_free=2, k_bound=1;
//signifies whether kinesin bound to MT or not
enum {null,ATP,ADPP,ADP}; //signifies which nucleotide is attached, if any

//motor state values - used in analysis of behaviour
enum {P_diffusion, P_stuck, P_processive};
//diffuses - no procession, frozen on MT, processive motor

//rectangular array representing section of cytosol
const int row_max=15, col_max=100;
int cytosol[row_max+1][col_max+1];
//NB indexing is positive from top left (cytosol[0][0])
const int cytosol_rgb=220;
//pale grey for empty cytosol box used in display.h

//markers for contents of cytosol are all negative (as used by clash routine)
const int MT_null=-99; //interior of MT
const int MT_alpha=-98, MT_beta=-97; //microtubule alpha and beta tubulin
const int head_display=-1; //kinesin head
const int head_prev=1; //represents where heads have been so a track of //kinesin motion is
displayed

//timing parameter variables - used to trigger event when head timing //counter (see
head_struct, above )reaches value
int t_ATPhyd; //KT -> KDP
int t_Prel; //KDP -> KDu
int t_ADPrel; //KD -> K0
int t_ATPbind; //K0 -> KT

const int t_MTbind=0; //delay before KD binds MT when close to MT

int blockage_count=0;
//blockage timer - to time how long blockage placed in the way of kinesin
const int blockage_limit=7; //number of time slices blockage in place

//trace and analysis declarations
bool trace=false; //switch trace of kinesin movement on or off
int uucount;
//tally of successive KDu.KDu states - used in trace_heads() in analysis.h

//flags used in update_heads.h to indicate if step taken
bool step; //set by forward_step()
bool backstep; //set by back_step()

//-----------------------------------------------
//FUNCTION DECLARATIONS AND DEFINITIONS
//-----------------------------------------------

//display functions
```

```
#include "display.h"

//Brownian motion routines
#include "Brownian.h"

//head update routines
#include "update_heads.h"

//analysis and trace routines
#include "analysis.h"

//routines used by mp8()
void wait(float num)
//slows down computer if motor goes too fast to observe movement
  {
  for (float x=0.; x<=num && x>=0; x=x+0.001);
  }

bool motor_in_play(void)
//if motor has reached RHS of MT then false else true
  {//***Code NOT ready for multiple motor experiments
  struct head_struct *head1=&motor_array[0].heada,*head2=&motor_array[0].headb;

  if (head1->posx >= col_max-1 || head2->posx >= col_max-1)
    return false;
      //kinesin has reached rightmost end of MT
  return true; //kinesin still moving about in cytosol box
  }

//-------------------------
//START OF MAIN ROUTINE
//-------------------------
int mp8(HWND hwnd)
  {
  PAINTSTRUCT ps;
  HDC hdc = BeginPaint(hwnd,&ps); //MS Windows specific routine

  FILE *f=fopen((RBM?"mp8RBM.txt":"mp8PS.txt"),"w");
    //open file for output data
    //file name distinguishes stepping mechanism in use
  if (f == 0)
    {
    UpdateStatusBar("File open failed", 0, 0); //notify user if file open failure
    EndPaint(hwnd,&ps);
    return 0; //exit program
    }

  FILE *fa=fopen((RBM?"mp8aRBM.txt":"mp8aPS.txt"),"w");
 //open file for analysis data (RBM or power stroke model)
  if (fa == 0)
    {
     UpdateStatusBar("Analysis file open failed", 0, 0);
    fprintf(f,"\nError: Analysis file open failed ===mp8 finished.\n");
    fclose(f); //close data file
    EndPaint(hwnd,&ps);
    return 0; //exit program
    }

  //write headers to files
  fprintf(f,"===mp8 started. Load variation %.1f Stall %.1f linker strain %.2f MT binding %d
  %s\n",
    load_variation,stall_load,linker_tension,t_MTbind,
    (ATP_gate?"ATP gate":"no ATP gate"));
  fprintf(fa,"===mp8 started.        Timing variations %d, Load variation %.1f Stall %.1f linker
    strain %.2f MT binding %d %s\n", result_max*result_max*result_max*ATP_result_max,
    load_variation,stall_load,linker_tension, t_MTbind,
    (ATP_gate?"ATP gate":"no ATP gate"));

  //set results array to -1 to show up any missing data
```

```
  for (int i=0; i<result_max; i++)
    for (int j=0; j<result_max; j++)
      for (int k=0; k<result_max; k++)
        for (int l=0; l < ATP_result_max; l++)
        {
        results[i][j][k][l]=-1;
        }

  //counters for values measured for each set of runs of experiment
  int heads_bound_count;
  int detachment_count;
  int steps;
  int backsteps;
  int timing_loop_count;

  //counters for average values over several runs with same parameters
  int av_steps;
  int av_backsteps;
  int av_detachments;
  int av_time;

  int heads_stuck_max=200;
//if heads don't move for this amount of time then assume kinesin stuck

//-----------------------------------------------
//ALTERNATE CODE for different experiments
//-----------------------------------------------
//LINKER LOOP
//   for (linker_tension=linker_tension_max; linker_tension>=0.; linker_tension--)
//     {//outer loop for testing effect of changing LINKER tension
//     back_load=0; //set load to zero if varying linker tension
//     heads_stuck_max=500*result_max*(int)(linker_tension_max - linker_tension + 1);
       //expect increase of motor stall with decrease linker strain - not stuck

//OR fix tension
linker_tension=9.5;
//tension set to give occasional back steps (as determined by bead assays)

//LOAD LOOP
//   for (back_load=8.; back_load<=8.; back_load+=0.5)
//     {//outer loop for testing effect of changing LOAD
//heads_stuck_max=200*(result_max*3+ATP_result_max)*((int)back_load+1);
       //expect increase of motor stalling but not stuck with increase of load

//OR fix load
back_load=0.; //set load to zero

//TIMING LOOP - cycle through all combinations of timing parameters in ranges set
/*     for (t_ADPrel=0; t_ADPrel < result_max; t_ADPrel++)
     for (t_ATPbind=0; t_ATPbind < ATP_result_max; t_ATPbind++)
     for (t_ATPhyd=0; t_ATPhyd < result_max; t_ATPhyd++)
     for (t_Prel=0; t_Prel < result_max; t_Prel++)
     {TIMING loop
*/

//OR fix timings to known processive values
t_ADPrel=0;
t_ATPbind=0;
t_ATPhyd=0;
t_Prel=1;

//-----------------------
//END ALTERNATE CODE
//-----------------------

     //initialise variables used across runs
     av_steps=0;
     av_backsteps=0;
     av_detachments=0;
```

```
av_time=0;
timing_loop_count=0;

for (run=1; run<=5; run++)
//do several runs with same parameters so take average for data point
  {//RUN loop
  int time=0; //count the time slices i.e. elapsed time, for this run
  int MT_top; //vertical position of MT
  int step_count=0, backstep_count=0;
    //counters for number of steps in interrupted procession
  bool firststep=false, firstbackstep=false;
  //flags to indicate that first step taken

  uucount=0;//initialise tally of successive u.u states
  // - see trace_heads() in analysis.h
  timing_loop_count++;
  //count of time elapsed over all runs for this timing combo

  step=false; //step hasn't been taken
  backstep=false; //backstep hasn't been taken
  heads_bound_count=0; //both heads bound counter initialisation

  //variables for LOAD experiments
  steps=0; //initialise forward step counter
  backsteps=0; //initialise backstep counter
  detachment_count=0; //initialise detachment counter

  //Clear cytosol array
  for (int i=0; i<=row_max; i++) for (int j=0; j<=col_max; j++)
    cytosol[i][j]=0;

  //put in single MT filament along bottom of cytosol box
  MT_top=row_max; generate_MT(MT_top, 1 ,col_max-1);

  //OPTIONAL: put blockage in path of motor
  //put in 2 blob block to stop kinesin accessing binding site
  //generate_block(MT_top,col_max-13);//towards RHS of MT

  //initialise motor/s (code ready for multiple motors)
  for (int motor=0; motor<last_motor; motor++)
    {
    //get pointers to motor heads
    struct head_struct *head1=&motor_array[motor].heada,
      *head2=&motor_array[motor].headb;

    //position heads close together
    head2->posx=motor+3; //near LHS of MT
    head1->posx=head2->posx-1; //place head to the left of partner
    head1->posy=MT_top-3; //and above MT
    head2->posy=MT_top-4; //place second head below first

    //initialise previous position stores to current position
    head2->prev_posx=head2->posx;
    head1->prev_posx=head1->posx;
    head1->prev_posy=head1->posy;
    head2->prev_posy=head2->posy;

    //reset counters for each head
    head1->hydrolysis_count=0;          //timing counter for ATP hydrolysis
    head1->P_release_count=0;           //timing counter for phosphate release
    head1->MT_binding_count=0; //timing counter for MT binding
    head1->ADP_release_count=0; //timing counter for ADP release
    head1->ATP_binding_count=0; //timing counter for ATP binding

    head2->hydrolysis_count=0;          //timing counter for ATP hydrolysis
    head2->P_release_count=0;           //timing counter for phosphate release
    head2->MT_binding_count=0; //timing counter for MT binding
    head2->ADP_release_count=0; //timing counter for ADP release
    head2->ATP_binding_count=0;  //timing counter for ATP binding
```

```
  //both heads start free of MT and ADP bound
  head2->MTbinding=k_free;
  head2->nuc_binding=ADP;
  head1->MTbinding=k_free;
  head1->nuc_binding=ADP;
  }

InitDisplay(hdc); //initialise MS Windows screen display
display_paras(); //display parameters in status bar

do
  {//INNER LOOP - traversed until motor stuck or reaches RHS of MT
  //***Needs modification for several motor experiments
  for (int motor=0; motor<last_motor; motor++)
    {
    struct head_struct*head1=&motor_array[motor].heada,
      *head2=&motor_array[motor].headb;
    //pointers to current motor heads

    int head1_posx=head1->posx,head2_posx=head2->posx;
    //save current head positions

    //calculate next head states, exit loop if error
    if (update_heads(hdc,head1,head2)==false)
      {
      UpdateStatusBar("head update failed", 0, 0);
      fprintf(f,"\n***head update failed\n");
      break;
      }

    if (trace) trace_heads(f,time,head1,head2);

    if (step) //update_heads() has indicated forward step
      {
      steps++; //total forward steps
      step_count++; //forward steps for continuous run
      step=false; //reset flag
      }
    if (backstep) //update_heads() has indicated backward step
      {
      backsteps++; //total backward steps
      backstep_count++; //backsteps for continuous run
      backstep=false; //reset flag
      }

    //update display of system
    for (int motor=0; motor<last_motor; motor++)
      //display each motor in turn
display(hdc,&motor_array[motor].heada,&motor_array[motor].headb);

    time++; //increment system time
    }

  //if blockage timed out then remove
  if (blockage_count > blockage_limit)
    {//blockage_count incremented by forward_step() when clash()
    //(see update_heads.h)
    remove_block(MT_top,col_max-13, hdc);
    blockage_count=0;
    }

  } while (motor_in_play() &&heads_bound_count < heads_stuck_max);
  //continue while motor still moving
  //end of INNER LOOP

for (int motor=0; motor<last_motor; motor++)
  display(hdc, &motor_array[motor].heada, &motor_array[motor].headb);
//display each motor in turn
```

```
        //***Following code works for single motor experiments only
        if (detachment_count > 0)
          {//motor has detached from MT so diffusing
          UpdateStatusBar("Cycle failure/interrupted", 0, 0);
          fprintf(f,"Cycle failure/interrupted.......\t");
          results[t_ATPhyd][t_Prel][t_ADPrel][t_ATPbind]=P_diffusion;
          //display outcome for this set of timing parameters
          display_result(hdc,t_ATPhyd,t_Prel,t_ADPrel, t_ATPbind, t_MTbind,
          50, 50, 150);
          }
        else
        if (heads_bound_count >= heads_stuck_max)
          {//motor has frozen on MT i.e. stuck
          UpdateStatusBar("Cycle failure heads stuck", 0, 0);
          fprintf(f,"Cycle failure: heads stuck......\t");
          results[t_ATPhyd][t_Prel][t_ADPrel][t_ATPbind]=P_stuck;
          display_result(hdc,t_ATPhyd,t_Prel,t_ADPrel,t_ATPbind,t_MTbind,
          220,0,0);
          }
        else
          {//motor has walked to end of MT
          UpdateStatusBar("Cycle success", 0, 0);
          fprintf(f,"Cycle success (both heads bound %d)\t",heads_bound_count);
          results[t_ATPhyd][t_Prel][t_ADPrel][t_ATPbind]=P_processive;
          display_result(hdc,t_ATPhyd, t_Prel,
            t_ADPrel, t_ATPbind, t_MTbind, 0, 220, 0);
          }

        //output to file LOAD results for this set of parameters
        fprintf(f,"Dr%d Tb%d Th%d Pr%d load -%.1f linker %.2f time %4d\n",
        t_ADPrel, t_ATPbind, t_ATPhyd, t_Prel, back_load, linker_tension, time);
        fprintf(f,"steps %3d, backsteps %3d, net %3d\n", steps, backsteps, steps-backsteps);

        av_steps+=steps; //accumulate total steps
        av_backsteps+=backsteps; //accumulate total backsteps
        av_detachments+=detachment_count; //accumulate total detachments
        av_time+=time; //time accumulated - rough indicator of motor progress

        }//end RUN loop

//output to analysis file LOAD results for this set of parameters
//analyse_results(fa, av_time/timing_loop_count, av_steps/timing_loop_count,
//     av_backsteps/timing_loop_count, av_detachments/timing_loop_count);
    }//end TIMING loop

//  }//end LOAD or LINKER loop

  UpdateStatusBar("Finished", 0, 0);
  fprintf(f,"\n===mp8 finished.\n");
  fclose(f); //close data output file
  fprintf(fa,"\n===mp8 finished.\n");
  fclose(fa); //close analysis file
  EndPaint(hwnd,&ps); //finish with MS Windows
  return 0;
  }//mp8

//END of program
```

B.2 Main listing – single motor program for chapter 4 results

```
/*
  Author: Richard Wilson, MOAC DTC, Coventry House, Warwick University, CV4 7AL, UK
      richard.j.wilson@warwick.ac.uk

  This version incorporates more realistic modelling of ATP molecule arrival according to a Poisson
distribution.
  Run now terminates on detachment of motor.

  Purpose of this version:
    To investigate the effect of varying [ATP] on the motion of the molecular motor kinesin.

  Research questions:
    Is power stroke or RBM stepping more realistic model of kinesin?
    How does the motor react to load under variable [ATP]?

*/

#include <stdio.h>
#include <stdbool.h>
#include <math.h>

//GLOBAL DECLARATIONS

//simulation run parameters
bool ATP_gate=true;//*/false;  //ATP-hydrolysis or ATP-binding gate switch - see update_heads.h
//int gating_count; //number of times gating operates
bool RBM=/*true;//*/false; //true if rectified Brownian model (else power stroke model) -
impacts update_heads.h
bool AMP_PNP=false; //indicates whether non-hydrolysable analogue is bound to a head - see
update_heads.h

float linker_tension = 9.5; //linker tension set to give percentage of back steps determined from
experiment
float linker_tension_max=10.0; //maximum value of linker tension
//at this value of linker_tension, binding of free head prevented when kinesin in wait state
(K0.KDu)

float back_load; //hindering load applied to motor
const float stall_load=6.;//notional load that defeats zippering (see zippering routine in
update_heads.h)
const float load_variation=3.;//extent of random variation of load (see zippering routine in
update_heads.h)
//simulates dynamic load variation expected through stalk springiness
//equates to widening range of loads affecting zippering from
//stall_load-load_variation/2 to stall_load+load_variation/2
//so below this range no effect and above it no zippering

const int last_motor=1;//number of motors simulated

//kinesin head state
struct head_struct
  {
  int MTbinding;   //whether kinesin head bound to MT or free
  int nuc_binding;          //which nucleotide is bound to kinesin, if any

  int ATP_binding_count;  //timing counter for ATP binding (K0 -> KT)
  int hydrolysis_count;     //timing counter for ATP hydrolysis (KT -> KDP)
  int P_release_count;      //timing counter for phosphate and head release (KDP -> KDu)
  int MT_binding_count;     //timing counter for MT binding (KDu -> KD)
```

```
  int ADP_release_count;  //timing counter for ADP release (KD -> K0)

  int prev_posx; //last x pos
  int prev_posy; //last y pos
  int posx;         //current x position of head
  int posy;         //current y position of head
  };

//kinesin motor has 2 heads
struct motor_struct
  {
  struct head_struct heada;
  struct head_struct headb;
  };

//array of motors for multiple motor experiments
struct motor_struct motor_array[last_motor];

//head state values
const int k_free=2, k_bound=1; //signifies whether kinesin bound to MT or not
enum {null,ATP,ADPP,ADP}; //signifies which nucleotide is attached, if any

//rectangular array representing section of cytosol
const int row_max=15, col_max=256;
int cytosol[row_max+1][col_max+1];
//NB indexing is positive from top left (cytosol[0][0])
const int cytosol_rgb=230; //pale grey for empty cytosol box used in display.h

//markers for contents of cytosol are all negative (as used by clash routine)
const int MT_null=-99; //interior of MT
const int MT_alpha=-98, MT_beta=-97; //microtubule alpha and beta tubulin
const int head_display=-1; //kinesin head

const int head_prev=1; //represents where heads have been so a track of kinesin motion can be
displayed

//timing parameter delays - used to trigger event when head counter reaches value
//0 means no delay: the event occurs at the next simulation time, 1 means event occurs one
simulation time later...
// values used to accommodate range of [ATP] concentrations
//approximating Rosenfeld et al. 2002 using t_MT_binding as baseline
const int t_ATPhydrolysis = 8; //KT -> KDP
const int t_P_release = 13; //KDP -> KDu
const int t_ADP_release = 7; //KD -> K0
const int t_MT_binding = 1; //KDu -> KD
//
int t_ATP_binding; //K0 -> KT, depends on [ATP]

int ATP_count;//count of ATPs hydrolysed

int blockage_count=0;
//blockage timer - used to time how long blockage placed in the way of kinesin
const int blockage_limit=7; //number of time slices blockage in place

//trace and analysis declarations
bool trace=/*true;//*/false; //switch trace of kinesin movement on or off
int uucount; //tally of successive KDu.KDu states - used in trace_heads() in analysis.h

//flags used in update_heads.h to indicate if step taken etc.
bool step; //set by forward_step()
bool backstep; //set by back_step()
bool motor_detached;//flags that motor has detached
bool both_heads_bound;//set when both heads boud to MT
bool futile;
//set when head already in position when try to step: ATP hydrolysed when no step taken
bool initial_diffusion; //flag to indicate that motor has yet to engage with MT
bool motor_in_play=false;//flags whether motor at end of MT or not

int runs=100; //number of duplicate runs i.e. with same parameters
```

```
//-----------------------------------------------
//FUNCTION DECLARATIONS AND DEFINITIONS
//-----------------------------------------------

//display and trace functions
#include "display.h"

//Brownian motion routines
#include "Brownian.h"

//head update routines
#include "update_heads.h"

int two_power(int x)
  {//calculate 2^x, assumes x is positive integer
  int pot=1;

  for (int i=0;i<x;i++) pot*=2;
  return pot;
  }

//function to simulate random arrival of ATP
bool calc_ATP_delay(float L)
  {
  //L is negative exponential of nominal_ATP_delay (the nominal delay before the next molecule
of ATP arrives)
    //this function calculates a number according to the corresponding Poisson distribution
  float p = get_rand();//get random number between 0 and 1
  int k=0;

  if (1.0 < L < 0.0)
    {
    t_ATP_binding = 0;return false;
    }
  for (;p > L; k++) p = p * get_rand();

  t_ATP_binding = k;
//used in update_head_nuc() in update_heads.h to determine when ATP binds the motor
  return true;
  }

bool motor_not_reached_RHS(void) //test if motor has reached RHS of MT
  {
  struct head_struct *head1=&motor_array[0].heada,*head2=&motor_array[0].headb;

  motor_in_play = head1->posx < col_max-1 && head2->posx < col_max-1;
  //set flag to true if kinesin has yet to reach rightmost end of MT
  return motor_in_play; //flag used in main()
  }

//-------------------------
//START OF MAIN ROUTINE
//-------------------------
int mp8c(HWND hwnd)
  {
  PAINTSTRUCT ps;
  HDC hdc = BeginPaint(hwnd,&ps); //MS Windows specific routine
  FILE *f;

  if (ATP_gate) f = fopen(RBM?"mp8cRBMg.txt":"mp8cPSg.txt","a");
  else f = fopen(RBM?"mp8cRBM.txt":"mp8cPS.txt","a");
    //open file for output data
    //file name distinguishes stepping mechanism in use
  if (f == 0)
    {
     UpdateStatusBar("File open failed", 0, 0); //notify user if file open failure
    EndPaint(hwnd,&ps);
    return false; //exit program
```

```
    }
    //if fail return error

  //write header to file
  fprintf(f,"===mp8c started===\n");
  fprintf(f,"ADP release %d ATP hydrolysis %d P release %d Load variation %.1f Stall %.1f MT
binding %d %s with %s\n\n",
    t_ADP_release,t_ATPhydrolysis,t_P_release,load_variation,stall_load, t_MT_binding,
    (RBM?"RBM":"PS"),(ATP_gate?"ATP gate":"no ATP gate"));

  float v_m = 700.0 * (float)(t_ADP_release + t_ATPhydrolysis + t_P_release);
    //velocity multiplier to scale velocity calculation
  int nominal_ATP_delay; //for calculating ATP delay for each hydrolysis cycle

//LOAD LOOP
  for (back_load = 0.;back_load <= 8.;back_load++)
    {//outer loop for testing effect of changing LOAD
    fprintf(f,"LOAD %.1f\n",back_load);

  for (int ATP_base = 0;ATP_base < 7;ATP_base++)
    {//ATP concentration variation loop
    //count number of runs resulting in each category:
    int procession=0;
    int stuck=0;
    int diffusion=0;
    float lambda;

    //variables with which to calculate average values over several runs
    float av_steps = 0.0;
    float av_backsteps = 0.0;
    float av_futiles = 0.0;
    float av_time = 0.0;
    float av_ATPs = 0.0;//average number of ATPs hydrolysed
    float av_ATP_delay = 0.0;//average ATP arrival delay
    float av_v =0.0;

    nominal_ATP_delay = two_power(ATP_base) - 1;
    //start at 0 (equivalent to max [ATP] i.e. no delay in ATP arrival) and
    //increase nominal ATP delay by power sequence instead of incrementing
    // to give wide spread of values corresponding to wide spread of [ATP]

    lambda = expf(-(float)(nominal_ATP_delay));//calculate exponential for Poisson routine

  for (int run=1; run<=runs; run++)
      //do several runs with same parameters so can take average of raw data for data point
      {//RUN loop
      int time = 0; //elapsed time used for each run
      int time_max = 500*(t_ADP_release + t_ATPhydrolysis + t_P_release)*(ATP_base+1);
        //time limit beyond which assume motor stuck
      int MT_top = row_max; //vertical position of MT (at bottom of cytosol box)
      int steps = 0; //number of steps motor takes
      int backsteps = 0; //number of back steps
      int futiles = 0;//number of futile ATP hydrolyses i.e. ATP hydrolysed by motor doesn't move
      int minATPd = -1;     //minimum ATP delay of run
      int maxATPd = 0;      //maximum ATP delay of run
      float totATPd = 0.0; //total of ATP delays produced by Poisson function over run
      float avATPd = 0.0; //average ATP delay over run
      int head1_posx = 0;
      int head2_posx = 0;
      float v = 0.0;//velocity over run
      initial_diffusion = true; //motor starts off diffusing so not engaged with MT
      uucount = 0; //initialise tally of successive u.u states - see trace_heads() in analysis.h

      //initialise flags and variables used by functions in update_heads.h
      step = false; //step hasn't been taken
      backstep = false; //backstep hasn't been taken
      ATP_count = 0;//no ATPs hydrolysed

      //Clear cytosol array
```

```
for (int i=0; i<=row_max; i++) for (int j=0; j<=col_max; j++) cytosol[i][j]=0;

//put in single MT filament
generate_MT(MT_top,0,col_max-1);

//OPTIONAL: put blockage in path of motor
//put in 2 blob block to stop kinesin accessing binding site
//generate_block(MT_top,col_max-13);

//initialise motor/s (code ready for multiple motors)
for (int motor=0; motor<last_motor; motor++)
  {
  //get pointers to motor heads
  struct head_struct
  *head1=&motor_array[motor].heada,*head2=&motor_array[motor].headb;

  //set up heads close together
  //                              head2->posx=motor+3; //near LHS of MT
  head2->posx=motor+col_max/4;
  //as loading motor, start about quarter of the way along MT
  head1->posx=head2->posx-1; //place head to the left of the other
  head1->posy=MT_top-3; //and above MT
  head2->posy=MT_top-4; //place second head below first

  //initialise previous position to current position
  head2->prev_posx=head2->posx;
  head1->prev_posx=head1->posx;
  head1->prev_posy=head1->posy;
  head2->prev_posy=head2->posy;

  //initialise counters for each head

  head1->hydrolysis_count=0;          //timing counter for ATP hydrolysis
  head1->P_release_count=0;           //timing counter for phosphate release
  head1->MT_binding_count=0; //timing counter for MT binding
  head1->ADP_release_count=0; //timing counter for ADP release
  head1->ATP_binding_count=0; //timing counter for ATP binding

  head2->hydrolysis_count=0;          //timing counter for ATP hydrolysis
  head2->P_release_count=0;           //timing counter for phosphate release
  head2->MT_binding_count=0; //timing counter for MT binding
  head2->ADP_release_count=0; //timing counter for ADP release
  head2->ATP_binding_count=0;  //timing counter for ATP binding

  //both heads start free of MT and ADP bound
  head2->MTbinding=k_free;
  head2->nuc_binding=ADP;
  head1->MTbinding=k_free;
  head1->nuc_binding=ADP;
  }

InitDisplay(hdc); //initialise MS Windows screen display
display_paras(' '); //display parameters in status bar

//calculate ATP molecule arrival delay modelled as Poisson process
calc_ATP_delay(lambda);
if (minATPd == -1 || minATPd > t_ATP_binding) minATPd = t_ATP_binding;
//record minimum value
if (maxATPd < t_ATP_binding) maxATPd = t_ATP_binding; //record maximum value
totATPd += (float)t_ATP_binding; //accumulate so can take average over the run

do
  {//INNER LOOP - traversed until motor stuck or detaches or reaches RHS of cytosol box
  struct head_struct*head1=&motor_array[0].heada,*head2=&motor_array[0].headb;
  //pointers to current motor heads

  do {
    //if motor processing then update() checks for binding and hydrolysis events
    // and updates state of motor appropriately;
```

```
  if (update(hdc,head1,head2) == false)
    {// exit loop if error
    UpdateStatusBar("head update failed", 0, 0);
    fprintf(f,"\n***head update failed*** T %d h1 %d x %d y %d h2 %d x %d y %d\n",
    time,head1->nuc_binding,head1->posx,head1->posy,head2->nuc_binding,
            head2->posx,head2->posy);
    motor_in_play = false;
    break;
    }
  update_cytosol(hdc,head1,head2);
  } while (motor_detached && initial_diffusion && motor_not_reached_RHS());
   //repeat until motor engaged or has reached RHS of cytosol
  //it engages with MT or reaches RHS of cytosol box or there's an error

if (motor_detached) motor_in_play = false;//motor has detached so no longer in play

if (initial_diffusion == false && head1_posx == head1->posx &&
  head2_posx == head2->posx)
//heads haven't moved along MT since last hydrolysis
  {
  if (step || backstep)
    {
    step=false;backstep=false;futile=true;
    //not a real step: free head has re-bound to same binding site on MT
    }
  }

if (trace) trace_heads(f,time,head1,head2);

if (step || backstep || futile)
  {
  initial_diffusion = false; //initial diffusion has finished
    //calculate next ATP molecule arrival delay modelled as Poisson process
  calc_ATP_delay(lambda);
  if (minATPd == -1 || minATPd > t_ATP_binding) minATPd = t_ATP_binding;
  //record minimum value
  if (maxATPd < t_ATP_binding) maxATPd = t_ATP_binding; //record maximum value
  totATPd += (float)t_ATP_binding; //accumulate for calculation of average
  head1_posx = head1->posx;
  head2_posx = head2->posx;
  //save current head positions
  }

if (step) //forward step has been taken
  {
  steps++; //total forward steps
  step=false; //reset flag
  display_paras('>'); //forward step indicated in status bar
  }
else if (backstep) //backward step has been taken
  {
  backsteps++; //total backward steps
  backstep=false; //reset flag
  display_paras('<'); //backstep indicated in status bar
  }
else if (futile)//no step but hydrolysis
  {
  futile=false;//reset flag
  futiles++; //total futile hydrolyses
  display_paras('F'); //futile hydrolysis indicated in status bar
  }

  /*//update display of system to show blow-by-blow motor progress
  //NB significantly decreases simulation speed
  for (int motor=0; motor<last_motor; motor++)
    //display each motor in turn
    display(hdc,&motor_array[motor].heada,&motor_array[motor].headb);
  //*/
time++;//increment system time
```

```
///*
        //if blockage set and timed out then remove
        if (blockage_count > blockage_limit)
          {//blockage_count incremented by forward_step() when clash()
          //see update_heads.h
          remove_block(MT_top,col_max-13, hdc);
          blockage_count=0;
          }
//*/
        } while (motor_in_play && time > 0 && time <= time_max);
        //continue while motor not at end of MT or detached and not stuck
        //end of INNER LOOP

      //display end state of run
      for (int motor=0; motor<last_motor; motor++)
        display(hdc, &motor_array[motor].heada, &motor_array[motor].headb);

      //analyse what motor has done
      if (time >= time_max)
        {//motor has timed out - assume stuck
        UpdateStatusBar("Motor stuck", 0, 0);
        display_result(hdc,t_ATPhydrolysis,t_P_release,t_ADP_release,ATP_base,run,240,0,0);
        stuck++;
        }
      else
      if (steps + backsteps < 2)
        {//motor hasn't processed (needs at least 2 steps): assume no backward procession
        UpdateStatusBar("Motor detached", 0, 0);
  display_result(hdc,t_ATPhydrolysis,t_P_release,t_ADP_release,ATP_base,run,0,0,240);
        diffusion++;
        }
      else
        {//motor has processed
        UpdateStatusBar("Procession", 0, 0);
        display_result(hdc,t_ATPhydrolysis,t_P_release,t_ADP_release,ATP_base,run,0,240,0);
        procession++;
        }
      //calculate results for run
      v = (steps + backsteps < 2)?0.0:v_m * (float)(steps - backsteps)/(float)time;
      avATPd = totATPd/(float)ATP_count; //average ATP delay over total number of hydrolyses
completed in run
/*
      //output this run results to file
      fprintf(f,"Run %2d Steps forward %4d back %2d ATPs %4d Futiles (calc: %3d) %3d Time
%5d V %0.3f ",
          run,steps,backsteps,ATP_count,ATP_count-steps-backsteps,futiles,time,v);
      fprintf(f,"ATP delay av %.1f range %d - %d\n",avATPd,minATPd,maxATPd);
//*/

      //accumulate totals over all runs
      av_time += (float)time; //run times
      av_steps += (float)steps; //total steps
      av_backsteps += (float)backsteps; //total backsteps
      av_futiles += (float)futiles;
      av_ATP_delay += avATPd;//average ATP arrival delay
      av_ATPs += (float)ATP_count;
      av_v += v;//accumulate velocity
      //NB variables used per run are reset at top of run loop
      }//end RUN loop

    //calculate averages over runs and output to file
    av_time /= (float)runs;
    av_steps /= (float)runs;
    av_backsteps /= (float)runs;
    av_futiles /= (float)runs;
    av_ATP_delay /= (float)runs;
    av_ATPs /= (float)runs;
    av_v /= (float)runs;
```

```
    fprintf(f,"Poisson i/p %d lambda %f Av ATP delay %.1f Procession %d stuck %d diffusion
%d\n",
       nominal_ATP_delay,lambda,av_ATP_delay,procession,stuck,diffusion);
    fprintf(f,"%d run av: Steps %.1f Backsteps %.1f ATPs % .1f Futiles %.1f Time %.0f V %.2f
Dwell %.1f\n",

  runs,av_steps,av_backsteps,av_ATPs,av_futiles,av_time,av_v,av_time/(av_steps+av_backstep
s));

    }//end concentration variation loop
    }//end LOAD loop

  UpdateStatusBar("Finished", 0, 0);
  fprintf(f,"===mp8 finished===\n\n");
  fclose(f); //close data output file
  EndPaint(hwnd,&ps); //finish with MS Windows
  return 0;
  }//mp8c

//END of program
```

B.3 Main listing – multimotor program

```
/*
  Author: Richard Wilson, MOAC DTC, Coventry House, Warwick University, CV4 7AL, UK
      richard.j.wilson@warwick.ac.uk

Multimotor Program: this version devised to investigate 2 motors coupled by flexible link

  This version incorporates the Poisson ATP arrival distribution
  Run terminates if both motors detached but not if only one detaches
  Motors started in contact with MT to get consistent start position for each run
  Velocity derived from path length rather than stepping count (which only works for single
motor)

Purpose of this version:
    To investigate the effect of linking 2 motors

  Research questions:
    Does second motor coordinate with first analogous to two heads coordinating when bound to
same cargo?
    Measure by comparing velocity and run length behaviour of 1 motor to 2 motor system.
*/

#include <stdio.h>
#include <stdbool.h>
#include <math.h>

//GLOBAL DECLARATIONS

//simulation run parameters
bool ATP_gate=/*true;//*/false;  //ATP-hydrolysis or ATP-binding gate switch - see
update_heads.h
//int gating_count; //number of times gating operates
bool RBM=true;//*/false; //true if rectified Brownian model (else power stroke model) - impacts
update_heads.h
bool AMP_PNP=false; //indicates whether non-hydrolysable analogue is bound to a head - see
update_heads.h

float linker_tension = 9.5; //linker tension set to give percentage of back steps determined from
experiment
float linker_tension_max=10.0; //maximum value of linker tension
//at this value of linker_tension, binding of free head prevented when kinesin in wait state
(K0.KDu)

float back_load; //hindering load applied to motor
const float stall_load=6.;//notional load that defeats zippering (see zippering routine in
update_heads.h)
const float load_variation=3.;//extent of random variation of load (see zippering routine in
update_heads.h)
//simulates dynamic load variation expected through stalk springiness
//equates to widening range of loads affecting zippering from
//stall_load-load_variation/2 to stall_load+load_variation/2
//so below this range no effect and above it no zippering

const int last_motor=2;//number of motors simulated

//kinesin head state
struct head_struct
  {
  int MTbinding;  //whether kinesin head bound to MT or free
  int nuc_binding;          //which nucleotide is bound to kinesin, if any

  int ATP_binding_count;  //timing counter for ATP binding (K0 -> KT)
  int hydrolysis_count;     //timing counter for ATP hydrolysis (KT -> KDP)
  int P_release_count;      //timing counter for phosphate and head release (KDP -> KDu)
```

```
  int MT_binding_count;    //timing counter for MT binding (KDu -> KD)
  int ADP_release_count;  //timing counter for ADP release (KD -> K0)

  int prev_posx; //last x pos
  int prev_posy; //last y pos
  int posx;         //current x position of head
  int posy;         //current y position of head
  };

//kinesin motor has 2 heads
struct motor_struct
  {
  struct head_struct heada;
  struct head_struct headb;
  };

//array of motors for multiple motor experiments
struct motor_struct motor_array[last_motor];

//head state values
const int k_free=2, k_bound=1; //signifies whether kinesin bound to MT or not
enum {null,ATP,ADPP,ADP}; //signifies which nucleotide is attached, if any

//rectangular array representing section of cytosol
const int row_max=15, col_max=256;
int cytosol[row_max+1][col_max+1];
//NB indexing is positive from top left (cytosol[0][0])
const int cytosol_rgb=230; //pale grey for empty cytosol box used in display.h

//markers for contents of cytosol are all negative (as used by clash routine)
const int MT_null=-99; //interior of MT
const int MT_alpha=-98, MT_beta=-97; //microtubule alpha and beta tubulin
const int head_display=-1; //kinesin head

const int head_prev=1; //represents where heads have been so a track of kinesin motion can be
displayed

//timing parameter delays - used to trigger event when head counter reaches value
//0 means no delay: the event occurs at the next simulation time, 1 means event occurs one
simulation time later...
///* values used to accommodate range of [ATP] concentrations
//approximating Rosenfeld et al 2002 using t_MT_binding as baseline
const int t_ATPhydrolysis = 8; //KT -> KDP
const int t_P_release = 13; //KDP -> KDu
const int t_ADP_release = 7; //KD -> K0
const int t_MT_binding = 1; //KDu -> KD

int t_ATP_binding; //K0 -> KT, depends on [ATP]

int ATP_count;//count of ATPs hydrolysed

int blockage_count=0; //blockage timer - used to time how long blockage placed in the way of
kinesin
const int blockage_limit=7; //number of time slices blockage in place

//trace and analysis declarations
bool trace=/*true;//*/false; //switch trace of kinesin movement on or off
int uucount; //tally of successive KDu.KDu states - used in trace_heads() in analysis.h

//flags used in update_heads.h to indicate if step taken etc.
bool step; //set by forward_step()
bool backstep; //set by back_step()
bool motor_detached;//flags motor detachment
bool both_heads_bound;//set when both heads bound to MT

bool futile;//set when head already in position when try to step: ATP hydrolysed when no step
taken
bool motor_has_detached[last_motor];//flags whether motor processing
```

```
int runs=100;//number of duplicate runs i.e. with same parameters

const int motor_distance = 15;//default distance between motors

//-----------------------------------------------
//FUNCTION DECLARATIONS AND DEFINITIONS
//-----------------------------------------------

//display and trace functions
#include "display.h"

//Brownian motion routines
#include "Brownian.h"

//head update routines
#include "update_heads.h"

int two_power(int x)
  {//calculate 2^x, assumes x is positive integer
  int pot=1;

  for (int i=0;i<x;i++) pot*=2;
  return pot;
  }

//function to simulate random arrival of ATP
bool calc_ATP_delay(float L)
  {
  //L is negative exponential of nominal_ATP_delay  (the nominal delay before the next molecule
of ATP arrives)
    //this function calculates a number according to the corresponding Poisson distribution
  float p = get_rand();//get random number between 0 and 1
  int k=0;

  if (1.0 < L < 0.0)
    {
    t_ATP_binding = 0;return false;
    }
  for (;p > L; k++) p = p * get_rand();

  t_ATP_binding = k;
//used in update_head_nuc() in update_heads.h to determine when ATP binds the motor
  return true;
  }

int fn_m_distance(int motor)
  {//returns load value for leading motor increasing with distance between motors greater than
motor_distance
    //or load value for trailing motor decreasing with distance when too close

  //get pointers to motor heads
  struct head_struct
    *head1=&motor_array[motor].heada,
    *head2=&motor_array[motor].headb;
  int other_motor = (motor == 0)?1:0; //***assumes 2 motors
  struct head_struct
    *other_head1=&motor_array[other_motor].heada,
    *other_head2=&motor_array[other_motor].headb;

  int motor_pos = (head1->posx + head2->posx)/2;
  int other_motor_pos = (other_head1->posx + other_head2->posx)/2;
  int m_x = motor_pos - other_motor_pos;
  if (m_x < 0) //this is the trailing motor
    {
    m_x = -m_x;//absolute value
    if (m_x >= motor_distance) return 0;//motors at normal distance or greater
    else return (motor_distance - m_x);//load increases on trailing motor as motors get closer
    }
  //this is the leading motor
```

```
  if (m_x <= motor_distance) return 0;//motors at normal distance or less
  else return (m_x - motor_distance);//load increases as motors moves away from each other
  }

bool any_motors_in_play(void)//if all motors detached then returns false
  {
  for (int motor = 0; motor < last_motor;motor++)
    if (motor_has_detached[motor] == false) return true;
  return false;
  }

bool all_motors_in_play(void)//if any motors detached then returns false
  {
  for (int motor = 0; motor < last_motor;motor++) if (motor_has_detached[motor]) return false;
  return true;
  }

//-------------------------
//START OF MAIN ROUTINE
//-------------------------
int mp1(HWND hwnd)
  {
  PAINTSTRUCT ps;
  HDC hdc = BeginPaint(hwnd,&ps); //MS Windows specific routine
  FILE *f;

  if (ATP_gate) f = fopen(RBM?"mp1cRBMg.txt":"mp1cPSg.txt","a");
  else f = fopen(RBM?"mp1cRBM.txt":"mp1cPS.txt","a");
    //open file for output data
    //file name distinguishes stepping mechanism in use
  if (f == 0)
    {
     UpdateStatusBar("File open failed", 0, 0); //notify user if file open failure
    EndPaint(hwnd,&ps);
    return false; //exit program
    }
    //if fail return error
  //write header to file
  fprintf(f,"===mp1c started=== %d motor/s\n",last_motor);
  fprintf(f,"ADP release %d ATP hydrolysis %d P release %d Load variation %.1f Stall %.1f MT
binding %d %s with %s\n\n",
    t_ADP_release,t_ATPhydrolysis,t_P_release,load_variation,stall_load, t_MT_binding,
    (RBM?"RBM":"PS"),(ATP_gate?"ATP gate":"no ATP gate"));

  float v_m = 700.0 * (float)(t_ADP_release + t_ATPhydrolysis + t_P_release);
    //velocity multiplier to scale velocity calculation
  int nominal_ATP_delay; //for calculating ATP delay for each hydrolysis cycle

  for (int ATP_base = 0;ATP_base < 1;ATP_base++)//*****
    {//ATP concentration variation loop
    //count number of runs resulting in each category:
    int procession=0;
    int stuck=0;
    int diffusion=0;
    float lambda;
    //variables with which to calculate average values over several runs
    float av_steps = 0.0;
    float av_backsteps = 0.0;
    float av_futiles = 0.0;
    float av_time = 0.0;
    float av_ATPs = 0.0;//average number of ATPs hydrolysed
    float av_ATP_delay = 0.0;//average ATP arrival delay
    float av_runlength = 0.0;
    int min_runlength = -1;
    int max_runlength = 0;

    nominal_ATP_delay = two_power(ATP_base) - 1;
    //start at 0 (equivalent to max [ATP] i.e. no delay in ATP arrival)
```

```
    //increase nominal ATP delay by power sequence instead of incrementing to give wide spread
of values
    //corresponding to wide spread of [ATP]
    lambda = expf(-(float)(nominal_ATP_delay));//calculate exponential for Poisson routine

    for (int run=1; run<=runs; run++)
      //do several runs with same parameters so can take average of raw data for data point
      {//RUN loop
      int time = 0; //elapsed time used for each run
      int time_max = 500*(t_ADP_release + t_ATPhydrolysis + t_P_release)*(ATP_base+1);
        //time limit beyond which assume motor stuck
      int MT_top = row_max; //vertical position of MT (at bottom of cytosol box)
      int steps = 0; //number of steps motor takes
      int backsteps = 0; //number of back steps
      int futiles = 0;//number of futile ATP hydrolyses i.e. ATP hydrolysed by motor doesn't move
      float totATPd = 0.0; //total of ATP delays produced by Poisson function over run
      float avATPd = 0.0; //average ATP delay over run
      int head1_posx = 0;
      int head2_posx = 0;
      float v = 0.0;//velocity over run
      bool no_error = true;//error flag
      bool motor_not_at_RHS = true;
      int runlength = 0;//length motor/s traverse/s in a run
      int mstart[last_motor];//motor start position
      int mfinish[last_motor];//motor finish position
      int cstart = 0;//start position of run
      int cfinish = 0;//finish position of run

      uucount = 0; //initialise tally of successive u.u states - see trace_heads() in analysis.h

      //initialise flags and variables used by functions in update_heads.h
      step = false; //step hasn't been taken
      backstep = false; //backstep hasn't been taken
      ATP_count = 0;//no ATPs hydrolysed
      motor_detached = false;//motor starts attached

      //Clear cytosol array
      for (int i=0; i<=row_max; i++) for (int j=0; j<=col_max; j++) cytosol[i][j]=0;

      //put in single MT filament
      generate_MT(MT_top,0,col_max-1);

      //OPTIONAL: put blockage in path of motor
      //put in 2 blob block to stop kinesin accessing binding site
      //generate_block(MT_top,col_max-13);

      //initialise motor/s (code ready for multiple motors)
      for (int motor = 0; motor < last_motor; motor++)
        {
        //get pointers to motor heads
        struct head_struct
          *head1=&motor_array[motor].heada,
          *head2=&motor_array[motor].headb;

        head2->posx=motor*motor_distance + 4; //start first motor near LHS of MT
        //displace other motors to the right by motor_distance
        head1->posx=head2->posx-1; //place head to the left of the other
        head1->posy=MT_top-1; //and in contact with MT
        head2->posy=MT_top-2; //place second head above first

        //initialise previous position to current position
        head2->prev_posx=head2->posx;
        head1->prev_posx=head1->posx;
        head1->prev_posy=head1->posy;
        head2->prev_posy=head2->posy;

        //initialise counters for each head
        head1->hydrolysis_count=0;          //timing counter for ATP hydrolysis
        head1->P_release_count=0;           //timing counter for phosphate release
```

```
  head1->MT_binding_count=0; //timing counter for MT binding
  head1->ADP_release_count=0; //timing counter for ADP release
  head1->ATP_binding_count=0; //timing counter for ATP binding

  head2->hydrolysis_count=0;          //timing counter for ATP hydrolysis
  head2->P_release_count=0;           //timing counter for phosphate release
  head2->MT_binding_count=0; //timing counter for MT binding
  head2->ADP_release_count=0; //timing counter for ADP release
  head2->ATP_binding_count=0;  //timing counter for ATP binding

  //start with one head free of MT, the other bound; both ADP bound
  head2->MTbinding=k_free;
  head2->nuc_binding=ADP;
  head1->MTbinding=k_bound;
  head1->nuc_binding=ADP;

  mstart[motor] = (head1->posx + head2->posx)/2;//approx motor start position
  }

//calculate approx cargo start position
cstart = mstart[0];//start position for 1 motor
for (int motor = 1;motor < last_motor; motor++) cstart = (mstart[motor] + cstart)/2;
//mid start position for 2 motors

InitDisplay(hdc); //initialise MS Windows screen display
display_paras(' ',0); //display parameters in status bar

//calculate ATP molecule arrival delay modelled as Poisson process
calc_ATP_delay(lambda);
totATPd += (float)t_ATP_binding; //accumulate so can take average over the run

do
  {//INNER LOOP - traversed until motor stuck or detached
  for (int motor = 0;motor < last_motor;motor++)
    {
    struct head_struct
      *head1=&motor_array[motor].heada,
      *head2=&motor_array[motor].headb;
    //pointers to current motor heads

    motor_has_detached[motor] = false;
    //initialise detachment latch for this motor

    if (last_motor > 1 && all_motors_in_play())
      back_load = fn_m_distance(motor);
    //if more than one motor and all motors processing then apply load as required
    display_paras(' ',motor); //display parameters in status bar

    do      {
      //if motor processing then update() checks for binding and hydrolysis events
      // and updates state of motor appropriately;
      if (update(hdc,head1,head2) == false)
            {// exit loop if error
            UpdateStatusBar("head update failed", 0, 0);
            fprintf(f,"\n***head update failed*** T %d h1 %d x %d y %d h2 %d x %d y
%d\n",time,
                  head1->nuc_binding,head1->posx,head1->posy,head2-
>nuc_binding,head2->posx,head2->posy);
            no_error = false;
            break;
            }
      if (motor_detached) motor_has_detached[motor] = motor_detached;
      //this motor has detached latch set
      update_cytosol(hdc,head1,head2);
      if (motor_not_at_RHS)
       motor_not_at_RHS = (head1->posx < col_max-1 && head2->posx < col_max-1);
            //test whether motor reached RHS at each looping
            //flag acts as a latch - if either motor reaches RHS then set to true
      } while (motor_detached && any_motors_in_play() && motor_not_at_RHS);
```

```
                //repeat until motor engaged or hit RHS of cytosol box

          motor_detached = false; //reset flag which may be set in update_heads()

          if (head1_posx == head1->posx && head2_posx == head2->posx)
          //heads haven't moved along MT since last hydrolysis
            {
            if (step || backstep)
                {
                step=false;backstep=false;futile=true;
                //not a real step: free head has re-bound to same binding site on MT
                }
          }

          if (trace) trace_heads(f,time,head1,head2);

          if (step || backstep || futile)
            {
            //calculate next ATP molecule arrival delay modelled as Poisson process
            calc_ATP_delay(lambda);
            totATPd += (float)t_ATP_binding; //accumulate for calculation of average
            head1_posx = head1->posx;
            head2_posx = head2->posx;
                //save current head positions
            }

          if (step) //forward step has been taken
            {
            steps++; //total forward steps
            step=false; //reset flag
            display_paras('>',0); //forward step indicated in status bar
            }
          else if (backstep) //backward step has been taken
            {
            backsteps++; //total backward steps
            backstep=false; //reset flag
            display_paras('<',0); //backstep indicated in status bar
            }
          else if (futile)//no step but hydrolysis
            {
            futile=false;//reset flag
            futiles++; //total futile hydrolyses
            display_paras('F',0); //futile hydrolysis indicated in status bar
            }
          }//end motor loop

          /*//update display of system to show blow-by-blow motor progress
          //NB significantly decreases simulation speed
          for (int motor=0; motor<last_motor; motor++)
            //display each motor in turn
            display(hdc,&motor_array[motor].heada,&motor_array[motor].headb);
          //*/

        time++;//increment system time

///*
        //if blockage set and timed out then remove
        if (blockage_count > blockage_limit)
          {//blockage_count incremented by forward_step() when clash()
          //see update_heads.h
          remove_block(MT_top,col_max-13, hdc);
          blockage_count=0;
          }
//*/
        } while (any_motors_in_play() && time > 0 && time <= time_max && motor_not_at_RHS
&& no_error);
        //continue while motor processing, not at RHS and no error
        //end of INNER LOOP
```

```
      //display end state of run and find finish position
      for (int motor=0; motor<last_motor; motor++)
        {
        display(hdc, &motor_array[motor].heada, &motor_array[motor].headb);
        mfinish[motor] = (motor_array[motor].heada.posx + motor_array[motor].headb.posx)/2;
          //approx motor finish position/s
        }
      //calculate approx cargo finish position
      cfinish = mfinish[0];//finish position for 1 motor
      for (int motor = 1;motor < last_motor; motor++) cfinish = (mfinish[motor] + cfinish)/2;
        //mid finish position for 2 motors
      if (cfinish < 0) cfinish = -cfinish;

      //analyse what motor has done
      if (time >= time_max)
        {//motor has timed out - assume stuck
        UpdateStatusBar("Motor stuck", 0, 0);
        display_result(hdc,t_ATPhydrolysis,t_P_release,t_ADP_release,ATP_base,run,240,0,0);
        stuck++;
        }
      else
      if (steps + backsteps < 2)
        {//motor hasn't processed (needs at least 2 steps)
        UpdateStatusBar("Motor detached", 0, 0);
        display_result(hdc,t_ATPhydrolysis,t_P_release,t_ADP_release,ATP_base,run,0,0,240);
        diffusion++;
        }
      else
        {//motor has processed
        UpdateStatusBar("Procession", 0, 0);
        display_result(hdc,t_ATPhydrolysis,t_P_release,t_ADP_release,ATP_base,run,0,240,0);
        procession++;
        }
      //calculate results for run
      avATPd = totATPd/(float)ATP_count; //average ATP delay over total number of hydrolyses
completed in run
      runlength = ((cfinish - cstart) <= 0)?0:cfinish - cstart;
///*
      //output run results to file
      fprintf(f,"Run %2d L %d (%d to %d) Steps %4d back %2d ATP delay av %.1f T %5d V
%.1f\n",
          run,runlength,cstart,cfinish,steps,backsteps,avATPd,time,
          v_m*(float)runlength/(float)time);
//*/
      if (min_runlength == -1 || runlength < min_runlength) min_runlength = runlength;
      if (runlength > max_runlength) max_runlength = runlength;
      //accumulate totals over all runs
      av_time += (float)time; //run times
      av_steps += (float)steps; //total steps
      av_backsteps += (float)backsteps; //total backsteps
      av_futiles += (float)futiles;
      av_ATP_delay += avATPd;//average ATP arrival delay
      av_ATPs += (float)ATP_count;
      av_runlength += (float)runlength;
      }//end RUN loop

    //calculate averages over runs and output to file
    av_time /= (float)runs;
    av_steps /= (float)runs;
    av_backsteps /= (float)runs;
    av_futiles /= (float)runs;
    av_ATP_delay /= (float)runs;
    av_ATPs /= (float)runs;
    av_runlength /= (float)runs;
    fprintf(f,"Poisson i/p %d Av ATP delay %.1f Procession %d stuck %d diffusion %d\n",
      nominal_ATP_delay,av_ATP_delay,procession,stuck,diffusion);
    fprintf(f,"%d run av: L %.1f (%d to %d) Steps %.1f Backsteps %.1f ATPs % .1f Futiles %.1f
T %.0f V %.1f Dwell %.1f\n",
```

```
  runs,av_runlength,min_runlength,max_runlength,av_steps,av_backsteps,av_ATPs,av_futiles,av
_time,
       v_m*av_runlength/av_time,av_time/(av_steps+av_backsteps));
     }//end concentration variation loop

  UpdateStatusBar("Finished", 0, 0);
  fprintf(f,"===mp1 finished===\n\n");
  fclose(f); //close data output file
  EndPaint(hwnd,&ps); //finish with MS Windows
  return 0;
  }//mp1

//END of program
```

B.4 Update routines

```
//update_heads.h contains head state update routines
//entry routine is update_heads()

//The stepping conditions depend on the setting of the RBM flag:

//RBM
//===
//If RBM flag is true then following rectified Brownian mechanism (Fox and Choi 2001)
//stepping is a diffusive process with bias provided by zippering.
//Binding of ATP sets up zippering (through conformational change in head) which remains until P
released;
//(Asenjo, Weinberg & Sosa 2006)
//so, as soon as free head diffuses forward, the bound head's neck linker zippers
//and free head restrained local to forward binding site so binds i.e. a step is taken.

//If ATP_gate flag set then ATP hydrolysis gate applied
//This is hypothesised to make hydrolysis v slow unless/until free head binds to next site
//i.e. forward neck linker strain required to make hydrolysis efficient
//(Hancock and Howard, 1999 found that about 10 fold diff between
//hydrolysis time for single and twin head kinesin)

//PS
//==
//If RBM flag is false then
//following power stroke mechanism whereby ATP binding causes linker zippering
// which pulls free head forward to near next binding site so it binds and a step is taken
// (Vale and Milligan 2000)

//If ATP_gate flag also set then ATP binding gate applied
//if both heads are bound when the leading head is in K0 state, then the PS gate operates to stop
ATP binding
//until leading head hydrolyses its ATP and detaches
//(proposed as a head coordination mechanism by Rosenberg et al 2002)

bool zipper_active(struct head_struct *head)
  {//determine whether zipper active
  if ((back_load + load_variation*(get_rand()-0.5))>stall_load) return false;
    //if load sufficient to defeat zippering then return false
  if (RBM)
    //RBM model so if ATP or ADPP bound then head set up for linker zippering
    switch (head->nuc_binding)
        {
      case ATP:
      case ADPP: return true;
      default: return false;
      }
  else //power stroke model where ATP binding is the event causing zippering
    switch (head->nuc_binding)
        {
      case ATP: if (head->hydrolysis_count == 0) return true;
      default: return false;
      }
  }

bool update_head_nuc(int head_nuc_binding, struct head_struct *head, struct head_struct
*other_head)
  {//update nucleotide binding to this head
    //procession cycle is KD->K0->KT->KDP->KDu->KD...
    //Note KDP->KDu requires change to MT binding
  switch (head_nuc_binding)         //current nucleotide binding to this head
      {
    case null:
      if (head->MTbinding == k_bound)//always true in present model
```

```
        {
        if (ATP_gate && RBM == false && other_head->MTbinding == k_bound && other_head-
>posx < head->posx)
        //if PS with gating and both heads bound and this is the lead head then ATP gate operates
          head->ATP_binding_count++;
        //so count up amount of time in K0 state i.e. waiting for ATP
        else
          {
          if (head->ATP_binding_count >= t_ATP_binding)
            {
            head->ATP_binding_count=0;
            head->nuc_binding=ATP; //ATP binds because binding delay ended
            }
          else head->ATP_binding_count++;
          //count up amount of time in K0 state i.e. waiting for ATP
          }
        }
      else head->nuc_binding=ADP;
      //else ADP binds (since head is free)
      //this would only happen if head pulled off MT
      break;
    case ADP:
      if (head->MTbinding == k_bound)
        //binding to MT liberates ADP when ADP_release_count reached
        {
        if (head->ADP_release_count >= t_ADP_release)
          {
          head->ADP_release_count=0;
          head->nuc_binding=null;  //ADP released
          }
        else head->ADP_release_count++; //count up amount of time in KD state
        }
      else head->ADP_release_count=0; //count only happens when bound
        //else it's free to diffuse
      break;
    case ADPP:
      if (head->P_release_count >= t_P_release)
        {
        head->P_release_count=0;
        head->nuc_binding=ADP; //phosphate released
        }
      else head->P_release_count++; //count up amount of time in KDP state
      break;
    case ATP:
      if (RBM && ATP_gate && (other_head->MTbinding == k_free || other_head->posx !=
(head->posx+2)))
      //the RBM ATP gate is operating when switch is on and other head not bound to forward
site...
        {
        if (head->hydrolysis_count >= 10*(t_ATPhydrolysis+1))
        //increase effective delay to ATP binding by 10
          {
          head->hydrolysis_count=0;
          head->nuc_binding=ADPP;          //ATP hydrolysed
          ATP_count++; //increment count of number of ATPs hydrolysed
          break;
          }
        }
      else //either PS or RBM without ATP gate so normal hydrolysis
        if (head->hydrolysis_count >= t_ATPhydrolysis)
          {
          head->hydrolysis_count=0;
          head->nuc_binding=ADPP;          //ATP hydrolysed
          ATP_count++; //increment count of number of ATPs hydrolysed
          break;
          }
      head->hydrolysis_count++; //count up amount of time in KT state
      break;
    default:
```

```
      return false;
    }
  return true;
  }

bool update_head_states_nuc(struct head_struct *head1,struct head_struct *head2)
  //sets next head nucleotide binding state - both heads treated the same way
  {
  int head1_nuc_binding=head1->nuc_binding; //get current head nucleotide binding state
  int head2_nuc_binding=head2->nuc_binding; //get current head nucleotide binding state

  if (update_head_nuc(head1_nuc_binding,head1,head2)==false)
    {
    UpdateStatusBar("ERROR head1 nuc_binding value out of range", 0, 0);
    return false;
    }

  if (AMP_PNP && head2_nuc_binding == ATP) return true;
  //if non-hydrolysable analogue bound to head2 then prevent hydrolysis
  //ref Guydosh and Block experiments

  if (update_head_nuc(head2_nuc_binding,head2,head1)==false)
    {
    UpdateStatusBar("ERROR head2 nuc_binding value out of range", 0, 0);
    return false;
    }
  return true;
  }

bool head_near_MT(struct head_struct *head)
  {//motor has to be above MT before binding and binds at beta-tubulin
  return (cytosol[head->posy+1][head->posx] == MT_beta);
  //true iff head is above beta-tubulin (rather than alpha-tubulin)
  }

bool next_head_MT(struct head_struct *head,int prev_nuc_state,struct head_struct *other_head)
  {//returns MT binding state of this head given the nucleotide binding
  switch (head->nuc_binding) //look at which nucleotide is bound to this head
      {
    case ATP:
    case null:
    case ADPP:
      head->MTbinding=k_bound; //all these states have strong affinity for MT
      break;
    case ADP:
      if (head->MTbinding==k_free && head_near_MT(head)) //free but close to binding site
        {
        head->MTbinding=k_bound;//so dock head to MT
        break;
        }
      if (head->MTbinding==k_bound && prev_nuc_state==ADPP)
        //MT bound but previous state was KDP
        {
        head->MTbinding=k_free;
        //move free head towards other if linker tension
        if (linker_tension>0.)
          {
          if (other_head->posx>head->posx+1)
            head->posx=head->posx+1; //other head is in front
          else if(other_head->posx<head->posx-1)
            head->posx=head->posx-1; //other head is trailing
          //if neither then assume no tension
          }
        }
      break; //no change
    default: return false; //error - unknown nucleotide binding state
    }
  return true;
  }
```

```
void forward_step(struct head_struct *free_head,struct head_struct *MT_head)
  {
  if (free_head->posx != MT_head->posx+2)//if not already in position then
    if (Clash(MT_head->posx+2,MT_head->posy))
      //if something in the way of next MT binding site
        {//then don't step
        UpdateStatusBar("Blockage", 0, 0);
        blockage_count++; //count time blocked
        return;
        }
    else
        {//bring free head to next binding site
        free_head->posx=MT_head->posx+2;
        free_head->posy=MT_head->posy;
        }
  step=true; //flag that step has been taken
  }

void back_step(struct head_struct *free_head,struct head_struct *MT_head)
  {
  if (free_head->posx != MT_head->posx-2) //if not already in position then
    {//bring free head to previous binding site
    free_head->posx=MT_head->posx-2;
    free_head->posy=MT_head->posy;
    }
  backstep=true; //flag that backward step has been taken
  }

void process_wait_state(struct head_struct *free_head,struct head_struct *MT_head)
  {
  //one head free, other bound to MT
  //whether step is taken depends on linker strain, load, and zippering status
  if (linker_tension<linker_tension_max)
  //stepping can occur without zippering if linker tension is less than max
      if (get_rand() >= (linker_tension/linker_tension_max))
       //probability of stepping without zippering depends on linker strain
          {
          if (get_rand()>(0.5*(1.0+back_load/stall_load))) forward_step(free_head,MT_head);
        //either direction of step is equally probable at low load
        //but as load increases so does probability of backstep
          else back_step(free_head,MT_head);
        return;
        }
  //else stepping depends on zippering status
  if (zipper_active(MT_head)) forward_step(free_head,MT_head);
  }

bool update(HDC hdc,struct head_struct *head1,struct head_struct *head2)
  {//main head update routine
  struct head_struct current_head1,current_head2;

  //temporarily store part of current states
  current_head1.nuc_binding=head1->nuc_binding;
  current_head1.MTbinding=head1->MTbinding;
  current_head1.posx=head1->posx;
  current_head2.nuc_binding=head2->nuc_binding;
  current_head2.MTbinding=head2->MTbinding;
  current_head2.posx=head2->posx;

  both_heads_bound = false;//reset flag

  //if kinesin free of MT then give heads some Brownian motion
  if (head1->MTbinding==k_free && head2->MTbinding==k_free) //both heads free
    {
    motor_detached = true;//set flag to indicate that motor is diffusing
    if (Brownian(head1,head2) == false) return false; //if false then motor stuck
    }
  else//at least one head bound
```

```
    {
    motor_detached = false;
    if (head1->MTbinding == k_free) //head1 free so
         process_wait_state(head1,head2); //make step depending on conditions
    else
      if (head2->MTbinding == k_free) //head2 free so
         process_wait_state(head2,head1); //make step depending on conditions
      else both_heads_bound = true;//both heads bound to MT so step impossible
    }

  //update nucleotide binding
  if (update_head_states_nuc(head1,head2)==false) return false; //if error return false

  //update MT binding given current and previous nucleotide binding
  if (next_head_MT(head1,current_head1.nuc_binding,&current_head2)==false) return false;
  if (next_head_MT(head2,current_head2.nuc_binding,&current_head1)==false) return false;

  return true;
  }
```

B.5 Brownian motion routines

```
//Brownian.h
//provides rough diffusive motion when motor free of MT
//entry function is Brownian()

float get_rand(void)//return random number between 0.0 and 1.0
  {
  return ((float)rand()/RAND_MAX);
  }

bool Clash(int posx, int posy)
  {//returns true if position blocked
  if (cytosol[posy][posx] < 0) return TRUE; //all contents of cytosol represented by negative
numbers
  return FALSE;
  }

void Diffuse(int *posx1, int *posx2, int *posy1, int *posy2)
  {//move heads together one position at random
  float randno=get_rand();
  bool move_forward = get_rand() > 0.5

  //there are 8 possibilities of moving together...
  if (randno > 0.5)
    {
    if (get_rand() > 0.5)
      {
      if (move_forward)
        { (*posx1)++; (*posx2)++;          //move both forward
        }
      else
        { (*posx1)--; (*posx2)--;   //move both back
        }
      }
    else
      {
      if (get_rand() > 0.5)
        { (*posy1)++; (*posy2)++;          //move both down
        }
      else
        { (*posy1)--; (*posy2)--;   //move both up
        }
      }
    }
  else
    {
```

```
    if (get_rand() > 0.5)//diagonal: up, forward or back
      {
      (*posy1)++; (*posy2)++;
      if (move_forward)
        { (*posx1)++;(*posx2)++;
        }
      else
        { (*posx1)--;(*posx2)--;
        }
      }
    else  //diagonal: down, forward or back
      {
      (*posy1)--; (*posy2)--;
      if (move_forward)
        { (*posx1)++;(*posx2)++;
        }
      else
        { (*posx1)--;(*posx2)--;
        }
      }
    }
  }

void Rotate(int *posx1,int *posx2,int *posy1,int *posy2)
  {//limited rotate of molecule at random
  float randno=get_rand();

  if (*posy1 == *posy2)//heads abreast
    {
    if (randno < 0.2) (*posy1)++; //move head1 down
    else if (0.2 <= randno < 0.4) (*posy1)--; //move head1 up
    else if (0.4 <= randno < 0.6) (*posy2)--; //move head2 up
    else if (0.6 <= randno < 0.8) (*posy2)++; //move head2 down
      //else no change
    }
  else if (*posy1 > *posy2) //1 down, 2 up - assume either stays or rotates clockwise
    {
    if (randno>0.5) (*posy1)--;
    else (*posy2)++;
    }
  else //1 up, 2 down - assume either stays or rotates anticlockwise
    {
    if (randno>0.5) (*posy1)++;
    else (*posy2)--;
    }
  }

bool Brownian(struct head_struct *head1_ptr,struct head_struct *head2_ptr)
  {// Brownian moves heads at random
  int posx1,posx2,posy1,posy2;

  //assumes both heads free so move together at random
  int i=0;
    do {
    if (i++ >= 50) return false;//molecule is stuck somehow
    posx1=head1_ptr->posx; posx2=head2_ptr->posx; //get x positions of heads
    posy1=head1_ptr->posy; posy2=head2_ptr->posy; //get y positions of heads
    //move heads small distance at random
    Diffuse(&posx1,&posx2,&posy1,&posy2); //move heads together
    Rotate(&posx1,&posx2,&posy1,&posy2); //rotate molecule
    //but not outside cytosol box or into MT (or other motor)
    } while (posx1 < 0 || posx2 < 0 || posx1 >= col_max || posx2 >= col_max ||
        posy1 < 0 || posy2 < 0 || posy1 >= row_max || posy2 >= row_max ||
        Clash(posx1,posy1) || Clash(posx2,posy2));

  //remove heads from current position in cytosol
  cytosol[head1_ptr->posy][head1_ptr->posx]=0;
  cytosol[head2_ptr->posy][head2_ptr->posx]=0;
  //update store of head positions
```

```
  head1_ptr->posx=posx1; head1_ptr->posy=posy1;
  head2_ptr->posx=posx2; head2_ptr->posy=posy2;
  //place heads in new positions
  cytosol[posy1][posx1]=head_display;
  cytosol[posy2][posx2]=head_display;
  return true;
  }//end of Brownian function
```

B.6 Analysis routines

```
//analysis.h includes data analysis, results display and trace routines

void display_result(HDC hdc,int i,int j,int k,int l,int m, int r,int g,int b)
  //plots rough graphs of results under cytosol box display for indicative purposes during run
  {//displays i as x coordinate, j for y coordinate of top left and k+2 for size of colour square
    //l adjusts y to give separate plot areas, one below the other
  int leftx,topy,extent;

  j=j+(result_max+1)*k*4;//combine j with k to move graphs vertically
  leftx=((result_max+1)*i*4+4)+//i horizontally displaces top left corner of each graph
      (result_max+1)*4*(run-1);    //also displace to right for each run
  topy=12*row_max+//display results below active transport diagram
      j*4*(result_max+1)+//displaced downwards by j
      4*(result_max+1)*(int)(back_load*2); //and LOAD
//       (result_max+1)*(int)(linker_tension); //and LINKER tension
  extent=4*l*(ATP_result_max+1);
  //k displaces series of graphs horizontally

  for (int x=leftx+extent;x<=leftx+extent+3;x++)
    for (int y=topy+3;y>=topy;y--)
      SetPixel(hdc,x,y,RGB(r,g,b));
  }

void analyse_results(FILE *f, int time, int steps, int backsteps, int detachment_count)
  //sorts timing combination result array data into the 3 phases
  // and outputs result to file
  {
  int count=0;

  //write intro
  //fprintf(f,"===Run %d. ADP rel, ATP bind, ATP hydrolysis, P release\nLinker tension
%d\nsteps %d backstep %d detachments %d\nDiffusion phase: ",
  fprintf(f,"===Run %d Load -%.1f linker %.2f av time %d av steps %d av backsteps %d av
detachments %d\nDiffusion phase: ",
    run,back_load,linker_tension,time,steps,backsteps,detachment_count);
  //write data
  for (int i=0; i<result_max; i++)
    for (int j=0; j<result_max; j++)
      for (int k=0; k<result_max; k++)
        for (int l=0; l < ATP_result_max; l++)
          if (results[i][j][k][l] == P_diffusion)
            {
            //fprintf(f,"%d%d%d%d ",k,l,i,j);
            count++;
            }
//   fprintf(f,"\n***total %d\n",count);
  fprintf(f,"%d  ",count);
  count=0;
  //write intro
  fprintf(f,"Stuck phase: ");
  //write data
  for (int i=0;i<result_max;i++)
    for (int j=0;j<result_max;j++)
      for (int k=0;k<result_max;k++)
        for (int l=0;l < ATP_result_max;l++)
          if (results[i][j][k][l] == P_stuck)
            {
```

```
            //fprintf(f,"%d%d%d%d ",k,l,i,j);
            count++;
          }
//   fprintf(f,"\n***total %d\n",count);
  fprintf(f,"%d  ",count);
  count=0;
  //write intro
  fprintf(f,"Processive phase: ");
  //write data
  for (int i=0;i<result_max;i++)
    for (int j=0;j<result_max;j++)
      for (int k=0;k<result_max;k++)
        for (int l=0;l < ATP_result_max;l++)
          if (results[i][j][k][l] == P_processive)
            {
            //fprintf(f,"%d%d%d%d ",k,l,i,j);
            count++;
          }
//   fprintf(f,"\n***total %d\n",count);
  fprintf(f,"%d  ",count);
  count=0;
  //write intro
//   fprintf(f,"Missing data :");
  //write data
  for (int i=0;i<result_max;i++)
    for (int j=0;j<result_max;j++)
      for (int k=0;k<result_max;k++)
        for (int l=0;l < ATP_result_max;l++)
          if (results[i][j][k][l] == -1)
            {
            //fprintf(f,"%d%d%d%d ",k,l,i,j);
            count++;
          }
//   fprintf(f,"\ntotal %d\n",count);
  if (count>0) fprintf(f,"Missing data total %d\n",count);
  fprintf(f,"\n");
  }

void trace_heads(FILE *f,int time,struct head_struct *head1, struct head_struct *head2)
  {//outputs trace of head states to file
  bool selector=(head1->posx<=head2->posx);//position trace of heads reflecting actual position

  if (head1->MTbinding == k_free && head2->MTbinding == k_free)
    {
    if (head1->nuc_binding != ADP || head2->nuc_binding != ADP)
      fprintf(f,"\nBoth heads should be KDu?!\n");
    uucount++; //tally detached states
    if (uucount > 1) return; //since no point in outputting successive u.u
    }
  else
    {
    if (uucount > 1) fprintf(f,"(%d) ",uucount+1); //total contiguous u.u states
    uucount=0;
    }
  switch (selector?head1->nuc_binding:head2->nuc_binding) //trailing head
    {
    case null:fprintf(f,"0");break;
    case ATP:fprintf(f,"T");break;
    case ADPP:fprintf(f,"P");break;
    case ADP:if (selector) fprintf(f,"%c",(head1->MTbinding == k_bound)?'D':'u');
        else fprintf(f,"%c",(head2->MTbinding == k_bound)?'D':'u');
        break;
    }
  if (step) fprintf(f,">>");
  else if (backstep) fprintf(f,"<<");
  else fprintf(f,".");
  switch (selector?head2->nuc_binding:head1->nuc_binding)//leading head
    {
    case null:fprintf(f,"0");break;
```

```
    case ATP:fprintf(f,"T");break;
    case ADPP:fprintf(f,"P");break;
    case ADP:if (selector) fprintf(f,"%c",(head2->MTbinding == k_bound)?'D':'u');
        else fprintf(f,"%c",(head1->MTbinding == k_bound)?'D':'u');
        break;

    }
  fprintf(f," ");
  }
```

B.7 Display routines

```
//display.h
//display routines

void UpdateStatusBar(LPSTR lpszStatusString, WORD partNumber, WORD Flags);
  //displays text in status bar at bottom of window

void PlotBlob(HDC hdc, int leftx, int topy, int xlength, int ylength,COLORREF rgb)
  {//plots a rectangle xlength by ylength pixels with top left coordinates leftx, topy
    // in rgb colour with 1 pixel grey border
  xlength--; ylength--;
  for (int x=leftx; x<=leftx+xlength; x++)
    for (int y=topy+ylength; y>=topy; y--)
      {
      if (x==leftx && (y==topy+ylength || y==topy) || x==leftx+xlength &&
        (y==topy+ylength || y==topy)) //cytosol grey border
        SetPixel(hdc,x,y,RGB(cytosol_rgb,cytosol_rgb,cytosol_rgb));
      else SetPixel(hdc,x,y,rgb);
    }
  }

char DigitToChar(int digit)
  {
  switch (digit)
    {
    case 0: return '0';
    case 1: return '1';
    case 2: return '2';
    case 3: return '3';
    case 4: return '4';
    case 5: return '5';
    case 6: return '6';
    case 7: return '7';
    case 8: return '8';
    case 9: return '9';
    }
  return('*');
  }

void display_paras(void)
  {//displays head event timings in status bar
  static char Buffer[20];

  for (int i=4; i<15; i++) Buffer[i]=' ';

  Buffer[19]=0;
  Buffer[0]='T';
  Buffer[1]='h';
  Buffer[2]='=';
  Buffer[3]=DigitToChar(t_ATPhyd);

  Buffer[5]='P';
  Buffer[6]='r';
  Buffer[7]='=';
  Buffer[8]=DigitToChar(t_Prel);
```

```
  Buffer[10]='D';
  Buffer[11]='r';
  Buffer[12]='=';
  Buffer[13]=DigitToChar(t_ADPrel);

  Buffer[15]='T';
  Buffer[16]='b';
  Buffer[17]='=';
  Buffer[18]=DigitToChar(t_ATPbind);

  UpdateStatusBar(Buffer, 0, 0);
  }

void InitDisplay(HDC hdc)
  {//Initialise display of cytosol - grey box with MT filament at base
  int r,g,b;

  //pale grey background
  for (int y=0;y<=row_max;y++)
    {
    int teny=y*10;
    for (int x=0;x<=col_max;x++)
      {
      PlotBlob(hdc,x*10,teny,10,10,RGB(cytosol_rgb,cytosol_rgb,cytosol_rgb));
      }
    }

  //plot MT filament
  for (int y=0; y<=row_max; y++)
    {
    int teny=y*10;
    for (int x=0; x<=col_max; x++)
      {
      int tenx=x*10;
      r=0;g=0;b=0;
      switch (cytosol[y][x])
        {
        case 0:     continue;//already plotted the background
        case MT_beta: r=100; tenx=tenx+5; break;//beta-tubulin brown
        case MT_alpha: r=70; tenx=tenx+5; break;//alpha-tubulin deep brown
        case MT_null: r=10;g=10;b=10; tenx=tenx+5; break;//interior of MT almost black
        //bound kinesin head over beta-tubulin but also overhanging
        //alpha-tubulin (as per EM studies)
        default: b=255; //blue for error
        }
      PlotBlob(hdc,tenx,teny,10,10,RGB(r,g,b));
      }
    }
  }

void PlotHead(HDC hdc,int x,int y,struct head_struct *head)
  {//plots a blob to represent a head, colour-coded for nucleotide binding state
  int r=0,g=0,b=0;

  //traffic light colours for nucleotide, blue for no nucleotide bound
  switch (head->nuc_binding)
    {
    case ATP: r=255; break; //red
    case ADPP: g=255; r=255; break; //yellow
    case ADP: g=255; break; //green
    case null: b=255; break; //blue
    default: b=255; r=255; //purple for unknown state
    }

  PlotBlob(hdc,x*10,y*10,13,10,RGB(cytosol_rgb,cytosol_rgb,cytosol_rgb)); //clear head space
  if (head->MTbinding == k_bound) PlotBlob(hdc,x*10,y*10+2,13,8,RGB(r,g,b)); //display head
abutting MT
    //to indicate bound to MT
  else PlotBlob(hdc,x*10,y*10,13,8,RGB(r,g,b)); //not bound so display head close to MT
```

```
  }

void update_cytosol(HDC hdc, struct head_struct *head1,struct head_struct *head2)
  {//puts markers for heads into cytosol array
  //and updates previous positions in head structures

  //put previous head positions in display
  cytosol[head1->prev_posy][head1->prev_posx]=head_prev;
  cytosol[head2->prev_posy][head2->prev_posx]=head_prev;

  //put each head in current position
  cytosol[head1->posy][head1->posx]=head_display;
  cytosol[head2->posy][head2->posx]=head_display;

  //store new/current head positions
  head1->prev_posx=head1->posx;
  head1->prev_posy=head1->posy;
  head2->prev_posx=head2->posx;
  head2->prev_posy=head2->posy;
  }

void display(HDC hdc,struct head_struct *head1,struct head_struct *head2)
  //displays current state of model
  //only changes display where head activity
  //past position of head is grey, present position colour-coded by PlotHead()
  {
  int r,g,b;

  for (int y=0;y<=row_max;y++)
    {
    for (int x=0;x<=col_max;x++)
      {
      switch (cytosol[y][x])
        {
        case 0:
        case MT_beta:
        case MT_alpha:
        case MT_null:
          break;//ignore since don't move
        case head_prev:
          r=cytosol_rgb-20;g=cytosol_rgb-20;b=cytosol_rgb-20;//pale grey
          PlotBlob(hdc,x*10,y*10,13,10,RGB(r,g,b));
          //overwrite previous head position with ghost
          //showing where it was
          break;
        case head_display:
          break;
        default: b=255;r=255;g=0; //error: unknown item in cytosol
          PlotBlob(hdc,x*10,y*10,13,10,RGB(r,g,b));
            //plot a purple blob
          break;
        }
      }
    }
  //display current position of heads
  PlotHead(hdc,head1->posx,head1->posy,head1);
  PlotHead(hdc,head2->posx,head2->posy,head2);
  }

void generate_MT(int MT_top,int left,int right)
  {//Generates MT filament as line of dimers composed of alternate a-tubulin and b-tubulin
  for (int j=left; j<right-1; j=j+2)
    {
    cytosol[MT_top][j]=MT_alpha;
    cytosol[MT_top][j+1]=MT_beta;
    }
  }

void generate_block(int MT_top, int block_pos)
```

```
  {//cover MT dimer at block_pos in horizontal direction
  cytosol[MT_top-1][block_pos-1]=MT_null;
  cytosol[MT_top-1][block_pos]=MT_null;
  }

void remove_block(int MT_top, int block_pos, HDC hdc)
  {//remove block from MT in cytosol array and update display
  int x=block_pos-1, y=MT_top-6;

  cytosol[y][x]=0;
  cytosol[y][block_pos]=0;

  //remove from display by overwriting with background grey
  PlotBlob(hdc,x*10,y*10,10,10,RGB(cytosol_rgb,cytosol_rgb,cytosol_rgb));
  PlotBlob(hdc,(x+1)*10,y*10,10,10,RGB(cytosol_rgb,cytosol_rgb,cytosol_rgb));
  }
```

Appendix C Published papers

Wilson, R. J. (2009). Kinesin's walk: springy or gated head coordination? *BioSystems* **96**, 121-6.

Wilson, R. J. (2008). Simulating the Kinesin Walk: a Small Step towards Understanding Dementia. In *UKSim European Symposium on Computer Modelling and Simulation*, pp. 226-31. IEEE.

Wilson, R. J. (2008). Towards a cure for dementia: the role of axonal transport in Alzheimer's disease. *Sci Prog* **91**, 65-80.

```
                //repeat until motor engaged or hit RHS of cytosol box

          motor_detached = false; //reset flag which may be set in update_heads()

          if (head1_posx == head1->posx && head2_posx == head2->posx)
          //heads haven't moved along MT since last hydrolysis
            {
            if (step || backstep)
                {
                step=false;backstep=false;futile=true;
                //not a real step: free head has re-bound to same binding site on MT
                }
          }

          if (trace) trace_heads(f,time,head1,head2);

          if (step || backstep || futile)
            {
            //calculate next ATP molecule arrival delay modelled as Poisson process
            calc_ATP_delay(lambda);
            totATPd += (float)t_ATP_binding; //accumulate for calculation of average
            head1_posx = head1->posx;
            head2_posx = head2->posx;
                //save current head positions
            }

          if (step) //forward step has been taken
            {
            steps++; //total forward steps
            step=false; //reset flag
            display_paras('>',0); //forward step indicated in status bar
            }
          else if (backstep) //backward step has been taken
            {
            backsteps++; //total backward steps
            backstep=false; //reset flag
            display_paras('<',0); //backstep indicated in status bar
            }
          else if (futile)//no step but hydrolysis
            {
            futile=false;//reset flag
            futiles++; //total futile hydrolyses
            display_paras('F',0); //futile hydrolysis indicated in status bar
            }
          }//end motor loop

          /*//update display of system to show blow-by-blow motor progress
          //NB significantly decreases simulation speed
          for (int motor=0; motor<last_motor; motor++)
            //display each motor in turn
            display(hdc,&motor_array[motor].heada,&motor_array[motor].headb);
          //*/

        time++;//increment system time

///*
        //if blockage set and timed out then remove
        if (blockage_count > blockage_limit)
          {//blockage_count incremented by forward_step() when clash()
          //see update_heads.h
          remove_block(MT_top,col_max-13, hdc);
          blockage_count=0;
          }
//*/
        } while (any_motors_in_play() && time > 0 && time <= time_max && motor_not_at_RHS
&& no_error);
        //continue while motor processing, not at RHS and no error
        //end of INNER LOOP
```

```
      //display end state of run and find finish position
      for (int motor=0; motor<last_motor; motor++)
        {
        display(hdc, &motor_array[motor].heada, &motor_array[motor].headb);
        mfinish[motor] = (motor_array[motor].heada.posx + motor_array[motor].headb.posx)/2;
          //approx motor finish position/s
        }
      //calculate approx cargo finish position
      cfinish = mfinish[0];//finish position for 1 motor
      for (int motor = 1;motor < last_motor; motor++) cfinish = (mfinish[motor] + cfinish)/2;
        //mid finish position for 2 motors
      if (cfinish < 0) cfinish = -cfinish;

      //analyse what motor has done
      if (time >= time_max)
        {//motor has timed out - assume stuck
        UpdateStatusBar("Motor stuck", 0, 0);
        display_result(hdc,t_ATPhydrolysis,t_P_release,t_ADP_release,ATP_base,run,240,0,0);
        stuck++;
        }
      else
      if (steps + backsteps < 2)
        {//motor hasn't processed (needs at least 2 steps)
        UpdateStatusBar("Motor detached", 0, 0);
        display_result(hdc,t_ATPhydrolysis,t_P_release,t_ADP_release,ATP_base,run,0,0,240);
        diffusion++;
        }
      else
        {//motor has processed
        UpdateStatusBar("Procession", 0, 0);
        display_result(hdc,t_ATPhydrolysis,t_P_release,t_ADP_release,ATP_base,run,0,240,0);
        procession++;
        }
      //calculate results for run
      avATPd = totATPd/(float)ATP_count; //average ATP delay over total number of hydrolyses
completed in run
      runlength = ((cfinish - cstart) <= 0)?0:cfinish - cstart;
///*
      //output run results to file
      fprintf(f,"Run %2d L %d (%d to %d) Steps %4d back %2d ATP delay av %.1f T %5d V
%.1f\n",
          run,runlength,cstart,cfinish,steps,backsteps,avATPd,time,
          v_m*(float)runlength/(float)time);
//*/
      if (min_runlength == -1 || runlength < min_runlength) min_runlength = runlength;
      if (runlength > max_runlength) max_runlength = runlength;
      //accumulate totals over all runs
      av_time += (float)time; //run times
      av_steps += (float)steps; //total steps
      av_backsteps += (float)backsteps; //total backsteps
      av_futiles += (float)futiles;
      av_ATP_delay += avATPd;//average ATP arrival delay
      av_ATPs += (float)ATP_count;
      av_runlength += (float)runlength;
      }//end RUN loop

    //calculate averages over runs and output to file
    av_time /= (float)runs;
    av_steps /= (float)runs;
    av_backsteps /= (float)runs;
    av_futiles /= (float)runs;
    av_ATP_delay /= (float)runs;
    av_ATPs /= (float)runs;
    av_runlength /= (float)runs;
    fprintf(f,"Poisson i/p %d Av ATP delay %.1f Procession %d stuck %d diffusion %d\n",
      nominal_ATP_delay,av_ATP_delay,procession,stuck,diffusion);
    fprintf(f,"%d run av: L %.1f (%d to %d) Steps %.1f Backsteps %.1f ATPs % .1f Futiles %.1f
T %.0f V %.1f Dwell %.1f\n",
```

```
  runs,av_runlength,min_runlength,max_runlength,av_steps,av_backsteps,av_ATPs,av_futiles,av
_time,
       v_m*av_runlength/av_time,av_time/(av_steps+av_backsteps));
     }//end concentration variation loop

  UpdateStatusBar("Finished", 0, 0);
  fprintf(f,"===mp1 finished===\n\n");
  fclose(f); //close data output file
  EndPaint(hwnd,&ps); //finish with MS Windows
  return 0;
  }//mp1

//END of program
```

B.4 Update routines

```
//update_heads.h contains head state update routines
//entry routine is update_heads()

//The stepping conditions depend on the setting of the RBM flag:

//RBM
//===
//If RBM flag is true then following rectified Brownian mechanism (Fox and Choi 2001)
//stepping is a diffusive process with bias provided by zippering.
//Binding of ATP sets up zippering (through conformational change in head) which remains until P
released;
//(Asenjo, Weinberg & Sosa 2006)
//so, as soon as free head diffuses forward, the bound head's neck linker zippers
//and free head restrained local to forward binding site so binds i.e. a step is taken.

//If ATP_gate flag set then ATP hydrolysis gate applied
//This is hypothesised to make hydrolysis v slow unless/until free head binds to next site
//i.e. forward neck linker strain required to make hydrolysis efficient
//(Hancock and Howard, 1999 found that about 10 fold diff between
//hydrolysis time for single and twin head kinesin)

//PS
//==
//If RBM flag is false then
//following power stroke mechanism whereby ATP binding causes linker zippering
// which pulls free head forward to near next binding site so it binds and a step is taken
// (Vale and Milligan 2000)

//If ATP_gate flag also set then ATP binding gate applied
//if both heads are bound when the leading head is in K0 state, then the PS gate operates to stop
ATP binding
//until leading head hydrolyses its ATP and detaches
//(proposed as a head coordination mechanism by Rosenberg et al 2002)

bool zipper_active(struct head_struct *head)
  {//determine whether zipper active
  if ((back_load + load_variation*(get_rand()-0.5))>stall_load) return false;
    //if load sufficient to defeat zippering then return false
  if (RBM)
    //RBM model so if ATP or ADPP bound then head set up for linker zippering
    switch (head->nuc_binding)
        {
      case ATP:
      case ADPP: return true;
      default: return false;
      }
  else //power stroke model where ATP binding is the event causing zippering
    switch (head->nuc_binding)
        {
      case ATP: if (head->hydrolysis_count == 0) return true;
      default: return false;
      }
  }

bool update_head_nuc(int head_nuc_binding, struct head_struct *head, struct head_struct
*other_head)
  {//update nucleotide binding to this head
    //procession cycle is KD->K0->KT->KDP->KDu->KD...
    //Note KDP->KDu requires change to MT binding
  switch (head_nuc_binding)         //current nucleotide binding to this head
      {
    case null:
      if (head->MTbinding == k_bound)//always true in present model
```

```
        {
        if (ATP_gate && RBM == false && other_head->MTbinding == k_bound && other_head-
>posx < head->posx)
        //if PS with gating and both heads bound and this is the lead head then ATP gate operates
          head->ATP_binding_count++;
        //so count up amount of time in K0 state i.e. waiting for ATP
        else
          {
          if (head->ATP_binding_count >= t_ATP_binding)
            {
            head->ATP_binding_count=0;
            head->nuc_binding=ATP; //ATP binds because binding delay ended
            }
          else head->ATP_binding_count++;
          //count up amount of time in K0 state i.e. waiting for ATP
          }
        }
      else head->nuc_binding=ADP;
      //else ADP binds (since head is free)
      //this would only happen if head pulled off MT
      break;
    case ADP:
      if (head->MTbinding == k_bound)
        //binding to MT liberates ADP when ADP_release_count reached
        {
        if (head->ADP_release_count >= t_ADP_release)
          {
          head->ADP_release_count=0;
          head->nuc_binding=null;  //ADP released
          }
        else head->ADP_release_count++; //count up amount of time in KD state
        }
      else head->ADP_release_count=0; //count only happens when bound
        //else it's free to diffuse
      break;
    case ADPP:
      if (head->P_release_count >= t_P_release)
        {
        head->P_release_count=0;
        head->nuc_binding=ADP; //phosphate released
        }
      else head->P_release_count++; //count up amount of time in KDP state
      break;
    case ATP:
      if (RBM && ATP_gate && (other_head->MTbinding == k_free || other_head->posx !=
(head->posx+2)))
      //the RBM ATP gate is operating when switch is on and other head not bound to forward
site...
        {
        if (head->hydrolysis_count >= 10*(t_ATPhydrolysis+1))
        //increase effective delay to ATP binding by 10
          {
          head->hydrolysis_count=0;
          head->nuc_binding=ADPP;          //ATP hydrolysed
          ATP_count++; //increment count of number of ATPs hydrolysed
          break;
          }
        }
      else //either PS or RBM without ATP gate so normal hydrolysis
        if (head->hydrolysis_count >= t_ATPhydrolysis)
          {
          head->hydrolysis_count=0;
          head->nuc_binding=ADPP;          //ATP hydrolysed
          ATP_count++; //increment count of number of ATPs hydrolysed
          break;
          }
      head->hydrolysis_count++; //count up amount of time in KT state
      break;
    default:
```

```
      return false;
    }
  return true;
  }

bool update_head_states_nuc(struct head_struct *head1,struct head_struct *head2)
  //sets next head nucleotide binding state - both heads treated the same way
  {
  int head1_nuc_binding=head1->nuc_binding; //get current head nucleotide binding state
  int head2_nuc_binding=head2->nuc_binding; //get current head nucleotide binding state

  if (update_head_nuc(head1_nuc_binding,head1,head2)==false)
    {
    UpdateStatusBar("ERROR head1 nuc_binding value out of range", 0, 0);
    return false;
    }

  if (AMP_PNP && head2_nuc_binding == ATP) return true;
  //if non-hydrolysable analogue bound to head2 then prevent hydrolysis
  //ref Guydosh and Block experiments

  if (update_head_nuc(head2_nuc_binding,head2,head1)==false)
    {
    UpdateStatusBar("ERROR head2 nuc_binding value out of range", 0, 0);
    return false;
    }
  return true;
  }

bool head_near_MT(struct head_struct *head)
  {//motor has to be above MT before binding and binds at beta-tubulin
  return (cytosol[head->posy+1][head->posx] == MT_beta);
  //true iff head is above beta-tubulin (rather than alpha-tubulin)
  }

bool next_head_MT(struct head_struct *head,int prev_nuc_state,struct head_struct *other_head)
  {//returns MT binding state of this head given the nucleotide binding
  switch (head->nuc_binding) //look at which nucleotide is bound to this head
      {
    case ATP:
    case null:
    case ADPP:
      head->MTbinding=k_bound; //all these states have strong affinity for MT
      break;
    case ADP:
      if (head->MTbinding==k_free && head_near_MT(head)) //free but close to binding site
        {
        head->MTbinding=k_bound;//so dock head to MT
        break;
        }
      if (head->MTbinding==k_bound && prev_nuc_state==ADPP)
        //MT bound but previous state was KDP
        {
        head->MTbinding=k_free;
        //move free head towards other if linker tension
        if (linker_tension>0.)
          {
          if (other_head->posx>head->posx+1)
            head->posx=head->posx+1; //other head is in front
          else if(other_head->posx<head->posx-1)
            head->posx=head->posx-1; //other head is trailing
          //if neither then assume no tension
          }
        }
      break; //no change
    default: return false; //error - unknown nucleotide binding state
    }
  return true;
  }
```

```
void forward_step(struct head_struct *free_head,struct head_struct *MT_head)
  {
  if (free_head->posx != MT_head->posx+2)//if not already in position then
    if (Clash(MT_head->posx+2,MT_head->posy))
      //if something in the way of next MT binding site
        {//then don't step
        UpdateStatusBar("Blockage", 0, 0);
        blockage_count++; //count time blocked
        return;
        }
    else
        {//bring free head to next binding site
        free_head->posx=MT_head->posx+2;
        free_head->posy=MT_head->posy;
        }
  step=true; //flag that step has been taken
  }

void back_step(struct head_struct *free_head,struct head_struct *MT_head)
  {
  if (free_head->posx != MT_head->posx-2) //if not already in position then
    {//bring free head to previous binding site
    free_head->posx=MT_head->posx-2;
    free_head->posy=MT_head->posy;
    }
  backstep=true; //flag that backward step has been taken
  }

void process_wait_state(struct head_struct *free_head,struct head_struct *MT_head)
  {
  //one head free, other bound to MT
  //whether step is taken depends on linker strain, load, and zippering status
  if (linker_tension<linker_tension_max)
  //stepping can occur without zippering if linker tension is less than max
      if (get_rand() >= (linker_tension/linker_tension_max))
       //probability of stepping without zippering depends on linker strain
          {
          if (get_rand()>(0.5*(1.0+back_load/stall_load))) forward_step(free_head,MT_head);
        //either direction of step is equally probable at low load
        //but as load increases so does probability of backstep
          else back_step(free_head,MT_head);
        return;
        }
  //else stepping depends on zippering status
  if (zipper_active(MT_head)) forward_step(free_head,MT_head);
  }

bool update(HDC hdc,struct head_struct *head1,struct head_struct *head2)
  {//main head update routine
  struct head_struct current_head1,current_head2;

  //temporarily store part of current states
  current_head1.nuc_binding=head1->nuc_binding;
  current_head1.MTbinding=head1->MTbinding;
  current_head1.posx=head1->posx;
  current_head2.nuc_binding=head2->nuc_binding;
  current_head2.MTbinding=head2->MTbinding;
  current_head2.posx=head2->posx;

  both_heads_bound = false;//reset flag

  //if kinesin free of MT then give heads some Brownian motion
  if (head1->MTbinding==k_free && head2->MTbinding==k_free) //both heads free
    {
    motor_detached = true;//set flag to indicate that motor is diffusing
    if (Brownian(head1,head2) == false) return false; //if false then motor stuck
    }
  else//at least one head bound
```

```
    {
    motor_detached = false;
    if (head1->MTbinding == k_free) //head1 free so
         process_wait_state(head1,head2); //make step depending on conditions
    else
      if (head2->MTbinding == k_free) //head2 free so
         process_wait_state(head2,head1); //make step depending on conditions
      else both_heads_bound = true;//both heads bound to MT so step impossible
    }

  //update nucleotide binding
  if (update_head_states_nuc(head1,head2)==false) return false; //if error return false

  //update MT binding given current and previous nucleotide binding
  if (next_head_MT(head1,current_head1.nuc_binding,&current_head2)==false) return false;
  if (next_head_MT(head2,current_head2.nuc_binding,&current_head1)==false) return false;

  return true;
  }
```

B.5 Brownian motion routines

```
//Brownian.h
//provides rough diffusive motion when motor free of MT
//entry function is Brownian()

float get_rand(void)//return random number between 0.0 and 1.0
  {
  return ((float)rand()/RAND_MAX);
  }

bool Clash(int posx, int posy)
  {//returns true if position blocked
  if (cytosol[posy][posx] < 0) return TRUE; //all contents of cytosol represented by negative
numbers
  return FALSE;
  }

void Diffuse(int *posx1, int *posx2, int *posy1, int *posy2)
  {//move heads together one position at random
  float randno=get_rand();
  bool move_forward = get_rand() > 0.5

  //there are 8 possibilities of moving together...
  if (randno > 0.5)
    {
    if (get_rand() > 0.5)
      {
      if (move_forward)
        { (*posx1)++; (*posx2)++;          //move both forward
        }
      else
        { (*posx1)--; (*posx2)--;   //move both back
        }
      }
    else
      {
      if (get_rand() > 0.5)
        { (*posy1)++; (*posy2)++;          //move both down
        }
      else
        { (*posy1)--; (*posy2)--;   //move both up
        }
      }
    }
  else
    {
```

```
    if (get_rand() > 0.5)//diagonal: up, forward or back
      {
      (*posy1)++; (*posy2)++;
      if (move_forward)
        { (*posx1)++;(*posx2)++;
        }
      else
        { (*posx1)--;(*posx2)--;
        }
      }
    else  //diagonal: down, forward or back
      {
      (*posy1)--; (*posy2)--;
      if (move_forward)
        { (*posx1)++;(*posx2)++;
        }
      else
        { (*posx1)--;(*posx2)--;
        }
      }
    }
  }

void Rotate(int *posx1,int *posx2,int *posy1,int *posy2)
  {//limited rotate of molecule at random
  float randno=get_rand();

  if (*posy1 == *posy2)//heads abreast
    {
    if (randno < 0.2) (*posy1)++; //move head1 down
    else if (0.2 <= randno < 0.4) (*posy1)--; //move head1 up
    else if (0.4 <= randno < 0.6) (*posy2)--; //move head2 up
    else if (0.6 <= randno < 0.8) (*posy2)++; //move head2 down
      //else no change
    }
  else if (*posy1 > *posy2) //1 down, 2 up - assume either stays or rotates clockwise
    {
    if (randno>0.5) (*posy1)--;
    else (*posy2)++;
    }
  else //1 up, 2 down - assume either stays or rotates anticlockwise
    {
    if (randno>0.5) (*posy1)++;
    else (*posy2)--;
    }
  }

bool Brownian(struct head_struct *head1_ptr,struct head_struct *head2_ptr)
  {// Brownian moves heads at random
  int posx1,posx2,posy1,posy2;

  //assumes both heads free so move together at random
  int i=0;
    do {
    if (i++ >= 50) return false;//molecule is stuck somehow
    posx1=head1_ptr->posx; posx2=head2_ptr->posx; //get x positions of heads
    posy1=head1_ptr->posy; posy2=head2_ptr->posy; //get y positions of heads
    //move heads small distance at random
    Diffuse(&posx1,&posx2,&posy1,&posy2); //move heads together
    Rotate(&posx1,&posx2,&posy1,&posy2); //rotate molecule
    //but not outside cytosol box or into MT (or other motor)
    } while (posx1 < 0 || posx2 < 0 || posx1 >= col_max || posx2 >= col_max ||
        posy1 < 0 || posy2 < 0 || posy1 >= row_max || posy2 >= row_max ||
        Clash(posx1,posy1) || Clash(posx2,posy2));

  //remove heads from current position in cytosol
  cytosol[head1_ptr->posy][head1_ptr->posx]=0;
  cytosol[head2_ptr->posy][head2_ptr->posx]=0;
  //update store of head positions
```

```
  head1_ptr->posx=posx1; head1_ptr->posy=posy1;
  head2_ptr->posx=posx2; head2_ptr->posy=posy2;
  //place heads in new positions
  cytosol[posy1][posx1]=head_display;
  cytosol[posy2][posx2]=head_display;
  return true;
  }//end of Brownian function
```

B.6 Analysis routines

```
//analysis.h includes data analysis, results display and trace routines

void display_result(HDC hdc,int i,int j,int k,int l,int m, int r,int g,int b)
  //plots rough graphs of results under cytosol box display for indicative purposes during run
  {//displays i as x coordinate, j for y coordinate of top left and k+2 for size of colour square
    //l adjusts y to give separate plot areas, one below the other
  int leftx,topy,extent;

  j=j+(result_max+1)*k*4;//combine j with k to move graphs vertically
  leftx=((result_max+1)*i*4+4)+//i horizontally displaces top left corner of each graph
      (result_max+1)*4*(run-1);    //also displace to right for each run
  topy=12*row_max+//display results below active transport diagram
      j*4*(result_max+1)+//displaced downwards by j
      4*(result_max+1)*(int)(back_load*2); //and LOAD
//       (result_max+1)*(int)(linker_tension); //and LINKER tension
  extent=4*l*(ATP_result_max+1);
  //k displaces series of graphs horizontally

  for (int x=leftx+extent;x<=leftx+extent+3;x++)
    for (int y=topy+3;y>=topy;y--)
      SetPixel(hdc,x,y,RGB(r,g,b));
  }

void analyse_results(FILE *f, int time, int steps, int backsteps, int detachment_count)
  //sorts timing combination result array data into the 3 phases
  // and outputs result to file
  {
  int count=0;

  //write intro
  //fprintf(f,"===Run %d. ADP rel, ATP bind, ATP hydrolysis, P release\nLinker tension
%d\nsteps %d backstep %d detachments %d\nDiffusion phase: ",
  fprintf(f,"===Run %d Load -%.1f linker %.2f av time %d av steps %d av backsteps %d av
detachments %d\nDiffusion phase: ",
    run,back_load,linker_tension,time,steps,backsteps,detachment_count);
  //write data
  for (int i=0; i<result_max; i++)
    for (int j=0; j<result_max; j++)
      for (int k=0; k<result_max; k++)
        for (int l=0; l < ATP_result_max; l++)
          if (results[i][j][k][l] == P_diffusion)
            {
            //fprintf(f,"%d%d%d%d ",k,l,i,j);
            count++;
            }
//   fprintf(f,"\n***total %d\n",count);
  fprintf(f,"%d  ",count);
  count=0;
  //write intro
  fprintf(f,"Stuck phase: ");
  //write data
  for (int i=0;i<result_max;i++)
    for (int j=0;j<result_max;j++)
      for (int k=0;k<result_max;k++)
        for (int l=0;l < ATP_result_max;l++)
          if (results[i][j][k][l] == P_stuck)
            {
```

```
            //fprintf(f,"%d%d%d%d ",k,l,i,j);
            count++;
          }
//   fprintf(f,"\n***total %d\n",count);
  fprintf(f,"%d  ",count);
  count=0;
  //write intro
  fprintf(f,"Processive phase: ");
  //write data
  for (int i=0;i<result_max;i++)
    for (int j=0;j<result_max;j++)
      for (int k=0;k<result_max;k++)
        for (int l=0;l < ATP_result_max;l++)
          if (results[i][j][k][l] == P_processive)
            {
            //fprintf(f,"%d%d%d%d ",k,l,i,j);
            count++;
          }
//   fprintf(f,"\n***total %d\n",count);
  fprintf(f,"%d  ",count);
  count=0;
  //write intro
//   fprintf(f,"Missing data :");
  //write data
  for (int i=0;i<result_max;i++)
    for (int j=0;j<result_max;j++)
      for (int k=0;k<result_max;k++)
        for (int l=0;l < ATP_result_max;l++)
          if (results[i][j][k][l] == -1)
            {
            //fprintf(f,"%d%d%d%d ",k,l,i,j);
            count++;
          }
//   fprintf(f,"\ntotal %d\n",count);
  if (count>0) fprintf(f,"Missing data total %d\n",count);
  fprintf(f,"\n");
  }

void trace_heads(FILE *f,int time,struct head_struct *head1, struct head_struct *head2)
  {//outputs trace of head states to file
  bool selector=(head1->posx<=head2->posx);//position trace of heads reflecting actual position

  if (head1->MTbinding == k_free && head2->MTbinding == k_free)
    {
    if (head1->nuc_binding != ADP || head2->nuc_binding != ADP)
      fprintf(f,"\nBoth heads should be KDu?!\n");
    uucount++; //tally detached states
    if (uucount > 1) return; //since no point in outputting successive u.u
    }
  else
    {
    if (uucount > 1) fprintf(f,"(%d) ",uucount+1); //total contiguous u.u states
    uucount=0;
    }
  switch (selector?head1->nuc_binding:head2->nuc_binding) //trailing head
    {
    case null:fprintf(f,"0");break;
    case ATP:fprintf(f,"T");break;
    case ADPP:fprintf(f,"P");break;
    case ADP:if (selector) fprintf(f,"%c",(head1->MTbinding == k_bound)?'D':'u');
        else fprintf(f,"%c",(head2->MTbinding == k_bound)?'D':'u');
        break;
    }
  if (step) fprintf(f,">>");
  else if (backstep) fprintf(f,"<<");
  else fprintf(f,".");
  switch (selector?head2->nuc_binding:head1->nuc_binding)//leading head
    {
    case null:fprintf(f,"0");break;
```

```
    case ATP:fprintf(f,"T");break;
    case ADPP:fprintf(f,"P");break;
    case ADP:if (selector) fprintf(f,"%c",(head2->MTbinding == k_bound)?'D':'u');
        else fprintf(f,"%c",(head1->MTbinding == k_bound)?'D':'u');
        break;

    }
  fprintf(f," ");
  }
```

B.7 Display routines

```
//display.h
//display routines

void UpdateStatusBar(LPSTR lpszStatusString, WORD partNumber, WORD Flags);
  //displays text in status bar at bottom of window

void PlotBlob(HDC hdc, int leftx, int topy, int xlength, int ylength,COLORREF rgb)
  {//plots a rectangle xlength by ylength pixels with top left coordinates leftx, topy
    // in rgb colour with 1 pixel grey border
  xlength--; ylength--;
  for (int x=leftx; x<=leftx+xlength; x++)
    for (int y=topy+ylength; y>=topy; y--)
      {
      if (x==leftx && (y==topy+ylength || y==topy) || x==leftx+xlength &&
        (y==topy+ylength || y==topy)) //cytosol grey border
        SetPixel(hdc,x,y,RGB(cytosol_rgb,cytosol_rgb,cytosol_rgb));
      else SetPixel(hdc,x,y,rgb);
    }
  }

char DigitToChar(int digit)
  {
  switch (digit)
    {
    case 0: return '0';
    case 1: return '1';
    case 2: return '2';
    case 3: return '3';
    case 4: return '4';
    case 5: return '5';
    case 6: return '6';
    case 7: return '7';
    case 8: return '8';
    case 9: return '9';
    }
  return('*');
  }

void display_paras(void)
  {//displays head event timings in status bar
  static char Buffer[20];

  for (int i=4; i<15; i++) Buffer[i]=' ';

  Buffer[19]=0;
  Buffer[0]='T';
  Buffer[1]='h';
  Buffer[2]='=';
  Buffer[3]=DigitToChar(t_ATPhyd);

  Buffer[5]='P';
  Buffer[6]='r';
  Buffer[7]='=';
  Buffer[8]=DigitToChar(t_Prel);
```

```
  Buffer[10]='D';
  Buffer[11]='r';
  Buffer[12]='=';
  Buffer[13]=DigitToChar(t_ADPrel);

  Buffer[15]='T';
  Buffer[16]='b';
  Buffer[17]='=';
  Buffer[18]=DigitToChar(t_ATPbind);

  UpdateStatusBar(Buffer, 0, 0);
  }

void InitDisplay(HDC hdc)
  {//Initialise display of cytosol - grey box with MT filament at base
  int r,g,b;

  //pale grey background
  for (int y=0;y<=row_max;y++)
    {
    int teny=y*10;
    for (int x=0;x<=col_max;x++)
      {
      PlotBlob(hdc,x*10,teny,10,10,RGB(cytosol_rgb,cytosol_rgb,cytosol_rgb));
      }
    }

  //plot MT filament
  for (int y=0; y<=row_max; y++)
    {
    int teny=y*10;
    for (int x=0; x<=col_max; x++)
      {
      int tenx=x*10;
      r=0;g=0;b=0;
      switch (cytosol[y][x])
        {
        case 0:     continue;//already plotted the background
        case MT_beta: r=100; tenx=tenx+5; break;//beta-tubulin brown
        case MT_alpha: r=70; tenx=tenx+5; break;//alpha-tubulin deep brown
        case MT_null: r=10;g=10;b=10; tenx=tenx+5; break;//interior of MT almost black
        //bound kinesin head over beta-tubulin but also overhanging
        //alpha-tubulin (as per EM studies)
        default: b=255; //blue for error
        }
      PlotBlob(hdc,tenx,teny,10,10,RGB(r,g,b));
      }
    }
  }

void PlotHead(HDC hdc,int x,int y,struct head_struct *head)
  {//plots a blob to represent a head, colour-coded for nucleotide binding state
  int r=0,g=0,b=0;

  //traffic light colours for nucleotide, blue for no nucleotide bound
  switch (head->nuc_binding)
    {
    case ATP: r=255; break; //red
    case ADPP: g=255; r=255; break; //yellow
    case ADP: g=255; break; //green
    case null: b=255; break; //blue
    default: b=255; r=255; //purple for unknown state
    }

  PlotBlob(hdc,x*10,y*10,13,10,RGB(cytosol_rgb,cytosol_rgb,cytosol_rgb)); //clear head space
  if (head->MTbinding == k_bound) PlotBlob(hdc,x*10,y*10+2,13,8,RGB(r,g,b)); //display head
abutting MT
    //to indicate bound to MT
  else PlotBlob(hdc,x*10,y*10,13,8,RGB(r,g,b)); //not bound so display head close to MT
```

```
  }

void update_cytosol(HDC hdc, struct head_struct *head1,struct head_struct *head2)
  {//puts markers for heads into cytosol array
  //and updates previous positions in head structures

  //put previous head positions in display
  cytosol[head1->prev_posy][head1->prev_posx]=head_prev;
  cytosol[head2->prev_posy][head2->prev_posx]=head_prev;

  //put each head in current position
  cytosol[head1->posy][head1->posx]=head_display;
  cytosol[head2->posy][head2->posx]=head_display;

  //store new/current head positions
  head1->prev_posx=head1->posx;
  head1->prev_posy=head1->posy;
  head2->prev_posx=head2->posx;
  head2->prev_posy=head2->posy;
  }

void display(HDC hdc,struct head_struct *head1,struct head_struct *head2)
  //displays current state of model
  //only changes display where head activity
  //past position of head is grey, present position colour-coded by PlotHead()
  {
  int r,g,b;

  for (int y=0;y<=row_max;y++)
    {
    for (int x=0;x<=col_max;x++)
      {
      switch (cytosol[y][x])
        {
        case 0:
        case MT_beta:
        case MT_alpha:
        case MT_null:
          break;//ignore since don't move
        case head_prev:
          r=cytosol_rgb-20;g=cytosol_rgb-20;b=cytosol_rgb-20;//pale grey
          PlotBlob(hdc,x*10,y*10,13,10,RGB(r,g,b));
          //overwrite previous head position with ghost
          //showing where it was
          break;
        case head_display:
          break;
        default: b=255;r=255;g=0; //error: unknown item in cytosol
          PlotBlob(hdc,x*10,y*10,13,10,RGB(r,g,b));
            //plot a purple blob
          break;
        }
      }
    }
  //display current position of heads
  PlotHead(hdc,head1->posx,head1->posy,head1);
  PlotHead(hdc,head2->posx,head2->posy,head2);
  }

void generate_MT(int MT_top,int left,int right)
  {//Generates MT filament as line of dimers composed of alternate a-tubulin and b-tubulin
  for (int j=left; j<right-1; j=j+2)
    {
    cytosol[MT_top][j]=MT_alpha;
    cytosol[MT_top][j+1]=MT_beta;
    }
  }

void generate_block(int MT_top, int block_pos)
```

```
  {//cover MT dimer at block_pos in horizontal direction
  cytosol[MT_top-1][block_pos-1]=MT_null;
  cytosol[MT_top-1][block_pos]=MT_null;
  }

void remove_block(int MT_top, int block_pos, HDC hdc)
  {//remove block from MT in cytosol array and update display
  int x=block_pos-1, y=MT_top-6;

  cytosol[y][x]=0;
  cytosol[y][block_pos]=0;

  //remove from display by overwriting with background grey
  PlotBlob(hdc,x*10,y*10,10,10,RGB(cytosol_rgb,cytosol_rgb,cytosol_rgb));
  PlotBlob(hdc,(x+1)*10,y*10,10,10,RGB(cytosol_rgb,cytosol_rgb,cytosol_rgb));
  }
```

Appendix C Published papers

Wilson, R. J. (2009). Kinesin's walk: springy or gated head coordination? *BioSystems* **96**, 121-6.

Wilson, R. J. (2008). Simulating the Kinesin Walk: a Small Step towards Understanding Dementia. In *UKSim European Symposium on Computer Modelling and Simulation*, pp. 226-31. IEEE.

Wilson, R. J. (2008). Towards a cure for dementia: the role of axonal transport in Alzheimer's disease. *Sci Prog* **91**, 65-80.

BioSystems 96 (2009) 121–126

Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

Kinesin's walk: Springy or gated head coordination?

Richard J. Wilson*

MOAC Centre, University of Warwick, Coventry CV4 7AL, UK

ARTICLE INFO

Article history:
Received 11 July 2008
Received in revised form 12 December 2008
Accepted 18 December 2008

Keywords:
Rectified Brownian motion
ATP gating
Motor protein procession
Entropic strain
Agent-based modeling

ABSTRACT

Conventional kinesin (kinesin-1) is a motor protein that performs a vital function in the eukaryotic cell: it actively transports cargo to required destinations. Kinesin pulls cargo along microtubule tracks using twin linked motor domains (heads) that bind the microtubule, hydrolyse ATP, and alternately step forward. The detail of the kinesin walk has yet to be discovered but a prominent theory is that the mechanism is rectified Brownian motion (RBM) biased by linker zippering. There is evidence that an ATP binding gate coordinates the heads. The hypothesis proposed here is that the gate is unnecessary, that entropic linker strain is sufficient to enable procession. An agent-based computer simulation has been devised to explore head coordination in the RBM model. Walking was found to emerge *in silico* without a gate to synchronise the heads. Further investigation of the model by applying a range of hindering loads resulted in backstepping or detachment with similar characteristics to behaviour observed *in vitro*. It is unclear whether kinesin waits at an obstacle but adding an ATP hydrolysis gate to the model in order to force waiting resulted in the model behaving less realistically under load. It is argued here that an RBM model free of gating is a good candidate for explaining kinesin procession.

1. Introduction

Cell survival is dependent upon the active transport of macromolecules, vesicles and organelles to their functional destinations. Long distance active transport is particularly important to neurons which have extended projections, the longest being the axon (Hirokawa, 1998). Failure of axonal transport is implicated in neurodegenerative diseases including motor neuron disease, Huntington's disease, amyotrophic lateral sclerosis and Alzheimer's disease (Gunawardena and Goldstein, 2004; Roy et al., 2005) and is thought to be an early event in the development of Alzheimer's (Stokin et al., 2005). Improving our understanding of axonal transport is therefore an important task in the programme to conquer such disease.

Long distance axonal transport is accomplished by motor proteins that traverse microtubules. Microtubules are hollow, 25 nm diameter tubes, typically composed of 13 laterally bound filaments consisting of 8 nm long tubulin heterodimers that spontaneously bond end to end. This structure makes a microtubule a polar polymer with a ring of α-tubulins at the minus end and one of β-tubulins at the plus end (Nogales et al., 1999). The motor protein under investigation here is the most studied member of the kinesin family of motor proteins known as conventional kinesin or kinesin-1, referred to herein simply as kinesin. Kinesin is a homodimer comprising 2 heavy and 2 light chains (Vale, 2003). Each heavy chain N-terminal region forms a globular motor domain (head) connected by a short, flexible, single polypeptide neck linker to a long, coiled-coil stalk. The stalks form the dimer and, at their C-terminal region, combine with the light chains to form a fan-like tail which binds to cargo. The heads have two binding sites: one binds and hydrolyses the nucleotide ATP, the other binds a microtubule with nucleotide-dependent strength (Uemura et al., 2002).

Kinesin walks along the outside of a microtubule towards the plus end by processively stepping along a filament for some hundreds of steps (Howard et al., 1989; Ray et al., 1993). A single ATP molecule is hydrolysed at each step (Schnitzer and Block, 1997) which moves the molecule forward by the length of a tubulin dimer (Svoboda et al., 1993). The weight of experimental evidence favours an asymmetric hand-over-hand stepping over the alternative possibilities of symmetric hand-over-hand or inchworm motion: kinesin walks in a similar manner to toeing a line (Asbury, 2005).

The way kinesin utilises the free energy of ATP hydrolysis to tow cargo has provoked controversy. Rice et al. (1999) observed that ATP binding causes a conformational change in the normally flexible neck linker: it becomes fixed (zippered) to the head and aligned in the direction of motion. Vale and Milligan (2000) proposed that this change constitutes a power stroke whereby zippering pulls the free head forwards to the next binding site via its neck linker. Given a stall force of 6 pN together with a step of 8 nm, kinesin develops 48 pN nm of work per step (29 kJ/mol) but the free energy of zippering is calculated to be about 3 kJ/mol (Rice et al., 2003) which is clearly not enough to power kinesin. An alternative role for

* Tel.: +44 2476 574695; fax: +44 2476 575795.
E-mail address: richard.j.wilson@warwick.ac.uk.

0303-2647/$ – see front matter © 2008 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2008.12.002

zippering has been proposed by Fox and Choi (2001) on the basis that viscosity and thermal forces are dominant at the nanoscale. In their model, stepping is achieved by rectified Brownian motion (RBM). Rather than being a source of force, zippering provides directionality by forward biasing the otherwise random motion of the free head. The work done by kinesin in transporting its load is then powered by diffusion and binding of the free head to the next site on the microtubule.

There is controversy about kinesin's configuration when the molecule is waiting for ATP to bind during procession (the wait state). Yildiz et al. (2004) propose that both heads are bound to the microtubule whereas Mori et al. (2007) propose that one head is detached. There are four possible configurations of the ADP-bound head: bound and waiting to step forward, free to diffuse, parked, or bound after stepping, but no conclusive evidence determining which configuration is correct (Hackney, 2007). The configuration assumed in this study is that the head is free to diffuse. Given that the neck linkers are flexible, the question then is why the free head does not bind the microtubule and so take a forward or backward step before ATP arrives. The proposal made here is that the mechanism is entropic linker strain, whereby thermal forces render linkers spring-like.

Entropic linker strain has previously been proposed as a mechanism to pull the free head away from the microtubule following phosphate release and to prevent re-binding (Rice et al., 2003). Since occasional backstepping has been observed during procession (Svoboda and Block, 1994), it is proposed here that linker strain does not prevent re-binding but does make it unlikely. If the linkers were completely flexible then the free head could diffuse as far as either binding site but the spring-like nature of the linkers significantly reduces the likelihood of it reaching either site (Fig. 1a). Normal stepping occurs after ATP binds causing zippering in the forward direction (Rice et al., 1999). The free head can now no longer reach the rear MT binding site (Fig. 1b) and stepping forward is promoted by the entropic strain reduction of zippering.

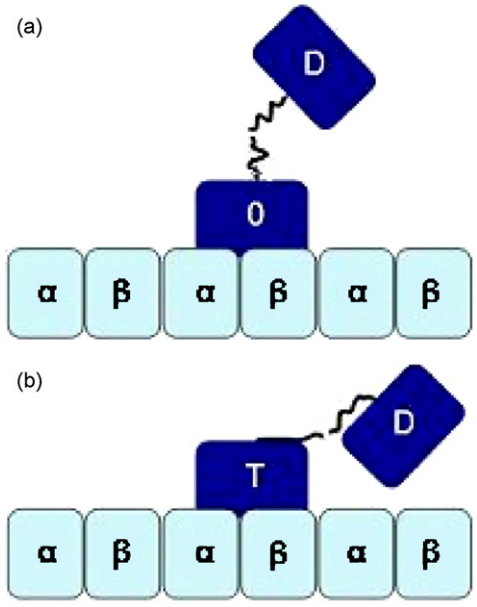

Fig. 1. Diagrammatic depiction of kinesin's motor domain in (a) wait state; (b) after zippering of linker. Head depicted as dark blue rectangle (D indicates ADP-bound; T indicates ATP-bound; 0 indicates nucleotide-free), neck linkers as irregular lines, microtubule as light blue rectangles labelled α for α-tubulin, β for β-tubulin. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

This paper focuses on the mechanism of procession with special reference to the role of head coordination. Procession requires that one head detaches and steps forward while the other stays bound to the microtubule. Detachment occurs when phosphate is released after ATP hydrolysis. If the bound head completes hydrolysis and detaches before the free head binds then kinesin would diffuse away from the microtubule. Rosenfeld et al. (2003) propose that neck linker strain resulting from both heads being bound prevents ATP from binding until the strain is released by one head detaching thus providing a coordinating point in the kinetic cycle. Mather and Fox (2006) incorporate this gating mechanism in their RBM model. The hypothesis put forward in this paper is that ATP gating is not necessary for head coordination; rather, entropic neck linker strain suffices to enable kinesin procession. An agent-based simulation has been developed to test the hypothesis and investigate the resulting model. Procession was found to emerge without an ATP binding gate *in silico* thus lending support to the hypothesis. Subjecting simulated kinesin to hindering loads resulted in backstepping or detachment, behaviour observed *in vitro* (Nishiyama et al., 2002; Carter and Cross, 2005). Placing a barrier on the track caused the simulated kinesin to diffuse away from the microtubule as observed *in vitro* by Crevel et al. (2004). Seitz and Surrey (2006), however, found that kinesin waits at an obstacle. An ATP hydrolysis gate was necessary to replicate waiting behaviour *in silico* but was found to have a negative effect on processivity and load characteristics. A non-gated RBM model is proposed as the most likely candidate to explain kinesin's procession.

2. The Simulation

Most previous modelling of kinesin has taken one of two approaches: Brownian ratchet or chemical-kinetic (Kolomeisky and Fisher, 2007). In the Brownian (or thermal) ratchet model a particle moves stochastically between potentials; in the kinetic model it moves through a series of chemical states linked by rate constants. The focus of these models is to re-create data relationships found in single-molecule laboratory experiments such as that between load and velocity.

Here, a systems approach is taken with the focus on the relationship between procession of the molecule and the motive component parts: the heads and their linkers; procession is not built into the model but emerges if the heads coordinate. The author has designed and programmed a discrete, event-driven simulation with a fixed-increment clock incorporating elements of previous approaches to produce an original method of modelling kinesin. Previous work with this model indicated that rectified Brownian motion is a better candidate than a power stroke for the stepping mechanism (Wilson, 2008b). The new work reported here builds on the earlier study by further exploring the RBM theory with the immediate intent of providing indicative results relating to its theoretical viability without ATP gating. Definitive results require experimental evidence but the work presented and discussed here in the light of laboratory findings can at least stimulate debate and further experiments which will progress our knowledge about the kinesin walk. Simple, agent-based modelling was chosen for its scalability and it is intended that a model of axonal transport will be built up from this preliminary work in order to investigate failure modes relevant to neurodegeneration.

2.1. Modelling the Motor Domain

Kinesin's motor domain is composed of twin heads connected by neck linkers to the base of the stalk. The motor is confined to a 2D box representing a small section of cytosol containing a microtubule filament. The heads are treated as identical agents, following

the same hydrolysis and binding rules. Each head is modelled as a separate finite state machine (FSM) or finite state automaton: a set of discrete states with transitions between them where the next state depends on the previous state and the current input (if any). There are five possible states of nucleotide and microtubule binding for a kinesin head. These are denoted by KD, K0, KT, KDP and KDu which represent, respectively, a kinesin head bound to ADP, to no nucleotide, to ATP, to hydrolysed ATP (all bound to the microtubule) and to the ADP-bound head free of the microtubule. The transition sequence between the states of the machine is:

$$\mathrm{KDu} \rightarrow \mathrm{KD} \rightarrow \mathrm{K0} \rightarrow \mathrm{KT} \rightarrow \mathrm{KDP} \rightarrow \mathrm{KDu} \ldots$$

Procession occurs if each head goes through the transition sequence but out of phase as can be seen in Fig. 2 which illustrates the procession cycle resulting from head coordination. Entry into the cycle is initiated when kinesin in solution encounters the microtubule. One head binds to the microtubule which causes the head to release its ADP while the other head remains free (Hackney, 1994). This configuration is assumed to be identical to the wait state that occurs during procession when the molecule is awaiting ATP. ATP binds the nucleotide-free head causing its neck linker to zipper to the head (Rice et al., 1999) which results in the ADP-bound head binding to the next microtubule binding site. This in turn causes ADP release followed by ATP binding and hydrolysis. Meanwhile the other head hydrolyses ATP, releases phosphate and detaches (Rosenfeld et al., 2003; Klumpp et al., 2004). Thus each head alternately steps forward and hydrolyses ATP: kinesin walks along the microtubule.

The following series of rules embodies the hydrolysis cycle and the interaction between individual head and microtubule:

(1) If an ADP-bound head encounters the microtubule, it binds (KDu → KD)
(2) Binding to the microtubule causes ADP release (KD → K0)
(3) ATP binds the empty head (K0 → KT)
(4) The bound head hydrolyses ATP (KT → KDP)
(5) Head detachment occurs with phosphate release (KDP → KDu)

The corresponding FSM is shown in Fig. 3; it applies to both heads. Inter-head gating is deliberately left out of the kinetic cycle described by these rules as the purpose is to see under what conditions the heads operating independently can coordinate.

2.2. Linker Strain and Stepping

In this model linker strain influences stepping during the wait state which occurs during procession when kinesin is waiting for ATP to bind, just before stepping. There are several possible configurations for the wait state but no definitive evidence selecting one option (Hackney, 2007). The controversial assumption made here is that the wait state comprises one head bound and one head free. This occurs in the model after rule 2 has been applied to one

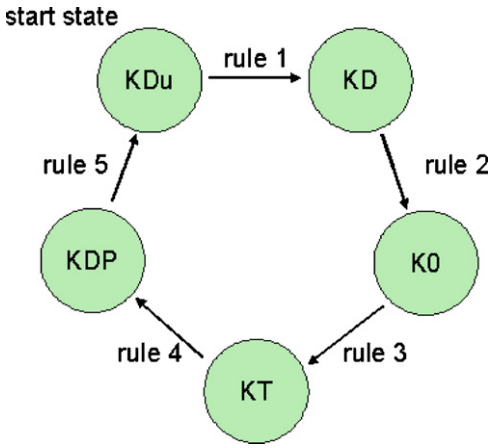

Fig. 3. FSM state transition diagram.

head and rule 5 has been applied to the other. One head is then nucleotide free and bound to the microtubule awaiting ATP and the other is ADP-bound and diffusing subject to restraint by the neck linkers as illustrated in Fig. 1a and the rightmost state depicted in Fig. 2.

The stepping of the free head in the wait state is simulated by a pseudo-random number function such that there is an equal probability of the head moving forwards or backwards. Whether or not the head binds the microtubule depends on the entropic linker strain though this does not affect the probability of forward compared to rearward binding. The strain is treated as a variable when examining its effect on kinesin's processive behaviour. At maximum strain the free head cannot reach either binding site while, for lower values of strain, the probability of binding depends inversely on the value of the strain. Note that this is not the relationship between binding probability and strain in the loading experiments where the strain value (and hence the probability) is fixed to a realistic value in that the resulting frequency of backstepping matches *in vitro* observations.

2.3. Zippering

Zippering is modelled as a switch that is activated when the wait state is exited by ATP binding (rule 3 above) and reset when phosphate is released (rule 5 above). Thus activation of the switch simulates the setting up of zippering of the neck linker to the bound head and resetting the switch simulates the linker unzippering. The probability of a forward step is made certain when the free head diffuses forwards and the zippering switch is set (see Fig. 1b).

2.4. Load

The effect of hindering load is simulated by altering the operation of the zippering switch. Loads less than 4 pN are assumed to

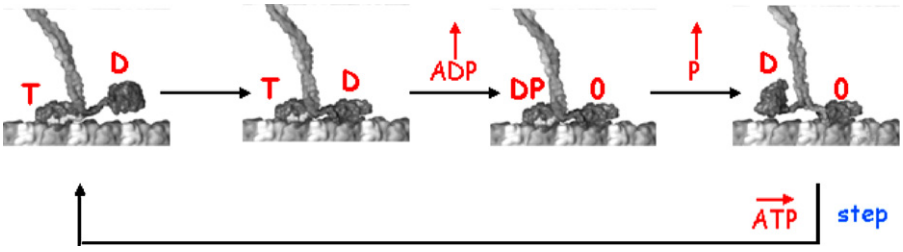

Fig. 2. Kinesin procession as a series of states or snapshots of the motor domain (adapted from Vale and Milligan, 2000). A short section of microtubule is depicted as alternating light and dark tubulins at the base of each snapshot. Kinesin is stepping along the microtubule towards the right. The capital letter above each head indicates which nucleotide is bound: D for ADP; T for ATP; DP for hydrolysed ATP; 0 for none.

have no effect on zippering, the probability of zippering is progressively reduced as the load is increased from 4 pN to 7 pN, and loads above 7 pN prevent zippering. There is no attempt to model any effect load may have on head binding.

2.5. Obstacle

An obstacle can be placed towards the plus end of the microtubule for a given time interval. Initial simulation results with a barrier on the microtubule (Section 3.4) indicated the possible need for an additional modelling constraint: an ATP hydrolysis gate. The gate was implemented by slowing ATP hydrolysis tenfold unless the partner head is bound to the forward binding site. This mirrors the experimental finding by Hancock and Howard (1999) that single-headed kinesin hydrolyses ATP ten times slower than native kinesin.

In order to show the progression of the simulation, the state of the system is displayed in a graphic window. The activity of the motor is displayed at the top of the screen while a results summary is plotted underneath. The experimenter can thus keep a visual check on the system's behaviour. As a first step towards modelling axonal transport, the heads are contained within a two-dimensional box representing an area of cytosol containing a length of microtubule filament laid out laterally as alternate α- and β-tubulins. Each simulation run starts with kinesin positioned at the same location near the minus end of the microtubule. Pseudo-random motion is applied to each head to approximate diffusion until the motor engages with the microtubule. The simulation run is terminated when the motor reaches the plus end of the microtubule or becomes stuck with both heads permanently bound. In order to minimise pseudo-random bias affecting the results, each run of the program was repeated five times and an average over the values taken as a data point. Results are output to file for analysis.

3. Results and Discussion

3.1. Head Coordination and Procession

The mechanical conditions for procession are that at least one head is bound to the microtubule at all times and that the heads take it in turns to detach and step forward. If the first condition were not met then kinesin would diffuse away from the microtubule. If the second condition were not met then kinesin would stall, remaining tightly bound to the microtubule. To achieve the required head coordination, an ATP binding gate has been proposed whereby binding of ATP to the nucleotide-free head is prevented by neck linker strain resulting from both heads being bound to the microtubule (Rosenfeld et al., 2003). Support for this view comes from *in vitro* experiments with mutant kinesins indicating that detachment of the partner head is necessary to enable ATP binding (Klumpp et al., 2004). It is possible that native kinesin does not employ this coordination mechanism; the alternative proposed here is that entropic linker strain is all that is necessary.

In order to test the hypothesis that entropic neck linker strain is sufficient to coordinate the heads, the simulation was used to assess the effect of varying the strain on processivity. Processivity was measured by determining whether procession arose under different head event timing conditions. The timing of 3 events was varied: ADP release (KD → K0), ATP hydrolysis (KT → KDP), and phosphate release with head detachment (KDP → KDu). Each parameter was given a value in the range 1–3 and all possible combinations ($3^3 = 27$) run through for each level of linker strain. These values are relative to the timing of head binding (KDu → KD) which is treated here as a constant. Neck linker strain was linearly varied from 0 (representing no strain) to 10 (representing enough strain to prevent binding without zippering). The number of timing combinations which resulted in uninterrupted procession along the microtubule was counted for each strain value. This was deemed a suitable measure of head coordination as any interruption to procession would entail both heads detaching from the microtubule which can only happen if the heads lose coordination.

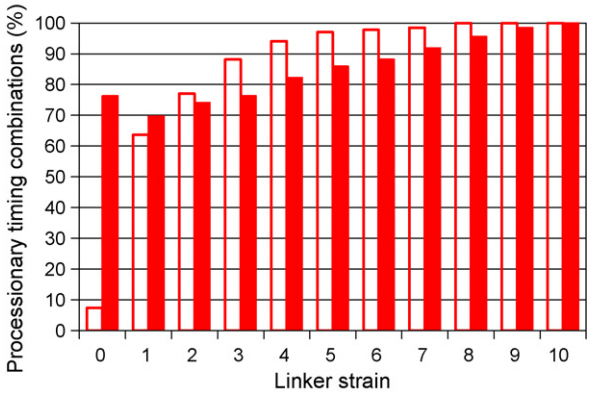

Fig. 4. Relationship between neck linker strain and timing combinations giving rise to procession. Hollow bars show values for the model without the ATP gate, filled bars show values for the model with the ATP gate.

The hypothesis is supported by the simulation results which are displayed in histogram form as hollow bars in Fig. 4. The model shows procession without an ATP binding gate under the whole range of timing conditions at high linker strain (values above 7). The percentage of timing conditions producing procession slowly reduces with strain though remaining above 60% until a dramatic drop to below 10% when the strain is reduced to zero. Thus linker strain coordinates the heads over a range of timings though it is not essential since some timing combinations gave rise to procession even without strain.

The filled bars in Fig. 4 show that the ATP hydrolysis gate decreases the incidence of procession except at maximum and very low linker strain. The effect of ATP gating is most striking with zero linker strain where over 70% of the timings resulted in procession: comparable to an ungated linker strain of 2. The simulation confirms that ATP gating is therefore a potential alternative stabilising factor for kinesin as would be expected given its direct influence in head synchronisation though linker strain is more effective than ATP gating at head coordination in this model.

3.2. Load, Stepping and Detachment

The proposed model accounts for the occasional backward movement observed under light load and the increase in backstepping with increasing load observed *in vitro* (Svoboda and Block, 1994; Nishiyama et al., 2002; Carter and Cross, 2005). Backstepping at light loads (loads that do not affect zippering), can be explained by considering the wait state. The wait state is the period after one head has released ADP and before ATP binds. The model assumes that its partner head is free to diffuse. In this configuration, if we further assume that entropic neck linker tension makes binding and ADP release improbable (rather than impossible), there is a small window of opportunity for the free head to bind the rear site and release ADP thus the model predicts that kinesin takes an occasional backstep even at low load. As the load is increased it begins to counteract the biasing effect of zippering so that the number of backsteps increases. At stall, the load is high enough to counteract zippering so that equal numbers of forward and backward steps are taken resulting in no net movement.

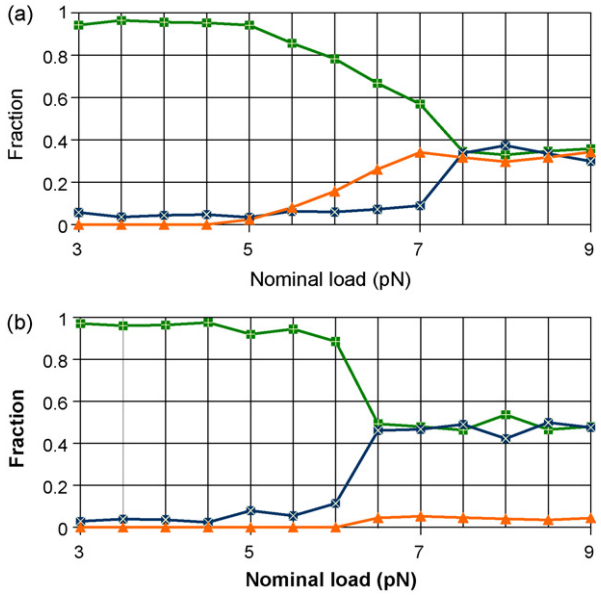

Fig. 5. Relationship between hindering load, stepping and detachment for model (a) without ATP gate; (b) with ATP gate. Green square with plus sign inset indicates fraction of forward steps, blue square with cross inset is fraction of backward steps, orange triangle is fraction of detachments in terms of total steps. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

Fig. 5 shows the result of varying hindering load on the modelled kinesin. The plot shows approximate equalisation of forward and backward steps at a load of 7.5 pN which is within the range of stall force (7–8 pN) reported by Nishiyama et al. (2002) and Carter and Cross (2005). The model shows the same trends of increased backstepping and detachments with increasing load above about 5 pN as reported by Nishiyama et al. (2002) (Fig. 5a, page 792). The results therefore favour the ungated model.

Intriguing behaviour in the presence of a non-hydrolysable analogue of ATP *in vitro* was discovered by Guydosh and Block (2006). They observed isolated backsteps during a long pause (up to several seconds) culminating in a final backstep before return to normal procession. They hypothesise that the backward linker strain caused by a backstep increases the probability that the analogue is released from the leading head to be replaced by ATP thus restarting normal procession. The model proposed in this paper predicts the pause because the analogue behaves like ATP, causing the linker to zipper so preventing the free head from reaching the rear site. Thus, at low load, kinesin is stuck in place on the microtubule with the leading head futilely hydrolysing ATP. Guydosh and Block used an optical force clamp to provide hindering loads of 4.5 pN and 5.3 pN. It is proposed here that fluctuations in the load would occasionally apply sufficient force to unzipper the linker in which case a backstep may occur though it is likely to be isolated and infrequent as they observed.

3.3. Contrary Evidence of Wait State Configuration

The model proposed here depends on the detached head being free to diffuse in the wait state. Evidence for a different configuration is therefore a challenge to the model. Yildiz et al. (2004) suggest that both heads are bound in the wait state. They labelled one head with a fluorophore and observed kinesin at low ATP concentration to extend the duration of the wait state. Alternating movement averaging 0 nm and 17 nm (the length of 2 tubulin dimers) was recorded corresponding to alternate stepping as would be expected with hand-over-hand motion. They suggest that this result also points to the wait state configuration being both heads bound since a free fluorescent head would introduce a further signal into the data. An alternative interpretation of the data depends on the lifetime of the wait state at the ATP concentration used in the experiment (340 nM). It is acknowledged that the stepping time (once ATP has bound) is much too short to register on the timescale of their image capture (0.33 s). Perhaps the wait state is also too short to affect the measurement: if the time that the fluorescent head is freely diffusing is much shorter than the image detection time then the signal from it will be lost in the noise.

Alonso et al. (2007) propose that one head is detached from the microtubule in the wait state but that it is not free to diffuse. They found that mixing kinesin with unpolymerised tubulin dimers caused only one head to bind in the absence of ATP. Their explanation is that the second head is parked, unable to bind, until released by the arrival of ATP. If this is the configuration of the wait state then clearly the model proposed in this paper is incorrect. A possible explanation for their data arises from the fact that, without cargo, kinesin is folded such that the tail inhibits normal procession (Cross and Scholey, 1999). The unbound head might then be effectively parked by the tail obscuring its tubulin binding site until ATP binding releases it. Thus the findings may only apply to kinesin in solution and not to kinesin pulling cargo. In any case, if the head were parked in the wait state during normal transport then it is difficult to see how backstepping could occur.

3.4. Blocked Kinesin and the ATP Hydrolysis Gate

A long-term goal of studying kinesin is to discover more about how transport fails since this is implicated in neurodegenerative disease such as Alzheimer's (Wilson, 2008a). A first step in this direction is to explore the effect of a blockage on the microtubule. In this study, confronting simulated processing kinesin with a blockage caused the molecule to detach and diffuse away from the microtubule. This behaviour is in line with the results of an *in vitro* study by Crevel et al. (2004) who found that, when confronted with an obstacle, kinesin detached after one hydrolysis cycle.

Contrary behaviour was observed by Seitz and Surrey (2006), however, who used a mutant kinesin that diffused away after stalling on the microtubule to provide a temporary blockage. Though native kinesin was slowed by the mutant there was little effect on procession distance (run length). They concluded that confronting kinesin with a temporary obstacle forces the motor into a wait state. The simulated kinesin diffuses away because the free head is prevented from reaching the next binding site by the barrier but the bound head hydrolyses ATP then detaches as it would during procession. Hancock and Howard (1999) compared the ATPase rate of native kinesin to that of a single-headed mutant: it was an order of magnitude faster. If kinesin behaves like the single-headed mutant when confronted by an obstacle then it would wait. An optional ATP gate was incorporated into the model whereby hydrolysis can be slowed by an order of magnitude unless both heads are bound to the microtubule with KT at the rear. A blockage prevents both heads from binding thus dramatically slowing hydrolysis and correspondingly increasing the duration of the wait state. If the blockage is removed before hydrolysis is complete, the free head binds the microtubule and the trailing head returns to the relatively fast hydrolysis characteristic of normal procession. This mechanism has the biologically satisfying consequence that kinesin waits at a temporary obstruction or snag but escapes a permanent one. This would make sense in terms of the efficiency of active transport since waiting at a blockage then continuing procession is faster than a diffusive search for a clear track (unless, of course, the wait is prolonged). If the blockage is long-term then the hydrolysis cycle will eventually complete, the head will detach from the microtubule and, since both heads are

then free, kinesin has a chance of diffusing around the obstruction.

On the other hand, switching the gate on has the destabilising effect of making kinesin more sensitive to timing variations at high linker strain (filled bars in Fig. 4) and load behaviour becomes less realistic as noted in Section 3.2. It would seem, therefore, that either there is no hydrolysis gate and so a different mechanism is in operation or kinesin does not wait at an obstacle.

4. Conclusion

A parsimonious model for the kinesin walk is proposed here that is capable of accounting for experimental evidence of backstepping. It is a modified form of the rectified Brownian motion model of Mather and Fox (2006) in which an ATP binding gate coordinates the heads. The new hypothesis is that no gating is necessary, that entropic neck linker strain is sufficient for procession. Theoretical support for this hypothesis comes from the results of computer simulation devised and implemented to investigate the model. Simulation results show that entropic neck linker strain is sufficient to coordinate the heads and that the ungated model also displays behaviour under load similar to that observed *in vitro*.

The simulation tool is currently being developed to respond more realistically to loading effects and to incorporate several kinesins in order to investigate crowding effects. The long-term aim is to increase the scope of the model to encompass aspects of axonal active transport and so assist in understanding failures in this system relevant to the early stages of neurodegeneration.

Acknowledgements

This work is supported financially by the UK Engineering and Physical Sciences Research Council. The author is grateful to Jacob Navia for supplying the software used to facilitate the writing of the simulation program (http://www.cs.virginia.edu/~lcc-win32/). My thanks go to colleagues (Sara Kalvala, Matthew Hodgkin and Brent Kiernan) and to reviewers for constructive comments on the manuscript.

References

Alonso, M.C., Drummond, D.R., Kain, S., Hoeng, J., Amos, L., Cross, R.A., 2007. An ATP gate controls tubulin binding by the tethered head of kinesin-1. Science 316, 120–123.
Asbury, C.L., 2005. Kinesin: world's tiniest biped. Curr. Opin. Cell Biol. 17, 89–97.
Carter, N.J., Cross, R.A., 2005. Mechanics of the kinesin step. Nature 435, 308–312.
Crevel, I.M., Nyitrai, M., Alonso, M.C., Weiss, S., Geeves, M.A., Cross, R.A., 2004. What kinesin does at roadblocks: the coordination mechanism for molecular walking. EMBO J 23, 23–32.
Cross, R., Scholey, J., 1999. Kinesin: the tail unfolds. Nat. Cell Biol. 1, 119–121.
Fox, R.F., Choi, M.H., 2001. Rectified Brownian motion and kinesin motion along microtubules. Phys. Rev. E 63 (051901), 1–12.
Gunawardena, S., Goldstein, L.S., 2004. Cargo-carrying motor vehicles on the neuronal highway: transport pathways and neurodegenerative disease. J. Neurobiol. 58, 258–271.
Guydosh, N.R., Block, S.M., 2006. Backsteps induced by nucleotide analogs suggest the front head of kinesin is gated by strain. Proc. Natl. Acad. Sci. 103 (21), 8054–8059.
Hackney, D.D., 1994. Evidence for alternating head catalysis by kinesin during microtubule-stimulated ATP hydrolysis. Proc. Natl. Acad. Sci. 91 (15), 6865–6869.
Hackney, D.D., 2007. Processive motor movement. Science 316, 58–59.
Hancock, W.O., Howard, J., 1999. Kinesin's processivity results from mechanical and chemical coordination between the ATP hydrolysis cycles of the two motor domains. Proc. Natl. Acad. Sci. 96 (23), 13147–13152.
Hirokawa, N., 1998. Kinesin and dynein superfamily proteins and the mechanism of organelle transport. Science 279, 519–526.
Howard, J., Hudspeth, A.J., Vale, R.D., 1989. Movement of microtubules by single kinesin molecules. Nature 342, 154–158.
Klumpp, L.M., Hoenger, A., Gilbert, S.P., 2004. Kinesin's second step. Proc. Natl. Acad. Sci. 101, 3444–3449.
Kolomeisky, A.B., Fisher, M.E., 2007. Molecular motors: a theorist's perspective. Annu. Rev. Phys. Chem. 58, 675–695.
Mather, W.H., Fox, R.F., 2006. Kinesin's biased stepping mechanism: amplification of neck linker zippering. Biophys. J. 91, 2416–2426.
Mori, T., Vale, R.D., Tomishige, M., 2007. How kinesin waits between steps. Nature 450 (29), 750–754.
Nishiyama, M., Higuchi, H., Yanagida, T., 2002. Chemomechanical coupling of the forward and backward steps of single kinesin molecules. Nat. Cell Biol. 4, 790–797.
Nogales, E., Whittaker, M., Milligan, R.A., Downing, K.H., 1999. High-resolution model of the microtubule. Cell 96, 79–88.
Ray, S., Meyhofer, E., Milligan, R.A., Howard, J., 1993. Kinesin follows the microtubule's protofilament axis. J. Cell. Biol. 121, 1083–1093.
Rice, S., Lin, A.W., Safer, D., Hart, C.L., Naber, N., Carragher, B.O., Cain, S.M., Pechatnikova, E., Wilson-Kubalek, E.M., Whittaker, M., Pate, E., Cooke, R., Taylor, E.W., Milligan, R.A., Vale, R.D., 1999. A structural change in the kinesin motor protein that drives motility. Nature 402, 778–784.
Rice, S., Cui, Y., Sindelar, C., Naber, N., Matuska, M., Vale, R., Cooke, R., 2003. Thermodynamic properties of the kinesin neck-region docking to the catalytic core. Biophys. J. 84, 1844–1854.
Rosenfeld, S.S., Fordyce, P.M., Jefferson, G.M., King, P.H., Block, S.M., 2003. Stepping and stretching. How kinesin uses internal strain to walk processively. J. Biol. Chem. 278, 18550–18556.
Roy, S., Zhang, B., Lee, V.M-Y., Trojanowski, J.Q., 2005. Axonal transport defects: a common theme in neurodegenerative diseases. Acta Neuropathol. 109, 5–13.
Schnitzer, M.J., Block, S.M., 1997. Kinesin hydrolyses one ATP per 8 nm step. Nature 388, 386–390.
Seitz, A., Surrey, T., 2006. Processive movement of single kinesins on crowded microtubules visualized using quantum dots. EMBO J. 25, 267–277.
Stokin, G.B., Lillo, C., Falzone, T.L., Brusch, R.G., Rockenstein, E., Mount, S.L., Raman, R., Davies, P., Masliah, E., Williams, D.S., Goldstein, L.S., 2005. Axonopathy and transport deficits early in the pathogenesis of Alzheimer's disease. Science 307, 1282–1288.
Svoboda, K., Schmidt, C.F., Schnapp, B.J., Block, S.M., 1993. Direct observation of kinesin stepping by optical trapping interferometry. Nature 365, 721–727.
Svoboda, K., Block, S.M., 1994. Force and velocity measured for single kinesin molecules. Cell 77, 773–784.
Uemura, S., Kawaguchi, K., Yajima, J., Edamatsu, M., Toyoshima, Y.Y., Ishiwata, S., 2002. Kinesin–microtubule binding depends on both nucleotide state and loading direction. Proc. Natl. Acad. Sci. 99 (9), 5977–5981.
Vale, R.D., 2003. The molecular motor toolbox for intracellular transport. Cell 112, 467–480.
Vale, R.D., Milligan, R.A., 2000. The way things move: looking under the hood of molecular motor proteins. Science 288 (5463), 88–95.
Wilson, R.J., 2008a. Towards a cure for dementia: the role of axonal transport in Alzheimer's disease. Sci. Prog. 91 (1), 65–80.
Wilson, R.J., 2008b. IEEE Proceedings: Second UKSIM European Symposium on Computer Modeling and Simulation. Simulating the kinesin walk: a small step towards understanding dementia, 226–231.
Yildiz, A., Tomishige, M., Vale, R.D., Selvin, P.R., 2004. Kinesin walks hand-over-hand. Science 303, 676–679.

Simulating the Kinesin Walk: a Small Step towards Understanding Dementia

Richard J. Wilson
MOAC Centre, University of Warwick, Coventry CV4 7AL, UK
richard.j.wilson@warwick.ac.uk

Abstract

Dementia results from neurodegeneration, a cause of which is the failure of axonal transport. Axonal transport is the systematic movement of vital cargo between the neuron cell body and the synapse. The engines powering this transport are motor proteins, molecular nanomachines. Kinesin-1 (conventional kinesin) is a motor protein that carries cargo to the synapse by walking along microtubule tracks. Its twin motor domains (heads) alternately bind the microtubule, derive energy from hydrolysing ATP, and step forward. This motion cannot be directly observed so the details are a matter of debate based on indirect experimental observations. A rule-based, spatial computer simulation has been built to better understand how kinesin walks. Results show a preference for rectified Brownian motion over a power stroke mechanism for normal stepping. A small step towards investigation of transport failure - placing a blockage on the microtubule - indicates a possible novel gating mechanism for kinesin.

1. Introduction

Macromolecules, vesicles and organelles need to be actively transported about the eukaryotic cell since diffusion is indiscriminate and ineffective. Active transport is especially important in neurons as they have extended projections, the longest being the axon which can reach over a metre [1]. Failure of axonal transport (active transport along the axon) is implicated in neurodegenerative diseases such as Alzheimer's [2]. Improving our understanding of the axonal transport system is therefore an important task in the programme to find a cure for these terrible diseases.

Long distance axonal transport is accomplished by kinesin and dynein motor proteins that bind cargo and ferry it unidirectionally along microtubule tracks [1]. Kinesin-1 (herein referred to as kinesin) is a member of the kinesin family of motor proteins that transfers cargo from the neuron cell body to the synapse. Local transport at the synapse is conducted by the myosin-actin system which is outside the scope of this paper.

Microtubules are hollow, 25 nm diameter rigid tubes, typically composed of 13 laterally bound filaments that provide linear tracks for the motors [3]. Filaments consist of heterodimers of the protein tubulin about 8 nm long that spontaneously bond end to end. A microtubule is a polar polymer with a ring of α-tubulins bounding the minus end and β-tubulins bounding the plus end [4].

Kinesin is a homodimer, a protein comprising 2 identical monomer parts, extending to about 70 nm in length [5]. The monomer is composed of a heavy chain and a light chain. The N-terminal region of the heavy chain forms a globular motor domain (head) of comparable size to tubulin. The head has two binding sites: one binds and hydrolyses the nucleotide ATP (adenosine triphosphate), the other binds a microtubule with nucleotide-dependent strength [6]. The head is connected by a short, flexible, single polypeptide neck linker to a long, coiled-coil stalk. The stalks form the dimer and, at their C-terminal region, combine with the light chains to form a fan-like tail which binds to cargo.

Kinesin walks along a microtubule filament in an asymmetric head-over-head (hand-over-hand) manner, similar to toeing a line [7]. It hydrolyses one ATP molecule at each step [8] and making hundreds of steps without detaching from the microtubule [9]. The direction of procession (continuous stepping) is determined by the orientation in which the heads bind to the microtubule and by zippering of the neck linker. A head binds to the microtubule in one direction, mainly to the β-tubulin [10] while the binding of ATP causes the linker to zipper to the head, pointing in the direction of motion thus conveying plus-end direction to the kinesin walk [11]. There are competing theories to explain how kinesin steps. Vale and Milligan [12] suggest a power stroke mechanism: zippering pulls the free head forward. Fox and Choi [13] consider that

978-0-7695-3325-4/08 $25.00 © 2008 IEEE
DOI 10.1109/EMS.2008.49

zippering rectifies the Brownian motion of the free head.

This paper reports a comparison of the power stroke and rectified Brownian motion models by varying the conditions necessary for head coordination in a computational simulation. Results support doubts about the power stroke model in terms of the energy of zippering [14] and physical properties at the nanoscale [13] as it proves much more sensitive to parameter variations and is thus a less likely route of motor protein development in evolutionary terms. The long-term goal of this research is to discover more about axonal transport dysfunction. As a first step, a barrier was placed on the microtubule preventing the motor from stepping. Both models of kinesin detached at the obstacle. Some laboratory experiments show that kinesin waits at a temporary obstacle [15]. It is proposed that this may be achieved through the operation of an ATP hydrolysis gate.

2. Modelling kinesin

Modelling of kinesin has primarily treated the molecule as a whole with the focus on accurately reproducing experimental biophysical data such as velocity response to load. Two main approaches have been used: Brownian ratchet or chemical-kinetic [16]. In the Brownian (or thermal) ratchet model a particle moves stochastically between potentials; in the kinetic model it moves through a series of chemical states linked by rate constants.

This study takes a more qualitative approach with the focus on how the components – the heads – interact to enable kinesin to walk. It is an attempt to formally investigate intuitive theory of the mechanism of the walk as derived from experiment. Elements of both the aforementioned approaches are employed, however, in that rectified Brownian motion is assumed in one simulation while in both simulations each head goes through the same series of chemical and binding states.

2.1. The simulation

A rule-based, two dimensional spatial simulation has been designed and implemented in the C programming language and runs on a standard laptop. The heads are bounded by a rectangular box representing part of an axon. A length of microtubule is laid out along the base of the box as alternate α- and β-tubulins. The heads are spatially constrained to simulate the physical neck linker connection between them and functionally constrained to simulate their interaction with nucleotide and microtubule.

A simulation run starts with kinesin in solution with both heads ADP-bound at the left end of the box close to the minus end of the microtubule. Constrained random motion is applied to each head to approximate diffusion until the microtubule is approached and kinesin engages with it. The simulation run is terminated when the motor reaches the plus end of the microtubule or procession fails. One run of the program consists of a series of simulation runs, one for each combination of parameter values thus exhaustively exploring all combinations of parameter values. The results are output to file for analysis.

In order to show the progression of the simulation, the state of the system is displayed in a graphic window. The activity of the motor is displayed at the top of the screen while a results summary is plotted underneath. The experimenter can thus keep a visual check on the system's behaviour.

2.2. Theory of kinesin procession

There is general agreement about the overall mechanism of procession though the details are a matter of debate. Entry into the cycle is initiated when kinesin in solution (both heads ADP-bound) encounters the microtubule. One head binds to the microtubule which causes the head to release its ADP while the other head remains free [17]. ATP binds the nucleotide-free head causing its neck linker to zipper to the head [11] which results in the free head binding to the next microtubule binding site. This in turn causes ADP release followed by ATP binding and hydrolysis. Meanwhile the other head hydrolyses ATP, releases phosphate and detaches. Thus each head alternately steps forward: kinesin walks along the microtubule.

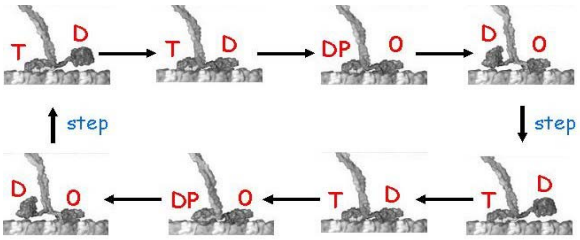

Figure 1. Kinesin procession as a series of states or snapshots. The red capital letters placed above the heads indicate which nucleotide is bound to a head: D for ADP, T for ATP, DP for hydrolysed ATP, 0 for none.

The procession cycle is illustrated in figure 1. Each of the 8 snapshots diagram the lower part of kinesin traversing a short section of microtubule which is shown as a line of alternate light and dark blobs (adapted from Vale and Milligan [12]). At the top left and bottom right of the figure, the free head has stepped forward after ATP has bound to its partner.

The free head then binds the microtubule and releases ADP while its companion hydrolyses ATP. The trailing head detaches after release of phosphate ready for the next step. Kinesin is moving to the right.

This mechanochemical cycle is used as a basis for the simulation. The cycle is deterministic with reverse transitions being ignored. Unlike the rotary motor ATP-synthase, kinesin does not produce ATP when run in reverse: both backward stepping and forward stepping are accompanied by ATP hydrolysis [18].

2.3. Stepping models

Two models of stepping are implemented: power stroke (PS) and rectified Brownian motion (RBM). What distinguishes them is the condition under which stepping occurs. PS stepping occurs following binding of ATP to the fixed head, the free head is propelled forward by zippering. In RBM stepping, the binding of ATP sets up zippering but zippering only occurs after the free head diffuses forward pulling the neck linker with it.

2.4. Head model

The heads are modelled as separate entities and the coordination of their mechanochemical cycles is not imposed as a constraint in the simulation: procession is an emergent phenomenon contingent on head coordination. Each head is modelled as an event-driven finite state machine (FSM). The theoretical maximum possible number of states of the FSM is 8 since a head has 4 states of nucleotide binding and 2 states of microtubule binding. Not all these states are physically realistic. Experimental evidence shows that the ADP-bound head has weak affinity for the MT otherwise it has a strong affinity for the MT [6]. The number of states modelled is therefore 5: KD, K0, KDP, KT and KDu which represent, respectively, a kinesin head bound to ADP, to no nucleotide, to ATP, or to hydrolysed ATP, (all bound to the MT) and the ADP-bound head unbound to the MT.

2.5. Rules

The following set of rules embodies the hydrolysis cycle and the interaction between individual head and microtubule and thus defines the events used to drive the FSMs:

1) If an ADP-bound head encounters the microtubule, it binds (KDu → KD)
2) Binding to the microtubule causes ADP release (KD → K0)
3) ATP binds the empty head (K0 → KT)
4) The bound head hydrolyses ATP (KT → KDP)
5) Head detachment occurs with phosphate release (KDP → KDu)
6) If the bound head makes the K0 -> KT transition and the other head is unbound then a step is taken.

Rule 6 is used in the PS model. For the RBM model, rule 6 is modified: the step is taken if one head is free and the bound head is in state KT or KDP.

2.6. Event timing

The timings of the transitions from one procession state to the next are parameterised. This enables investigation of the effect of changing the relative timing of the events on the behaviour of the motor. There are 4 parameters corresponding to the 4 transitions:

- Time taken to hydrolyse ATP (KT -> KDP)
- Time taken to detach (KDP -> KDu)
- ADP release time (KD -> K0)
- ATP binding time (K0 -> KT).

The parameters take positive integer values where the higher the value the longer the head stays in the state i.e. the longer the transition delay. The same set of parameters is applied to each head since kinesin is a homodimer (the heads are physically the same).

2.7. Neck linkers

The point in the procession cycle where one head is free and the bound head is nucleotide free is called the wait state as the molecule is waiting for ATP. The movement of the free head in the wait state is presumed to be diffusive and is simulated by a pseudo-random number function such that there is an equal probability of the head moving forwards or backwards.

The extent of the movement of the free head and thus the probability of binding the microtubule is influenced by the neck linkers. Entropic strain causes the linkers to act like springs [19]. In order to investigate the effect of the linkers on the behaviour of the molecule, the amount of strain is parameterised in the range 0 to 10 where 0 is no strain and 10 is maximum strain. At maximum linker strain, the free head cannot reach either binding site. For lower values of strain, the probability of binding is inversely proportional to the value of the strain, again with equal probability of binding forwards or backwards. The physical basis of the probability function is that the stronger the springs of the neck linkers are, the less likely the free head will bind either site so that, in the limit, the head would diffuse without binding.

2.8. Barrier

An obstacle can be placed towards the plus end of the microtubule for a given time interval. The obstacle acts as a barrier preventing kinesin from stepping forward to the next binding site.

3. Results and discussion

The power stroke (PS) and the rectified Brownian motion (RBM) models of stepping were compared by varying the timing parameters and the neck linker strain. There are 3 possible phases to the system: procession, stuck or diffusion. Kinesin either walks along the MT to the plus end (procession phase) or engages with the MT but then remains stuck in position (stuck phase) or fails to process and disengages from the MT (diffusion phase).

3.1. Power stroke

With linker strain, varying the timing parameters revealed a 2 phase system: procession or diffusion. A different phase map resulted from removing linker strain: procession did occur but the majority of timing combinations caused kinesin to get stuck.

The latter case is amenable to a simple analysis. The time taken for one head to get from one step to the next must equal the time taken for the other head to do the same (the dwell times are the same). Consider the first half of the cycle depicted in figure 2A (time 0 to 3). One head (K1) hydrolyses ATP, releases phosphate and detaches from the microtubule. In parallel, the other head (K2) docks to the microtubule, releases ADP then ATP binds. Thus the time taken for ATP hydrolysis (T1) plus phosphate release and head detachment (T2) must be the same as the time for head binding (T3) plus ADP release (T4) plus ATP binding (T5). Summing and equating these timing parameters gives the equality:

$$T1 + T2 = T3 + T4 + T5$$

This equation precisely defines the timings of the power stroke procession phase at zero neck linker tension. Though this is an unrealistic situation, it is interesting to note that the lowest timing values are consistent with Ma and Taylor's estimates where T2 is about twice the value of the other parameters [20].

3.2. Rectified Brownian motion

Unlike PS kinesin, RBM kinesin processed under all timing combinations at high values of linker strain. Reducing the linker strain resulted in a 2 phase system: procession or stuck. The relation between the parameters and the phase has not so far proved amenable to analysis.

3.3. Model comparison

Figure 2 plots the relationship between neck linker strain and timing combinations giving interrupted procession (failure to continuously step from one end of the microtubule to the other). The RBM model shows procession for the whole timing range at linker strain values above 7 whereas even at maximum strain the PS model fails to process under some timing conditions.

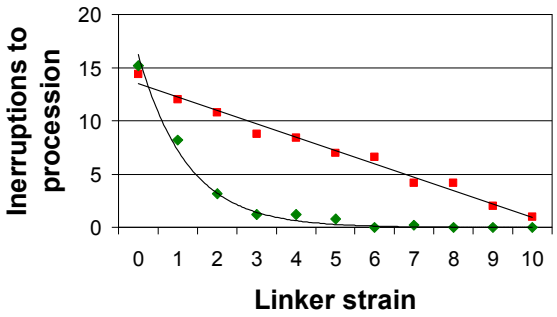

Figure 2. Relationship between neck linker strain and procession. Green diamonds are RBM values, red squares are power stroke values. A linear trend line is fitted to the power stroke data, an exponential trend line is fitted to the RBM data.

Figure 3 plots the average number of detachments of kinesin from the microtubule per run over the range of linker strain. The RBM model shows no detachments until the strain is below 2 whereas procession is interrupted during runs of the PS model over the whole range of strain.

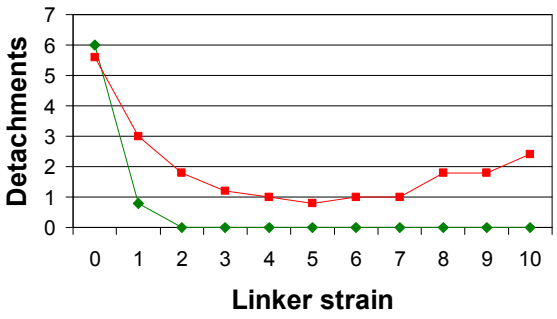

Figure 3. Relationship between neck linker strain and detachment of kinesin from the microtubule. Green diamonds are RBM values, red squares are power stroke values.

Svoboda and Block [21] observed a small proportion of backsteps during normal procession *in*

vivo. Setting the linker strain to 9.5 results in a similar proportion of backsteps *in silico* which puts the processive behaviour of RBM kinesin well within realistic parameters.

In summary, both models can support procession *in silico* but procession emerges from the RBM model over a much greater range of timings and strains than the PS model. This result favours RBM on evolutionary grounds as evolution tends to select robust mechanisms allowing wide parameter variations to not adversely affect the functioning of a system [21].

3.4. Axonal transport failure

Axonal transport failure is implicated in the neurodegenerative process [22] and there is evidence that it is an early event in the development of the most prevalent dementia, Alzheimer's disease [23].

As an initial attempt to look at dysfunctional transport, the effect of a blockage on the microtubule on kinesin has been investigated in the present study. Both PS and RBM kinesin detached and diffused away from the microtubule within the timing of a processionary cycle. This happens because the free head tries to step forward but is blocked while the bound head hydrolyses ATP then detaches as it would during procession.

There is conflicting *in vitro* experimental evidence concerning what happens when kinesin is confronted by an obstacle. Crevel *et al*. found that kinesin detaches from the microtubule after one hydrolysis cycle [24]. Seitz and Surrey, however, found that the motor waits at an obstacle for an order of magnitude longer than the normal hydrolysis cycle and then resumes processing after the obstacle is removed [15].

Evidence supporting the waiting behaviour comes from Hancock and Howard who found that twin-headed kinesin's ATP-ase rate is an order of magnitude faster than that of single headed kinesin [25]. They suggest that forward neck linker strain is required for the fast ATP hydrolysis characteristic of normally processing kinesin. Thus if the free head is prevented from binding the microtubule then hydrolysis is slowed. If the blockage is removed before the slow hydrolysis is complete, the free head binds the MT at the forward binding site and normal procession resumes. Thus kinesin waits at a temporary obstacle but escapes a permanent obstruction.

To make simulated kinesin wait in this manner, a new rule was added so that the rate of ATP hydrolysis is slowed by a factor of ten unless both heads are bound. Besides the waiting behaviour, the overall effect of adding this rule is to enhance head coordination.

3.5. Future work

The full impact of adding the new rule is being assessed in concert with investigating varying loads on the molecule (cargo effects have been ignored in the present work). It is planned to extend the simulation to encompass more elements of the axonal transport system and thus provide a more comprehensive window on its normal operation and its role in neurodegeneration.

4. Conclusion

This research was motivated by a desire to better understand the role of axonal transport in dementia. The motor protein kinesin is an important engine of axonal transport and the main focus of this study is kinesin's walking mechanism.

Two models of the kinesin walk have been compared under a range of timing and linker strain values to discover under what conditions the heads coordinate such that procession occurs. The rectified Brownian motion model has proven more robust than the power stroke model and would therefore seem to be the more likely candidate for the kinesin procession mechanism on evolutionary grounds. This conclusion confirms doubts about the power stroke mechanism expressed in terms of insufficient energy of zippering [14] and the domination by viscous and thermal forces of the nanoscale [13].

The effect of confronting kinesin with a blockage on the microtubule has also been investigated as a first step towards understanding dysfunctional transport. Results indicate that there may be an ATP hydrolysis gate operating to keep kinesin waiting at a temporary blockage. Whether or not this is the case can only be discovered in the laboratory.

Qualitative simulation is unable to provide definitive solutions but the author believes that it has an important place in firming up theoretical mechanisms of biological processes and can suggest areas for laboratory experiment. It is hoped that increasing the scope of the simulation to encompass more aspects of axonal transport will prove of significant value in understanding normal neuron function and relevant to understanding the early stages of neurodegeneration.

5. Acknowledgements

This work is supported by the UK Engineering and Physical Sciences Research Council. The author is grateful to Jacob Navia for supplying the lcc-win32 compiler software (http://www.cs.virginia.edu/~lcc-

win32/) and to colleagues for constructive comments on the structure of the paper.

6. References

[1] N. Hirokawa, "Kinesin and dynein superfamily proteins and the mechanism of organelle transport," *Science*, vol. 279, pp. 519-26, 1998.

[2] R. J. Wilson, "Towards a cure for dementia: the role of axonal transport in Alzheimer's disease," *Sci Prog*, vol. 91, pp. 65-80, 2008.

[3] S. Ray, E. Meyhofer, R. A. Milligan, and J. Howard, "Kinesin follows the microtubule's protofilament axis," *J Cell Biol*, vol. 121, pp. 1083-93, 1993.

[4] E. Nogales, M. Whittaker, R. A. Milligan, and K. H. Downing, "High-resolution model of the microtubule," *Cell*, vol. 96, pp. 79-88, 1999.

[5] R. D. Vale, "The molecular motor toolbox for intracellular transport," *Cell*, vol. 112, pp. 467-80, 2003.

[6] S. Uemura, K. Kawaguchi, J. Yajima, M. Edamatsu, Y. Y. Toyoshima, and S. Ishiwata, "Kinesin-microtubule binding depends on both nucleotide state and loading direction," *Proc Natl Acad Sci U S A*, vol. 99, pp. 5977-81, 2002.

[7] C. L. Asbury, "Kinesin: world's tiniest biped," *Curr Opin Cell Biol*, vol. 17, pp. 89-97, 2005.

[8] M. J. Schnitzer and S. M. Block, "Kinesin hydrolyses one ATP per 8-nm step," *Nature*, vol. 388, pp. 386-90, 1997.

[9] J. Howard, A. J. Hudspeth, and R. D. Vale, "Movement of microtubules by single kinesin molecules," *Nature*, vol. 342, pp. 154-8, 1989.

[10] K. Hirose, A. Lockhart, R. A. Cross, and L. A. Amos, "Nucleotide-dependent angular change in kinesin motor domain bound to tubulin," *Nature*, vol. 376, pp. 277-9, 1995.

[11] S. Rice, A. W. Lin, D. Safer, C. L. Hart, N. Naber, B. O. Carragher, S. M. Cain, E. Pechatnikova, E. M. Wilson-Kubalek, M. Whittaker, E. Pate, R. Cooke, E. W. Taylor, R. A. Milligan, and R. D. Vale, "A structural change in the kinesin motor protein that drives motility," *Nature*, vol. 402, pp. 778-84, 1999.

[12] R. D. Vale and R. A. Milligan, "The way things move: looking under the hood of molecular motor proteins," *Science*, vol. 288, pp. 88-95, 2000.

[13] R. F. Fox and M. H. Choi, "Rectified Brownian motion and kinesin motion along microtubules," *Phys Rev E Stat Nonlin Soft Matter Phys*, vol. 63, pp. 051901, 2001.

[14] S. Rice, Y. Cui, C. Sindelar, N. Naber, M. Matuska, R. Vale, and R. Cooke, "Thermodynamic properties of the kinesin neck-region docking to the catalytic core," *Biophys J*, vol. 84, pp. 1844-54, 2003.

[15] A. Seitz and T. Surrey, "Processive movement of single kinesins on crowded microtubules visualized using quantum dots," *Embo J*, vol. 25, pp. 267-77, 2006.

[16] A. B. Kolomeisky and M. E. Fisher, "Molecular motors: a theorist's perspective," *Annu Rev Phys Chem*, vol. 58, pp. 675-95, 2007.

[17] D. D. Hackney, "Evidence for alternating head catalysis by kinesin during microtubule-stimulated ATP hydrolysis," *Proc Natl Acad Sci U S A*, vol. 91, pp. 6865-9, 1994.

[18] N. J. Carter and R. A. Cross, "Mechanics of the kinesin step," *Nature*, vol. 435, pp. 308-12, 2005.

[19] W. H. Mather and R. F. Fox, "Kinesin's biased stepping mechanism: amplification of neck linker zippering," *Biophys J*, vol. 91, pp. 2416-26, 2006.

[20] Y. Z. Ma and E. W. Taylor, "Interacting head mechanism of microtubule-kinesin ATPase," *J Biol Chem*, vol. 272, pp. 724-30, 1997.

[21] H. Kitano, "Biological robustness," *Nat Rev Genet*, vol. 5, pp. 826-37, 2004.

[22] S. Roy, B. Zhang, V. M. Lee, and J. Q. Trojanowski, "Axonal transport defects: a common theme in neurodegenerative diseases," *Acta Neuropathol*, vol. 109, pp. 5-13, 2005.

[23] G. B. Stokin, C. Lillo, T. L. Falzone, R. G. Brusch, E. Rockenstein, S. L. Mount, R. Raman, P. Davies, E. Masliah, D. S. Williams, and L. S. Goldstein, "Axonopathy and transport deficits early in the pathogenesis of Alzheimer's disease," *Science*, vol. 307, pp. 1282-8, 2005.

[24] I. M. Crevel, M. Nyitrai, M. C. Alonso, S. Weiss, M. A. Geeves, and R. A. Cross, "What kinesin does at roadblocks: the coordination mechanism for molecular walking," *Embo J*, vol. 23, pp. 23-32, 2004.

[25] W. O. Hancock and J. Howard, "Kinesin's processivity results from mechanical and chemical coordination between the ATP hydrolysis cycles of the two motor domains," *Proc Natl Acad Sci U S A*, vol. 96, pp. 13147-52, 1999.

Science Progress (2008), 91(1), 65–80 doi: 10.3184/003685008X285375

Towards a cure for dementia: the role of axonal transport in Alzheimer's disease

Richard J. Wilson

ABSTRACT

Alzheimer's disease is an incurable, fatal illness characterised by years of progressive mental decline. It afflicts half a million people in the UK – more than any other dementia. The primary risk factor is old age so this number is rising as we live longer. Current treatment is palliative while more potent drugs have encountered problems during clinical trials. It is known that the disease results from brain deterioration associated with the formation of microscopic lesions. Genetic mutations cause a small minority of cases but our knowledge of the underlying biological mechanisms is limited. The key to improved understanding may be a process vital to brain cells called axonal transport. Disruption of axonal transport seems to be an early event in the progression of the disease and is linked to lesion formation and brain dysfunction so a full investigation of this process should lead to a cure, if not prevention.

Keywords: *Alzheimer's disease, axonal transport, kinesin, microtubule, amyloid hypothesis, tau hypothesis*

Richard Wilson is a final year PhD research student. Originally from a computer science background, he gave up studying pattern recognition in artificial neural networks for an IT career. Currently investigating motor proteins and axonal transport, his long term aim is to make a significant contribution to understanding ageing neurons and neurodegeneration. He may be contacted at
E-mail: richard.j.wilson@warwick.ac.uk

Alzheimer's disease: a growing scourge

The population is increasing and we are living longer but not necessarily healthier lives: more of us are suffering disease and infirmity in old age. One particularly distressing set of late-onset diseases is dementia. It is estimated that over 20 million people suffer dementia worldwide, about 700,000 in the UK. The serious impact of dementia on families and society is increasing as these Figures are expected to double every 20 years. In the UK, the current cost of care alone is calculated at over £17B per annum while the annual death toll is over 60,000. The majority of dementia sufferers have Alzheimer's disease (AD)[1].

Following initial diagnosis of AD, the individual endures (on average) 8 years of increasingly distressing and decreasingly manageable symptoms as their brain decays. The billions of specialised cells of the brain that enable our cognitive faculties are called neurons (or neurones). They interconnect via synapses to form the complex neural network of the brain. The disease initially destroys synapses then kills neurons; the damage spreads out from the interior of the brain to the surface: from the basal forebrain through the hippocampus to the cortex[2]. It is estimated that neurodegeneration starts more than 20 years before symptoms become apparent[3].

The first symptoms, notably abnormal memory deficit, are accompanied by a reduction in the normal production of the neurotransmitter acetylcholine. Neurotransmitters are the chemicals that nerve cells use to communicate with one another. Drugs have been developed to compensate for this decline but Mount and Downton[4] note that "...none of the currently approved drugs stops the underlying degeneration of brain cells or reverses the progression of Alzheimer disease."

This paper outlines our current knowledge of AD, recent advances towards a cure, and the role of axonal (axoplasmic) transport in normal and diseased brains.

Two types of Alzheimer's

There are two recognised types of AD: familial and sporadic

The familial form of AD is estimated to account for less that 5% of all AD cases and is normally early-onset *i.e.* most cases present before the age of 65. It is identified with specific genetic mutations affecting amyloid precursor protein (APP), presenilin-1 (PS1), and

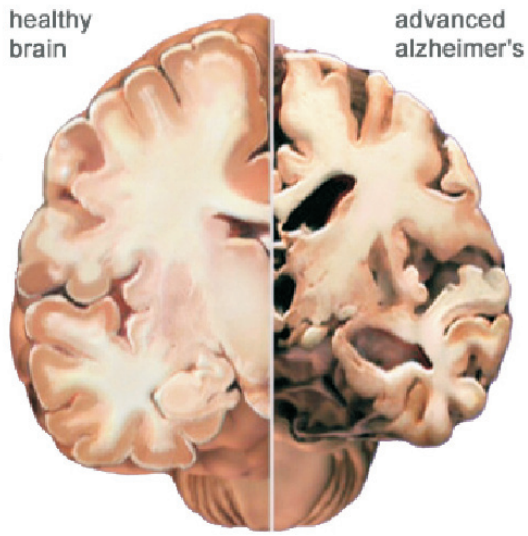

Fig. 1. *Brain section comparison.*

presenilin-2; most cases have the PS1 mutation. The main constituent of senile plaques, amyloid-β, is produced by the sequential cleaving of APP by β-secretase and γ-secretase. Presenilin is a component of γ-secretase. Faulty processing of APP is therefore suspected of causing familial AD[3].

The common, sporadic form of AD is late-onset: incidence rises steeply with age after 65. Sporadic AD doesn't correlate with any of the familial-form genetic mutations though there is a genetic link: a mutation in the gene for apolipoprotein E that is thought to compromise the protein's neuroprotective function. This mutation, however, is not a determinant but a risk factor for the disease; the cause of sporadic AD remains to be discovered[3].

Shrunken brain with lesions

Dementias result from the dysfunction, degeneration and loss of neurons in the brain. The most obvious characteristic of a post-mortem AD brain is massive neural atrophy (Figure 1).

Microscopic examination of stained sections of AD brain reveals the presence of myriad extracellular senile plaques and intracellular neurofibrillary tangles, significantly more than observed in normal brains of the same age[5]. The processes governing the formation of these inclusions, the toxicity of the various intermediates in their

formation and their mechanisms of damage are under intensive study.

The presence of plaques and tangles has given rise to two hypotheses as to the cause of this devastation: the amyloid hypothesis and the tau hypothesis respectively. To what extent either hypothesis is correct remains to be demonstrated.

Amyloid hypothesis

The amyloid hypothesis proposes that AD is caused by genetic mutations or environmental factors which favour the production of amyloid-β (Aβ) protein. The consequent excess Aβ aggregates into fibrils that accumulate as senile plaques. Some or all of these forms of Aβ are toxic so the result is the cascade of neural dysfunction and loss underlying dementia[6].

Glenner proposed that amyloid was at the heart of the disease following his group's discovery that Aβ is the main constituent of plaques[7]. As noted above, Aβ is produced from the sequential cleaving of APP by β-secretase followed by γ-secretase. There is an alternative proteolytic pathway that does not produce Aβ whereby APP is first cleaved by α-secretase. Little is known about the control of these pathways.

The incidence of plaques is not well correlated with the progression of AD[5] so the toxicity of intermediate forms of Aβ has been extensively investigated in the laboratory. A study of rhesus monkeys and rats compared the effects of microinjection of soluble or fibrillar Aβ (at a similar concentration to that found in plaques) into the cerebral cortex of young and old animals[8]. Only fibrillar Aβ caused extensive neuron loss, tau phosphorylation, and microglial proliferation (microglia are the brain's immune cells) and then only in older monkeys. It seems that rats are too short-lived to show a significant response and that, in higher mammals, young neurons are either protected from or dispose of the toxic fibrils. In a study of a mouse model expressing human familial APP, the physiological changes accompanying behavioural deficits were found to be intracellular Aβ aggregate deposition; plaque formation occurred later[9].

This evidence would seem to support the amyloid hypothesis but an alternative interpretation is that Aβ has a neuroprotective role. The bioflocculant hypothesis[10] proposes that its function is to bind toxins and form plaques to sequester neurotoxic solutes in a form inducing microglial phagocytosis (disposal by engulfing). Both hypotheses could be right: disposal of toxins may be the normal

function of Aβ but, perhaps, the process becomes dysfunctional in the ageing neurons of susceptible subjects. An intriguing possibility is that such a response may be induced by viral infection or reactivation of a dormant virus[11].

Search for an amyloid treatment

The amyloid hypothesis has prompted considerable effort to find a cure, the most advanced being targeted on developing a vaccine. In the late 1990s experiments using the AD mouse model showed positive results for both active vaccination (injecting Aβ or fragments thereof) and passive vaccination (injecting antibody to Aβ) [12]. Reduction in amyloid accumulation and plaque removal were reported. Further positive results followed but, in 2001, phase 2 clinical trials of an active vaccine were stopped because several subjects developed encephalitis. Work continues to develop a safe vaccine.

Alternative approaches to reducing or eliminating Aβ are also being pursued. A potential treatment based on enhancing Aβ degradation[13] has entered clinical trials. Another potential treatment may enable Aβ removal from the blood stream. A faulty protein mediating Aβ clearance from the mouse brain into the blood stream has been found[14]. An unlikely β-secretase antagonist has recently been discovered. In mouse cell cultures, normally functioning prion protein has been found to regulate β-secretase and thus Aβ production[15]. Mutant versions of prion were found to have no regulatory effect indicating a possible connection between prion diseases and AD. This study raises the exciting possibility that a single cure might be found for both diseases.

Tau hypothesis

The tau hypothesis proposes that abnormal tau disrupts axonal transport by microtubule destabilisation and physical blockage through aggregating into NFTs. This disruption compromises the normal functioning of the neuron, leads to axonal atrophy, cell death and thus to dementia[16].

Tau is a microtubule associated protein that, when normally phosphorylated, binds to and helps stabilise microtubules (MTs) in the axons of neurons. Neurofibrillary tangles (NFTs) are mainly composed of filaments of hyperphosphorylated tau (h-tau) protein, a form of tau that doesn't bind MTs. A cell culture study showed that addition of h-tau from AD brains caused sequestration of normal tau, MT disassembly and inhibited formation of MTs[17]. It

is not clear what triggers the production of abnormal tau but a complex picture has arisen whereby sequential phosphorylation of multiple sites is required to produce h-tau that both fails to bind to MTs and forms fibrils[18].

The incidence of NFTs correlates with the progression of AD[5] so the question arises as to whether NFTs are toxic or are markers of toxicity. To find out, researchers used a mouse model expressing human tau that develops progressive age-related NFT deposition, neuronal loss and memory impairments[19]. Switching off tau production stopped neuron loss and memory recovered though NFTs continued to form. It was concluded that NFTs are not sufficient to cause cognitive decline or neuronal death and could be part of a protective response. As the insoluble tau aggregate grows to dominate the cell body, however, interference in normal function is inevitable but study of this process is outside the scope of mouse models given the animal's short lifespan.

Search for a tau treatment

One line of investigation has been to try to stop NFT formation, the focus being on kinases (enzymes that phosphorylate proteins) and phosphatases (enzymes that remove phosphate). Glycogen synthase kinase-3β (GSK3β) has been shown to induce tau phosphorylation, NFTs, synaptic loss and neuronal death, resulting in cognitive impairment in animal models[20]. A possible treatment would therefore be to inhibit GSK3β. In the cell culture study cited above[17], MT disruption by h-tau was found to be reversible by introducing a phosphatase. A major problem with developing drugs along these lines is that these enzymes are active in many pathways so any treatment would have to be made selective of their action on tau. A possible further problem arises if NFT formation is neuroprotective: treatment would then prove counter-productive.

Another line of research is aimed at restabilising MTs. A study investigating the possibility of a replacement therapy gave positive results in mice expressing human tau protein[21]. The mice produced excess tau that caused progressive motor impairment with a reduction in the number of MTs, reduced axonal transport, and axonal degeneration. A series of injections of paclitaxel (an MT stabilising compound) ameliorated these deficits. Unfortunately such compounds cannot be used to treat AD either because they lack selectivity, or are resistant to crossing the blood-brain barrier or show toxicity[16].

Axonal transport and Alzheimer's

There is increasing evidence that breakdown of axonal transport is a key event in the neurodegenerative process[22]. Study of axonal transport and its failure holds out the promise of effective treatment for diseases such as AD. There is also evidence that failure of axonal transport occurs early in the pathology of AD[23]. An early diagnosis would give clinicians a wider scope for treatment and an increased probability of cure when suitable drugs become available.

Importance of axonal transport to neurons

Proteins, vesicles (hollow lipid containers) and organelles are transported in a timely and orderly fashion within a cell by specialised proteins operating on cytoskeletal tracks. In the axon of a neuron, this process is known as axonal transport. A neuron has a number of projections emanating from its cell body: many dendrites and a long, thin axon. Input signals are received by the dendrites, processed by the neuron and output signals sent along the axon which branches out at its tip to connect to as many as thousands of dendrites of other neurons. The length of the axon makes axonal transport especially important to neurons as vital cargo has a significantly increased distance to travel compared to the extent of other types of cell.

The functional importance of axonal transport to neurons can be appreciated by considering inter-neuron communication. In response to sufficient excitatory stimulation via the dendrites, neurons "fire": they generate electrical impulses (action potentials) that travel down the axon to influence the activity of the following (post-synaptic) neurons via synapses at the axon terminal. Most neurons communicate via chemical synapses. The arrival of the electrical impulse at the synapse triggers release of neurotransmitter which diffuses across the synaptic gap to bind to receptors in the dendritic membrane of the post-synaptic neuron thus modulating the firing of that neuron. Some neurotransmitter is recycled but some is lost by being broken down in the synaptic gap. Replacement neurotransmitter is manufactured in the cell body and transported to the synapses by axonal transport. Maintenance of axonal transport is therefore crucial to the proper functioning of a neuron not least by replenishing the supply of neurotransmitter.

Axonal transport – a complex system

Axonal transport comprises long-distance and local transport of proteins, vesicles and organelles within the axon of the neuron by motor proteins moving along protein polymer tracks. Local transport – the actomyosin system in which myosin motors traverse actin filaments – will not be discussed here. For a stimulating introduction to this subject in the context of the molecular biology of the cell see Alberts *et al.*[24]. Figure 2 diagrams some of the complexity of active transport in a neuron[25]. The framework for long-distance transport is the microtubule cytoskeleton.

Microtubules

Microtubules (MTs) are rigid, tubular assemblies of (typically) 13 protein filaments that associate side by side but slightly offset giving the tube a spiral shape some 25 nm in diameter. Filaments assemble from tubulin heterodimers (comprising two different proteins) that bind head-to-tail such that the α-tubulin protein alternates with its β-tubulin partner. MTs are polarised in the sense that the face of one end is a ring of α-tubulins (the minus end) while the other face is a ring of β-tubulins (the plus end). The axon contains a series of bundles of MTs all oriented with their plus ends farther away from the cell body than their minus ends (see Figure 2).

Motor proteins

Each MT filament provides a linear path on which two families of motor protein can travel: kinesins and dyneins. Most kinesins move towards the MT plus end *i.e.* away from the cell body (known as anterograde motion) whereas dyneins move in the opposite direction (retrograde motion). Cargo such as vesicles containing neurotransmitter is transported in one direction though mitochondria move in both directions pausing where ATP is needed. ATP (adenosine triphosphate) is an energy-rich molecule synthesised in mitochondria and widely used to fuel cellular processes including axonal transport. A mitochondrion binds kinesin, dynein and myosin motors and regulation of its movement involves phosphorylation but how coordination between the motors is achieved has yet to be determined[26].

Kinesins and dyneins share similar motor domains that hydrolyse ATP and alter conformation to variably associate with MTs.

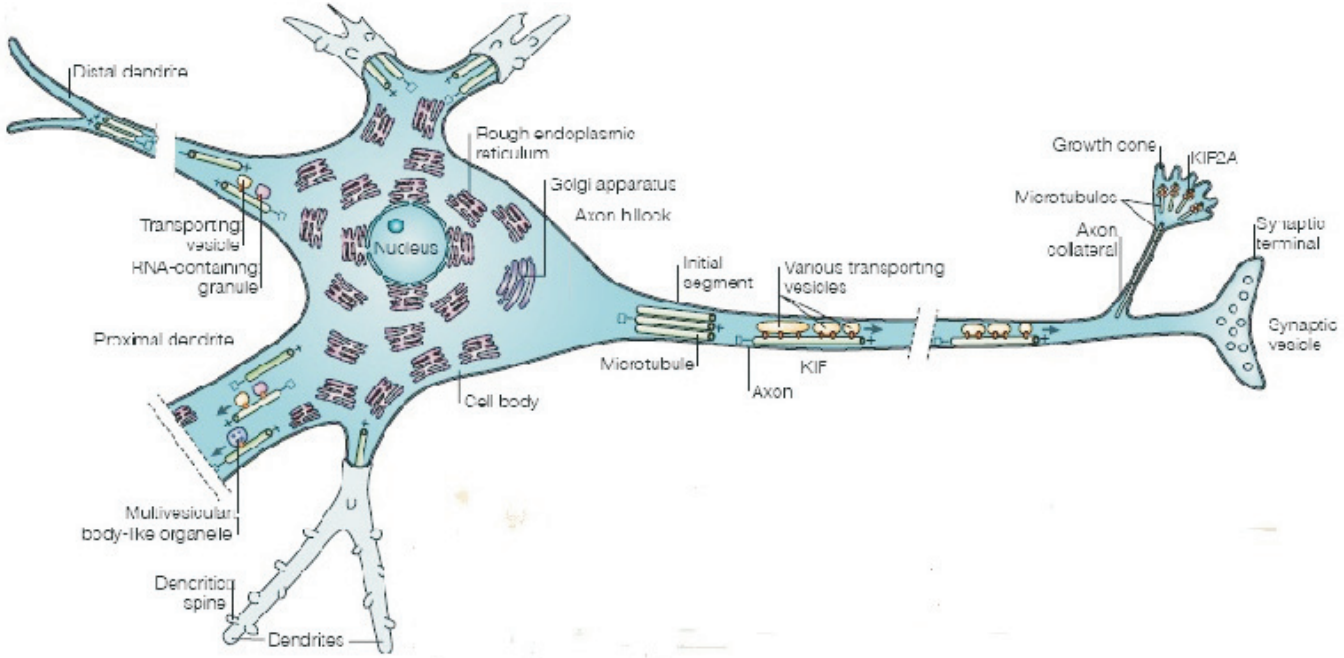

Fig. 2. *Diagram of typical neuron showing dendrites (left) and a long, thin axon (right) in which the microtubules are unipolar. Kinesins (KIFs) are shown transporting vesicles towards the synaptic terminal. (from Hirokawa and Takemura[25]).*

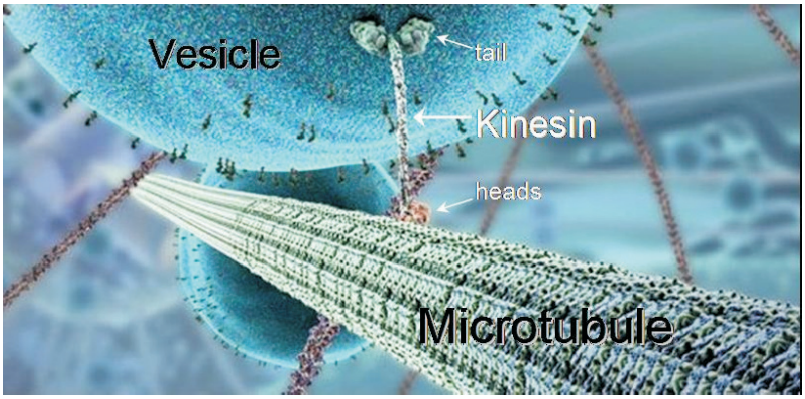

Fig. 3. *Depiction of kinesin pulling vesicle along microtubule (from Viel, Lue and Liebler[29]).*

The coordinated action of the motor domains enables motor proteins to reliably pull cargo through the crowded cytosol. As the motor is the dynamic heart of axonal transport, we shall take a closer look at the structure of a typical motor and how it functions.

Kinesin – a two-headed nanomachine

Conventional kinesin (kinesin I) is a representative example of the kinesin family of motor proteins. Its structure is a fan-like, cargo-binding tail connected by a long coiled-coil stalk to two short, flexible neck linkers terminating in the globular motor domains (heads). It measures about 70 nm from heads to tail when active. The stalk has a hinge region about which the protein jack-knifes when inactive. Each head has two binding sites: one for MTs, the other for nucleotide. During cargo transportation the nucleotide site performs hydrolysis of ATP (energy-providing removal of a phosphate to yield ADP – adenosine diphosphate). Members of the family share a common head structure and function[27].

The details of the movement mechanism are a matter of debate but it has been established that kinesin "walks" in a head-over-head (or hand-over-hand) fashion. Each head alternately steps past its partner to the next binding site along the MT moving the cargo a distance of about 8 nm (the length of a tubulin dimer)[28]. Figure 3 is a still picture from an animation[29] showing a vesicle being pulled along an MT by kinesin.

Head coordination

In order for kinesin to walk, each head in turn must step along the MT while at least one head must be bound to the MT at any one time. The affinity of a head for the MT has, therefore, to change during the walking cycle. The affinity depends on the bound nucleotide. A head is weakly attracted to the MT when ADP-bound whereas it is tightly bound to the MT when the nucleotide binding site is empty or ATP is bound[30]. Thus the hydrolysis cycle provides the necessary alteration in affinity for the MT that enables kinesin to walk. Kinesin in solution has ADP-bound heads. When a head encounters an MT it binds to it and releases ADP. ATP then binds to the head which proceeds to hydrolyse it, phosphate is released and the head detaches from the MT ready to take the next step. The second head follows the same cycle of events[31].

Note that the hydrolysis cycle of one head has to be out of phase with that of the other head. If it were in phase then the motor would be unable to walk as both heads would be bound to the MT (preventing stepping) or they would be free at the same time resulting in the kinesin diffusing away form the MT. It is not known how the hydrolysis cycle is synchronised to the walking motion but one theory is that tension in the neck linkers modulates nucleotide binding[32].

Walk this way

As the kinesin molecule is a homodimer (comprising two identical components) the directionality of its travel along the MT needs explanation. Kinesin walks towards the plus end of an MT for two reasons. The first is that the heads bind in only one direction (mainly to the β-tubulin of the dimer). The second is that binding of ATP to the empty head causes a conformational change in the head such that the neck linker zips to the head thus restricting the free head to binding the next rather than the previous MT binding site[31].

It was initially believed that kinesin moved forward by a power stroke action using the energy released by ATP hydrolysis. On this view, the trailing head is pulled forward by the zippering of the neck linker to the MT-bound head. It is now clear that the energy of zippering is insufficient to achieve a power stroke. An analysis of the physics of the motion indicates that the mechanism is biased Brownian motion[33]. On this account, the diffusive motion of the free head is biased in the direction of movement by neck linker

zippering. The energy released by ATP hydrolysis powers conformational change in the head and not the forward motion which is diffusive.

Transport regulation

Not much is known about the regulation of axonal transport. The movement of axonal mitochondria is a case in point[26]. For the system to work properly cargo has to be moved to where it is required. For a vesicle containing neurotransmitter this entails binding to kinesin which then has to take the appropriate path *e.g.* along the axon rather than a dendrite, with transfer to the actomyosin system for local transport and storage at the synapse.

Lethal pileups on the axon highway

Several possible causes of transport failure have been discovered besides lack of tau stabilisation of microtubules as discussed under the tau hypothesis.

The APP connection

Kinesin transports vesicles containing APP, β-secretase and presenilin-1 towards the synapse[34]. It is suggested that axonal blockage would result in the accumulation of these vesicles which might encourage proteolysis of the APP to generate Aβ and so cause the neurodegenerative amyloid cascade[23]. Mice expressing mutant human APP developed axonal swellings similar to those observed in AD brains. The swellings accumulated abnormal amounts of microtubule-associated and molecular motor proteins, organelles, and vesicles were observed. Reducing the dosage of kinesin enhanced the frequency of axonal defects, increased Aβ levels and amyloid deposition. The proposed mechanism was that axonal transport failure causes a build-up of toxic material that initiates β-secretase cleavage of APP resulting in Aβ production, development of senile plaques and AD.

The opposite view was proposed following a cell culture study. Soluble (but not fibrillar) Aβ in the presence of tau caused loss of MTs and numerous axonal swellings filled with membrane-bound organelles in rat cortical neurons[35]. Perhaps both proposals are correct: Aβ and axonal transport failure cause each other. It is clear that Aβ production is intimately bound up with axonal transport failure.

Hirano blockage

In addition to plaques and tangles there is a third type of lesion characteristic of AD brains: Hirano bodies. They contain actin filaments and tau protein, the latter is implicated in their formation though this process is not well understood[36]. It is proposed that these inclusions (like tau fibrils) physically disrupt axonal transport.

Axonal pathogen transport

Viral infection has been proposed as a cause of AD[37]. Viruses or their components use axonal transport to traverse neurons. A study of how HSV gets to the mucosal membrane after manufacture in the cell body used a giant squid axon to show that HSV uses axonal transport. APP is present in HSV and it appears to be instrumental in its binding to kinesin for transport[38]. A rat study has shown that axonal transport of the HIV-1 envelope glycoprotein gp120 from the striatum or hippocampus to distal neurons results in cell death. This finding indicates that axonal transport could be the mechanism for promoting widespread neuron loss causing dementia associated with AIDS[39]. Could this also apply to AD?

Conclusions – towards a cure for Alzheimer's

Alzheimer's disease (AD) is proving a monumental challenge. Progress is being made but our knowledge is fragmentary. We have yet to discover the cause of sporadic AD, how genetic mutations lead to the familial form, why only some types of neurons are affected, what governs the progress of the disease through the brain, what determines the pathway for APP proteolysis, which comes first: abnormal phosphorylation of tau or axonal transport dysfunction, whether lesions are formed as a result of defence or pathology or both, why plaques aren't cleared by microglia...

Current treatment, though helpful to many patients, is palliative. Search for a cure is focussed primarily on preventing the characteristic lesions – neurofibrillary tangles and senile plaques – found in AD brains. Though there are compounds that show promise in defeating tau pathology in animal models, there are significant barriers to the development of suitable drugs for humans. Evidence of the toxicity of pre-amyloid aggregations of Aβ in animal models has spurred development of drugs designed to eliminate or suppress

the production of Aβ. Clinical trials of a vaccine are well advanced though adverse reactions have so far prevented deployment. It is to be hoped that safe and effective drugs will soon be on the market.

The ideal position would be to fully understand AD so as to be able to diagnose it early, before any irreversible damage is done, if not prevent it altogether. Laboratory studies indicate that axonal transport dysfunction occurs early in the disease and that there are connections between disruption of axonal transport, abnormal proteins and neurodegeneration. Axonal transport seems, therefore, central to the enterprise of conquering this abominable disease, having the potential to pull together all the components so far identified[40].

Acknowledgement

The author's doctoral research is funded by the Engineering and Physical Sciences Research Council.

References

1. Knapp, M. and Prince, M. (2007) *The Rising Cost of Dementia in the UK*. Alzheimer's Society, London.
2. Terry, R.D., Masliah, E., Salmon, D.P., Butters, N., DeTeresa, R., Hill, R., Hansen, L.A. and Katzman, R. (1991) Physical basis of cognitive alterations in Alzheimer's disease: synapse loss is the major correlate of cognitive impairment. *Ann. Neurol.*, **30**, 572–580.
3. Goedert, M. and Spillantini, M.G. (2006) A Century of Alzheimer's Disease. *Science*, **314**, 777–781.
4. Mount, C. and Downton, C. (2006) Alzheimer disease: progress or profit? *Nat. Med.*, **12**(7), 780–3.
5. Arriagada, P.V., Growdon, J.H., Hedley-Whyte, E.T. and Hyman, B.T. (1992) Neurofibrillary tangles but not senile plaques parallel duration and severity of Alzheimer's disease. *Neurology*, **42**, 631–639.
6. Selkoe, D.J. (2002) Alzheimer's disease is a synaptic failure. *Science*, **298**, 789–791.
7. Glenner, D.G. and Wong, C.W. (1984) Alzheimer's disease: initial report of the purification and characterization of a novel cerebrovascular amyloid protein. *Biochem. Biophys. Res. Commun.*, **120**(3), 885–90.
8. Geula, C., Wu, C.K., Saroff, D., Lorenzo, A., Yuan, M. and Yankner, B.A. (1998) Aging renders the brain vulnerable to amyloid beta-protein neurotoxicity. *Nat. Med.*, **4**, 827–831.
9. Knobloch, M., Konietzko, U., Krebs, D.C. and Nitsch, R.M. (2007) Intracellular Aβ and cognitive deficits precede β-amyloid deposition in transgenic arc Aβ mice. *Neurobiol. Aging*, **28**, 1297–1306.

10. Robinson, S.R. and Bishop, G.M. (2002) A-β as a bioflocculant: implications for the amyloid hypothesis of Alzheimer's disease. *Neurobiol. Aging*, **23**, 1051–1072.
11. Dobson, C.B., Wozniak, M.A. and Itzhaki, R.F. (2003) Do infectious agents play a role in dementia? *Trends Microbiol.*, **11**(7), 312–317.
12. Schenk, D. (2002) Amyloid-β immunotherapy for Alzheimer's disease: the end of the beginning. *Nat. Rev. Neurosci.*, **3**(10), 824–8.
13. White, A.R., Du, T., Laughton, K.M., Volitakis, I., Sharples, R.A., Hoke, D.E., Holsinger, R.M.D., Evin, G., Cherny, R.A., Hill, A.F., Barnham, K.J., Li, Q.-X., Bush, A.I. and Masters, C.L. (2006) Degradation of the Alzheimer's Disease Amyloid Beta Peptide by Metal-dependent Up-regulation of Metallprotease Activity. *J. Biol. Chem.*, **281**, 17670–17680
14. Deane, R. and Zlokovic, B.V. (2007) Role of the blood-brain barrier in the pathogenesis of Alzheimer's disease. *Curr. Alzheimer Res.*, **4**(2), 191–7.
15. Parkin, E.T., Watt, N.T., Hussain, I., Eckman, E.A., Eckman, C.B., Manson, J.C., Baybutt, H.N., Turner, A.J. and Hooper, N.M. (2007) Cellular prion protein regulates β-secretase cleavage of the Alzheimer's amyloid precursor protein. *Proc. Natl. Acad. Sci.*, **104**(26), 11062–11067.
16. Ballatore, C., Lee, V.M.-Y. and Trojanowski, J.Q. (2007) Tau-mediated neurodegeneration in Alzheimer's disease and related disorders. *Nat. Rev. Neurosci.*, **8**, 663–672.
17. Li, B., Chohan, M.O., Grundke-Iqbal, I. and Iqbal, K. (2007) Disruption of microtubule network by Alzheimer abnormally hyperphosphorylated tau. *Acta Neuropathol.*, **113**(5), 501–11.
18. Wang, J.-Z., Grundke-Iqbal, I. and Iqbal, K. (2007) Kinases and phosphatases and tau sites involved in Alzheimer neurofibrillary degeneration. *Eur. J. Neurosci.*, **25**(1), 59–68.
19. SantaCruz, K., Lewis, J., Spires, T., Paulson, J., Kotilinek, L., Ingelsson, M., Guimaraes, A., DeTure, M., Ramsden, M., McGowan, E., Forster, C., Yue, M., Orne, J., Janus, C., Mariash, A., Kuskowski, M., Hyman, B., Hutton, M. and Ashe, K.H. (2005) Tau Suppression in a Neurodegenerative Mouse Model Improves Memory Function. *Science*, **309**, 476–4481.
20. Takashima, A. (2006) GSK-3 is essential in the pathogenesis of Alzheimer's disease. *J. Alzheimers Dis.*, **9**(3 Suppl), 309–17.
21. Zhang, B., Maiti, A., Shively, S., Lakhani, F., McDonald-Jones, G., Bruce, J., Lee, E.B., Xie, S.X., Joyce, S., Li, C., Toleikis, P.M., Lee, V.M.-Y. and Trojanowski, J.Q. (2005) Microtubule-binding drugs offset tau sequestration by stabilizing microtubules and reversing fast axonal transport deficits in a tauopathy model. *Proc. Natl. Acad. Sci.*, **102**(1), 227–231.
22. Roy, S., Zhang, B., Lee, V.M.-Y. and Trojanowski, J.Q. (2005) Axonal transport defects: a common theme in neurodegenerative diseases. *Acta Neuropathol.*, **109**, 5–13.
23. Stokin, G.B., Lillo, C., Falzone, T.L., Richard, G., Brusch, R.G., Rockenstein, E., Mount, S.L., Raman, R., Davies, P., Masliah, E., Williams, D.S. and Goldstein, L.S.B. (2005) Axonopathy and Transport Deficits Early in the Pathogenesis of Alzheimer's Disease. *Science*, **307**, 1282–88.

24. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. and Walter, P. (2002) *Molecular biology of the cell*. Garland Science, NY.
25. Hirokawa, N. and Takemura, R. (2005) Molecular motors and mechanisms of directional transport in neurons. *Nat. Rev. Neurosci.*, **6**, 201–214.
26. Hollenbeck, P.J. and Saxton, W.M. (2005) The axonal transport of mitochondria. *J. Cell. Sci.*, **118**, 5411–5419.
27. Vale, R.D. (2003) The molecular motor toolbox for intracellular transport. *Cell.*, **112**, 467–480.
28. Asbury, C.L. (2005) Kinesin: world's tiniest biped. *Curr. Opin. Cell. Biol.*, **17**, 89–97.
29. Viel, A., Lue, R.A. and Liebler, J. (2006) animation at http://multimedia.mcb.harvard.edu/anim_innerlife.html
30. Uemura, S., Kawaguchi, K., Yajima, J., Edamatsu, M., Toyoshima, Y.Y. and Ishiwata, S. (2002) Kinesin-microtubule binding depends on both nucleotide state and loading direction. *Proc. Natl. Acad. Sci.*, **99**(9), 5977–5981.
31. Rice, S., Lin, A.W., Safer, D., Hart, C.L., Naber, N., Carragher, B.O., Cain, S.M., Pechatnikova, E., Wilson-Kubalek, E.M., Whittaker, M., Pate, E., Cooke, R., Taylor, E.W., Milligan, R.A. and Vale, R.D. (1999) A structural change in the kinesin motor protein that drives motility. *Nature*, **402**, 778–784.
32. Sablin, P.E. and Fletterick, R.J. (2004) Coordination between Motor Domains in Processive Kinesins. *J. Biol. Chem.*, **279**(16), 15707–15710
33. Mather, W.H. and Fox, R.F. (2006) Kinesin's Biased Stepping Mechanism: Amplification of Neck Linker Zippering. *Biophys. J.*, **91**, 2416–2426.
34. Kamal, A., Almenar-Queralt, A., LeBlanc, J.F., Roberts, E.A. and Goldstein, L.S. (2001) Kinesin-mediated axonal transport of a membrane compartment containing beta-secretase and presenilin-1 requires APP. *Nature*, **414**, 643–648.
35. King, M.E., Kan, H.-M., Baas, P.W., Erisir, A., Glabe, C.G. and Bloom, G.S. (2006) Tau-dependent microtubule disassembly initiated by prefibrillar β-amyloid. *J. Cell. Biol.*, **175**, 541–546.
36. Gallo, G. (2007) Tau is actin up in Alzheimer's disease. *Nat. Cell. Biol.*, **9**, 133–134
37. Itzhaki, R.F., Lin, W.-R., Shang, D., Wilcock, G.K., Faragher, B. and Jamieson, G.A. (1997) Herpes simplex virus type 1 in brain and risk of Alzheimer's disease. *Lancet*, **349**, 241–4.
38. Satpute-Krishnan, P., DeGiorgis, J.A. and Bearer, E.L. (2003) Fast anterograde transport of herpes simplex virus: role for the amyloid precursor protein of Alzheimer's disease. *Aging Cell*, **2**, 305–18.
39. Bachis, A., Aden, S.A., Nosheny, R.L., Andrews, P.M. and Mocchetti, I. (2006) Axonal Transport of Human Immunodeficiency Virus Type 1 Envelope Protein Glycoprotein 120 Is Found in Association with Neuronal Apoptosis. *J. Neurosci.*, **26**(25), 6771–6780.
40. Adalbert, R., Gilley, J. and Coleman, M.P. (2007) Abeta, tau and ApoE4 in Alzheimer's disease: the axonal connection. *Trends Mol. Med.*, **13**(4), 135–42.