2013

Protein Structure Modeling and Its Application to the Mechanics of Actomyosin Interaction

Cynthia N. Prudence
University of Rhode Island, cynthia.prudence@gmail.com

PROTEIN STRUCTURE MODELING AND ITS APPLICATIONS TO THE MECHANICS OF ACTOMYOSIN INTERACTION

BY

CYNTHIA N. PRUDENCE

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

APPLIED MATHEMATICAL SCIENCES

UNIVERSITY OF RHODE ISLAND

2013

DOCTOR OF PHILOSOPHY DISSERTATION

OF

CYNTHIA N. PRUDENCE

APPROVED:

Dissertation Committee:

Major Professor Yana K. Reshetnyak

Major Professor Oleg A. Andreev

Roberta S. King

Leonard M. Kahn

Nasser H. Zawia
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2013

ABSTRACT

Study and modeling of protein structures provides an opportunity to elucidate the molecular mechanisms of key biological processes, as well as interpret experimental data obtained by variety of physical methods. The main goal of my work was to develop computational algorithms for the analysis of protein structures, the docking of protein-protein interactions, the implementation of these algorithms and the correlation of protein structure analysis with the results of steady-state and kinetics fluorescence measurements.

First, we developed and implemented algorithms for the decomposition of multi-component protein fluorescence spectra and the correlation of these spectral parameters with protein structural properties. The implementation is available as on-line toolkit PFAST (Protein Fluorescence And Structural Toolkit) at http://pfast.phys.uri.edu/. The results of our work would allow researchers to extract important novel information about protein structure and dynamics, which is probed by steady-state fluorescence spectroscopy.

Next, we developed and implemented a mathematical model to analyze the kinetic fluorescence data, which reflects the interaction of myosin subfragment 1 with one and two monomers of F-actin. Myosin and actin are the major proteins of muscles and acto-myosin interaction is responsible for the force generation in biological systems. By fitting experimental spectral data we have shown that the sequential binding of the myosin head initially with one actin monomer and then with the second actin monomer in F-actin can play a key role in force generation by actin-myosin and their directed movement.

Finally, we performed computational rigid body docking of the atomic structures of S1 and F-actin with two fixed points corresponding to the crosslinking sites and two distinct conformations of S1 on the F-actin surface. The minimization

of the binding free energy of the generated complexes was done using an empirical method developed previously, and it allowed us to select the best actomyosin models: where the myosin neck was pointed toward the barbed (model A) or the pointed (model B) ends of F-actin. The complex of model B was twice less stable than the complex of model A, which led to the conclusion that the second actin, with which S1 interacts, is located in the same strand, closer to the barbed end of F-actin. We concluded that, the formation of the actomyosin complex proceeds in ordered sequence: i) S1 initially binds to one actin; ii) rotates toward the barbed end of F-actin and binds with the second actin. The sequential mechanism of formation of actomyosin interface starting from one end and developing towards the barbed end could play an important role in force generation and directional movement in actin-myosin system.

ACKNOWLEDGMENTS

Yana and Oleg for teaching me what it means to be a scientist, and making me believe I could be a great one.

Nordia for being more then a friend but family. You are best editor a friend could ask for, even with your funny British words. You are my confidence and always push me to be the best version of myself.

Brad for always being willing to listen and for generally being a great best friend.

Lorraine for being there through every crisis and helping navigate every problem. Your presence was essential to my success.

Remo for your constant support and for being my champion. I am very glad they forgot about you in my office back in 2006.

Dr. Kulenovic for being the first math teacher to see promise in me as a mathematician. You showed me the joy in both math and doing what you love. You also tell the best stories, I will miss hearing them.

My committee for all their guidance, help and patience throughout this process.

PREFACE

This dissertation is written in the Manuscript Format using the Thesis/ Dissertation template of University of Rhode Island. There are three manuscripts, each organized into a chapter.

The results of our studies presented here were published in three papers:

1. **Cynthia N. Prudence**, Brad Boudreau, and Yana Reshetnyak. Protein Fluorescence Database. 2nd International Conference on Bioscience and Bioinformatics (ICBB'11) – WSEAS, EURO-SIAM, EUROPMENT International Conferences. Montreux, Switzerland. December 29 – 31, 2011

2. Yana K. Reshetnyak, **Cynthia N. Prudence**, Segala, James, Markin, Vladislav S., and Andreev, Oleg A. September 7, 2012. Parking Problem and Negative Cooperativity of Cinding of Myosin Subfragment 1 to F-actin. Biochemical and Biophysical Research Communications. Vol. 425 Issue 4 p. 746-749

3. **Cynthia N. Prudence**, Yana K. Reshetnyak and Oleg A. Andreev. Computational Modeling of Actomyosin Complexes. In preparation for submission to Journal of Theoretical Biology

0.1 *Conference Oral Presentations*

Cynthia N. Prudence, Brad Boudreau, and Yana Reshetnyak. Protein Fluorescence Database. 2nd International Conference on Bioscience and

Bioinformatics (ICBB'11) – WSEAS, EURO-SIAM, EUROPMENT International Conferences. Eden Palace Hotel. Montreux, Switzerland. December 29 – 31, 2011.

Cynthia N. Prudence and Yana Reshetnyak. 2011. Implementation and Applications of the Protein Fluorescence and Structural Toolkit (PFAST). BeNeLux Bioinformatics Conference 2011 (BBC). Alvisse Parc Hotel, Luxembourg, Luxembourg. December 12 – 13, 2011.

Cynthia N. Prudence, James Segala, Vladislav Markin, Yana Reshetnyak, and Oleg Andreev. 2010. Acto-myosin Interaction. Joint Fall Meeting of the New England Sections of the American Physics Society and the American Association of Physics Teachers. Barus and Holley Hall. Providence, RI, October 29 – 30, 2010.

0.2 *Conference Posters*

Cynthia N. Prudence and Yana Reshetnyak. 2011. Implementation and Applications of the Protein Fluorescence and Structural Toolkit (PFAST). BeNeLux Bioinformatics Conference 2011 (BBC). Alvisse Parc Hotel, Luxembourg, Luxembourg. December 12 – 13, 2011.

Cynthia N. Prudence, Vladislav Markin, Yana Reshetnyak, and Oleg Andreev. 2011. Modeling of Actomyosin Interactions. 19th Annual International Conference on Intelligent Systems for Molecular Biology and 10th European Conference on Computational Biology. Austria Center Vienna, Vienna, Austria. July 15 – 19, 2011.

Cynthia N. Prudence and Yana Reshetnyak. 2011. Implementation of

the Protein Fluorescence And Structural Toolkit (PFAST). 19th Annual International Conference on Intelligent Systems for Molecular Biology and 10th European Conference on Computational Biology. Austria Center Vienna, Vienna, Austria. July 15 – 19, 2011.

Cynthia N. Prudence, Vladislav Markin, Yana Reshetnyak, and Oleg Andreev. 2011. Modeling of Actomyosin Interactions. Great Lakes Bioinformatics Conference 2011. Ohio University, Athens, Ohio. May 2 – 4, 2011.

Cynthia N. Prudence and Yana Reshetnyak. 2011. Implementation of the Protein Fluorescence And Structural Toolkit (PFAST). Great Lakes Bioinformatics Conference 2011. Ohio University, Athens, Ohio. May 2 – 4, 2011.

Cynthia N. Prudence, Vladislav Markin, Yana Reshetnyak, and Oleg Andreev. 2011. Modeling of Actomyosin Interactions. 2011 The Empowering Leadership Alliance National Meeting. The Fairmont Hotel, San Francisco, CA, April 3, 2011.

Cynthia N. Prudence, James Segala, Vladislav Markin, Yana Reshetnyak, and Oleg Andreev. 2009. Acto-myosin Interaction. New England Science Symposium. The Joseph B. Martin Conference Center at Harvard Medical School, Boston, MA, April 3, 2009.

Cynthia N. Prudence, James Segala, Vladislav Markin, Yana Reshetnyak, and Oleg Andreev. 2009. Acto-myosin Interaction. Biophysical Society 53rd Annual Meeting. Boston Convention and Exhibition Center. Boston, MA,

28 February – 4 March 2009.

Cynthia N. Prudence, D. Das, C. Shen, R. Menon, N. Nahar, N. Bansal, S. Jaegle, N. Guduru, J. Peckham, and Y. Reshetnyak. 2008. PFAST: ProteinFluorescence and Structure Toolkit. Second Annual Rhode Island Research Alliance Symposium. RI Convention Center, Providence, Rhode Island, June 3, 2008.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

MANUSCRIPT 1

Protein Fluorescence Database

Published in the Conference Proceedings for the 2nd International Conference on Bioscience and Bioinformatics (ICBB'11)

Cynthia N. Prudence[1]2, Brad Boudreau[1], Yana K. Reshetnyak[1]

[1]Computer Sciences Department

[2]Physics Department

University of Rhode Island

East Hall, 2 Lippitt Rd

Kingston, RI 02881

reshetnyak@mail.uri.edu, http://biophys.phys.uri.edu

Abstract: The fluorescence properties of tryptophan residues are sensitive to the microenvironment of tryptophan fluorophores in proteins. Therefore, fluorescence characteristics are widely used to study structural transitions in proteins. We have developed algorithms for the decomposition of multi-component protein fluorescence spectra and the correlation of these spectral parameters with protein structural properties, which are available via PFAST (Protein Fluorescence And Structural Toolkit, http://pfast.phys.uri.edu/). We then applied these algorithms for the analysis of fluorescence properties of more than 300 proteins and created the first database of protein fluorescence properties, which is available on the PFAST website (http://pfast.phys.uri.edu/database/database.php).

Key Words: Protein fluorescence, database, protein structure, decomposition algorithms, statistical analysis, tryptophan residues

1. Introduction

Fluorescence spectroscopy is a powerful tool for the investigation of protein structure, conformations and dynamics, since fluorescence properties of tryptophan residues vary widely depending on the environment of the tryptophan in a given protein. Tryptophan fluorescence is traditionally employed in investigation of protein-protein, -DNA, -RNA, -membrane interactions and folding studies. The major goal in the application of tryptophan fluorescence spectroscopy is to interpret fluorescence properties in terms of structural parameters and to predict structural changes in a protein. One of the major obstacles in the analysis of fluorescence data lies in the complex nature of protein fluorescence. The overwhelming majority of proteins contain more than one fluorophore and therefore exhibit deceptively smooth spectra containing several components. The

multicomponent nature of protein spectra makes their unequivocal interpretation difficult. Mathematically, the problem of the decomposition of fluorescence spectrum into elementary components is an inverse ill-posed problem problems of this class are common in spectral and image analyses [1-4]. To address this problem we have implemented two mathematically different algorithms SIMS (SImple fitting procedure using the root-Mean-Square criterion) and PHEQ (PHase-plot-based REsolution using Quenchers) for the analysis of protein fluorescence spectra [5]. The stability of the solution is ensured by the use of additional physical data obtained from experiments and by the application of the log-normal function to describe the individual spectral components. The algorithms allow for a stable decomposition of tryptophan fluorescence spectra into at most three spectral components. Since the main goal of the application of tryptophan fluorescence spectroscopy is to correlate spectral properties with structural parameters of proteins, we have also designed and implemented algorithm for the analysis of structural parameters of microenvironment of tryptophan residues from the atomic structures of proteins from the Protein Data Bank (PDB) [6]. The algorithm reveals a set of structural parameters, which we have shown correlates with a set of spectral parameters obtained as a result of the application of the decomposition algorithms [7]. We applied multivariate classification algorithms to solve the task of correlating the set of protein spectral and structural parameters. The algorithms developed for spectral and structural analysis were applied and verified on the multi-tryptophan proteins and their single-tryptophan mutants [8-9]. Currently, various groups use the algorithms in the analysis of spectral data.

2. Problem Formulation

There are numerous databases available on-line that combine various physical and

structural parameters of proteins. However, there was no database containing protein fluorescence properties, despite the fact that protein fluorescence is one of the traditional methods used to study protein structure.

3. Problem Solution

We have implemented algorithms for the analysis of protein fluorescence and structural data and created web-based Protein Fluorescence And Structural Toolkit (PFAST) [10]. PFAST contains three main modules (see Diagram below): 1) FCAT is a fluorescence-correlation analysis tool, which decomposes protein fluorescence spectra to reveal the spectral components of individual tryptophan residues or groups of tryptophan residues located close to each other, and assigns spectral components to one of five previously established spectral-structural classes. 2) SCAT is a structural-correlation analysis tool, used to calculate the structural parameters of the environment of tryptophan residues from the atomic structures of proteins from the PDB, and to assignment of tryptophan residues to one of five spectral-structural classes. 3) The last module is the PFAST database. The PFAST database consists of two main parts: protein fluorescence data obtained from the results of the FCAT analysis and protein structural data obtained from the results of the SCAT analysis. Users can easily populate the protein structural portion of the PFAST database, as it only requires the user to provide a PDB file name, and the programs will pull the PDB file and run calculations. Since the PDB is a validated source, all entries generated by users will be valid entries of PFAST database. However, it is important to remember that for a proper comparison of protein fluorescence properties with protein structural parameters to be made, the user needs to verify that the atomic coordinates of the protein were obtained under conditions similar to the conditions

of the fluorescence spectra measurements. We also introduced an algorithm that allows the user to run the structural programs and perform analysis using a file containing the atomic coordinates of a protein molecule provided by user. These data, however, would not be included into the database.

In contrast, the database of protein spectral parameters requires an administrator to first validate the spectral data before it can be added the database. The main purpose of the validation is to ensure that all the spectral measurements were performed correctly. Validation requires information about the instrument settings and the conditions of the measurements and the fluorescence spectrum of tryptophan amino acid in aqueous solution as a standard obtained using same settings of the spectrofluorimeter.

A module bulk-insert was specially designed to allow for the uploading of multiple spectral data, and to populate the first database of protein fluorescence properties. This database contains the fluorescence spectra of more than 300 proteins in various conformational states, the results of the spectral decomposition analysis, and the assignment of the spectral components to one of five spectral-structural classes. The database is integrated with the protein structural analysis and it is a part of PFAST. The bulk-insert module can be used to populate the database with multiple entries, as well as to run multiple calculations (analysis of spectral data) as an entire package. For example, in the course of titration experiments, measurements of pH-dependences, or folding/unfolding studies, a user might collect a number of spectra, each measured at various conditions. Each spectrum needs to be analysed independently, which is a time consuming procedure. The bulk-insert module provides the ability to run the analysis of a set of data all at once.

3.1 Overview of the bulk-insert module

The system contains five components: two PHP based scripts, one Java class file, and two Visual Basic programs (PKK or PKM). The first PHP script, bulkinsert.php, is used to upload a text file containing the appropriately formatted protein fluorescence data. Next, the Java class file, bulkinsert.class, uses one of the two Visual Basic programs to perform calculations. Finally, the SQL statements and all the results data are generated.When the execution is completed, the second PHP script, perform_insert.php, finds the file containing the SQL statements from the bulkinsert.class and proceeds to query the database with the insertions. Once the process is completed, the perform_insert.php returns to the bulkinsert.php, while letting the user know that the upload is complete.

3.2 Structure of the input file

The input data file needs to be arranged according to the following guidelines: 1) the PFAST ID code (a unique combination of numbers and letters as each entry in PFAST database has a unique ID) 2) the protein name 3) the source of protein 4) the experimental conditions 5) the experimental settings 6) the quencher of fluorescence used in the study (if any) 7) any comments about the protein 8) the names of the people who did the measurements 9) the reference to the published source (if any) 10) the comment line for the input data file that will be used by the PKM or PKK Visual Basic programs 11) the experimental data (fluorescence spectra, which needs to be decomposed).

The data in the database of fluorescence properties can be searched by the PFAST ID code, the protein name, the source of the protein, the experimental

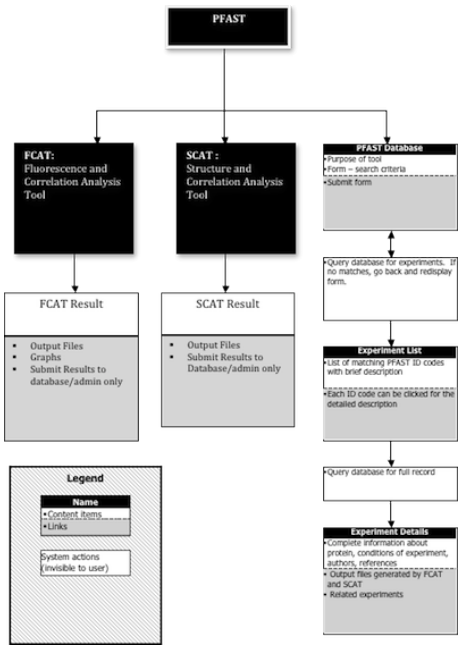

Figure 1.1. Structure of the PFAST database and the data presented in the database

conditions or the date of submission. Each entry contains information about the protein ID code, the protein name and the source, the conditions of the measurements and the settings of the instrument, the authors who performed the measurements and the references. Fluorescence analysis files contain links to the input.txt and three output files: results.txt, summary.txt and graph.txt. The first file, input.txt, is used to obtain the calculated data. The output summary file, summary.txt, is a short version of the obtained results of the decomposition. It contains information about the applied decomposition algorithms (SIMS and PHREQ) and the obtained spectral parameters, which are the position of the maximum and the contributions of the spectral components, the assignment of the components to one of five spectral-structural classes and the goodness-of-fit. The result file, results.txt, contains detailed information about the decomposition procedure including information about the spectral parameters in the wavelength and the wavenumber scales, the contributions of each component to the total

fluorescence spectrum measured at various concentrations of quenchers, and the Stern-Volmer constants, which represents the accessibility of the fluorophore to quencher. Clicking on the link, visualize graph will show the graphical representation of the obtained data. All values of the original experimental data, the data obtained as a result of the smoothing of the experimental data, the calculated spectral components, the residuals and the goodness-of-fit are combined in the graph file, which allows users to make their own plots. Finally, the page with the results contains links to similar experiments.

4. Conclusion

FCAT and SCAT are toolkits for calculating the spectral and structural properties of tryptophan residues in proteins. PFAST contains the first database of tryptophan fluorescence properties obtained by spectral decomposition analysis. This provides information on the spectral properties of individual tryptophan residues or clusters of nearby tryptophan residues, as well as on the assignment of the tryptophan fluorophores to one of the five spectral-structural classes.

References:

[1] A. Tikhonov, Ill-Posed Problems in the Natural Sciences. Moscow, 1987; Vol. 344.

[2] I. Craig, J. Brown, Inverse Problems in Astronomy. Adam Hilger Ltd.: Bristol, 1986.

[3] S. Henn, K. Witsch, A Multigrid Approach for Minimizing a Nonlinear Functional for Digital Image Matching. Computing, 64, 2000, 339-348.

[4] P.D. Wentzell, S.S. Nair, R.D. Guy, Three-Way Analysis of Fluorescence Spec-

tra of Polycyclic Aromatic Hydrocarbons with Quenching by Nitromethane. Anal. Chem. 73, 2001, 1408-1415.

[5] E.A. Burstein, S.M. Abornev, Y.K. Reshetnyak, Decomposition of Protein Tryptophan Fluorescence Spectra into Log-Normal Components. I. Decomposition Algorithms. Biophys J, 81, 2001, 1699-1709.

[6] Y.K. Reshetnyak, E.A. Burstein, Decomposition of Protein Tryptophan Fluorescence Spectra into Log-Normal Components. II. The Statistical Proof of Discreteness of Tryptophan Classes in Proteins. Biophys J, 81, 2001, 1710-1734.

[7] Y.K. Reshetnyak, Y. Koshevnik, E.A. Burstein, Decomposition of Protein Tryptophan Fluorescence Spectra into Log-Normal Components. III. Correlation between Fluorescence and Microenvironment Parameters of Individual Tryptophan Residues. Biophys J, 81, 2001 1735-1758.

[8] D. Linke, J. Frank, M.S. Pope, J. Soll, I. Ilkavets, P. Fromme, E.A. Burstein, Y.K. Reshetnyak, V.I. Emelyanenko, Folding Kinetics and Structure of OEP16. Biophys J, 86, 2004 1479-1487.

[9] M. Torrent, E. Cuys, E. Carreras, S. Navarro, O. Lpez, A. de la Maza, M.V. Nogus, Y.K. Reshetnyak, E. Boix, Topology Studies on the Membrane Interaction Mechanism of the Eosinopjil Cationic Protein. Biochemistry, 46, 2007, 720-733.

[10] C. Shen, R. Menon, D. Das, N, Bansal, N. Nahar, N. Guduru, S. Jaegle, J. Peckham, Y.K. Reshetnyak, The Protein Fluorescence And Structural Toolkit (PFAST): Database and Programs for the Analysis of Protein Fluorescence and Structural Data. Protein: Structure, Function, and Bioinformatics, 71, 2008, 1744-1754.

MANUSCRIPT 2

Parking problem and negative cooperativity of binding of myosin subfragment 1 to F-actin.

Published in Biochemical and Biophysical Research Communications

Vol. 425, pp. 476-479, 2012.

Yana K. Reshetnyak[1], Cynthia N. Prudence[1], James Segala[1], Vladislav S. Markin[2], Oleg A. Andreev[1]

[1]Physics Department, University of Rhode Island, 2 Lippitt Rd., Kingston, RI, 02881, USA

[2]Department of Neurology, University of Texas Southwestern Medical Center, Dallas, TX, 75390, USA

andreev@mail.uri.edu, http://biophys.phys.uri.edu

Abstract

Previously we provided evidence that myosin subfragment 1 (S1) can bind either one (state 1) or two actin monomers (state 2) in solution and in muscle fiber. Here we present results of the kinetics study of binding of S1 to F-actin labeled with fluorescent dye pyrene. A transition from state 1 to state 2 depends on probability that the second actin is free, which is high when molar ratio of S1/actin (R) is less than 0.5, and it decreases dramatically when R>2.0 due to the parking problem. The kinetics data obtained at different molar ratios were well fitted by two binding states model. The sequential binding of myosin head initially with one actin monomer and then with the second actin monomer in F-actin can play a key role in force generation by actin-myosin and their directed movement.

Keywords: myosin, actin, binding kinetics, muscle contraction.

1. Introduction

A muscle contraction results from the sliding of actin and myosin filaments induced by ATP-driven cyclic interaction of myosin with actin [1, 2]. Lymn and Taylor showed that the binding of ATP to myosin induces its dissociation from F-actin followed by ATP hydrolysis in the detached state and, then, re-binding of myosin to F-actin and release of hydrolysis products [3]. Huxley suggested that binding of myosin to F-actin occurs in several steps by transition from weak to strong contacts between myosin and actin (rolling of myosin on actin filament) [4]. We showed previously that myosin head, called subfragment 1 (S1), can bind either one or two actin monomers [5-8]. The ability of S1 to bind two actins was independently confirmed by other laboratories as well [9-12]. When S1 molecules are in a molar excess of actin molecules then each bound S1

interacts predominately with one actin (1:1 complex). When actin is in excess, S1 molecules bind predominately two actin monomers (1:2 complex) [5-7]. Using the fast quench crosslinking technique we demonstrated that the initial contact occurred between loop of 627-637 residues of S1 with N-terminus of actin monomer and later loop of 567-575 residues of S1 binds to N-terminus of the second actin monomer [13]. Here we present the results of kinetics study of S1 binding with F-actin, which confirm that binding occurs in two steps and transition from state 1 to state 2 is very fast and its rate is comparable with the myosin power stroke time (10 msec) in actively contracting muscle.

2. Materials and methods

2.1 Materials

Myosin subfragment 1 was prepared by digestion of myosin (Cytoskeleton, Inc.) by chymotrypsin and separation of two S1 isoforms (S1A1 and S1A2, A1 and A2 are alkali light chains 1 and 2, respectively) on DEAE column as described previously [14]. Actin (Cytoskeleton, Inc.) was labeled with N-(1-pyrenyl) iodoacetamide (Invitrogen, Inc.) as described before [7]. The degree of labeling actin with pyrene was 95%-100%. Actin-pyrene was passed through sephadex G-50 spin column equilibrated with 0.2 mM CaCl2, 2 mM Trsi-HCl pH 7.5 buffer solution and then polymerized by adding 50 mM KCl, 2 mM MgCl2 and equal molar amount of phalloidin (Invitrogen, Inc.).

2.2 Kinetic measurements

Fluorescence kinetics measurements were carried out on a SFM-300 mixing apparatus connected to a MOS-450 spectrometer (Biologic, Inc.) under temperature control. One syringe was loaded with F-actin-pyrene and the other with S1.

All solutions were degassed several minutes under a vacuum before loading into the syringes to minimize air bubbles. Experiments were done at different molar ratios of S1/actin. The binding of S1 to F-actin was monitored by measuring fluorescence signal from F-actin-pyrene at 410 nm with excitation at 368 nm. Each kinetic curve was recorded several times (10) and then averaged, excluding the first 2-3 shots. The measured signal was normalized by the fluorescence signal monitored at mixing of F-actin-pyrene with buffer solution at the same conditions. All experiments were done in 50 mM KCl, 2 mM MgCl2, 0.2 mM CaCl2, 2 mM Tris-HCl buffer pH 7.5 at 20°C.

2.3 The two-state binding model

The two-state model of actin-myosin interaction was proposed by us earlier [7]. We assumed that binding of myosin subfragment S1 (M) to F-actin (A) occurs in two steps: i) initially S1 binds one actin to form complex (AM) state 1:

$$A + M \underset{k_{-1}}{\overset{k_{+1}}{\leftrightarrow}} AM$$

ii) then it binds the second actin located in the same strand if it is free to form complex (AMA) state 2 (Figure 2.1)

$$AM + M \underset{k_{-2}}{\overset{k_{+2}}{\leftrightarrow}} AMA$$

where k_{+1}, k_{+2}, k_{-+1} and k_{-2} are the kinetic constants and the equilibrium adsorption constant for state 1 is

$$K_1 = \frac{k_{+1}}{k_{-1}} \tag{2.1}$$

the equilibrium constant of transition from state 1 to state 2 is:

$$K_2 = \frac{k_{+2}}{k_{-2}} \tag{2.2}$$

The rate of adsorption of S1 for state 1 is:

$$v_1 = k_{+1} M_F A_F k_1 M_1 \tag{2.3}$$

where A_F, M_F and M_1 are the concentration of unoccupied actin, the concentration of free S1 in the solution and the amount of bound S1 in state 1, respectively.

$$A_F = A - M_1 - 2M_2 \tag{2.4}$$

$$M_F = M - M_1 - M_2 \tag{2.5}$$

where A, M and M_2 are the total amount of actin, myosin and the amount of bound S1 in state 2, respectively.
The rate of transition from state 1 to state 2 is given by:

$$v_2 = k_{+2} M_1 P_F k_2 M_2 \tag{2.6}$$

where P_F is the probability of the existence of free site next to a given singly bound S1. Any free actin or any acto-S1 complex can be the nearest neighbor to

a single-actin bound S1. The number of ways for free actin to be in this place is equal to a number of free actins, A_F. The number of ways of all occupations (by free actin or by acto-S1) of this place is equal to the sum of free actins and acto-S1 complexes, i.e., $A_F + M_1 + M_1$. Thus the probability P_F is:

$$P_F = \frac{A_F}{A_F + M_1 + M_2} \frac{A - M_1 - 2M_2}{A - M_2} \tag{2.7}$$

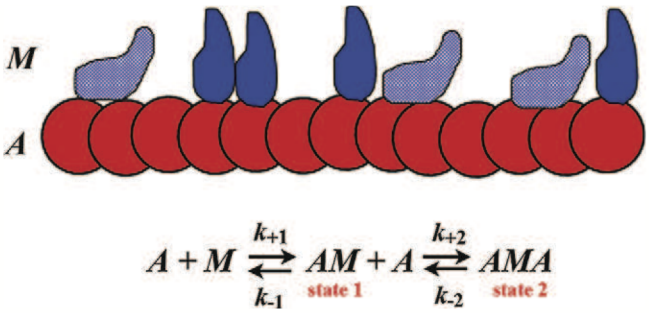

Figure 2.1. Schematic diagram of the two-state binding of myosin subfragment 1 (M, shown in blue) to F-actin (A, shown in red) is presented. S1 can bind to one actin monomer in the vertical position to form AM complex, state 1 (dark blue) or to two actin monomers in the horizontal position to form AMA complex, state 2 (light blue).

The change of the number of myosin molecules in different states with time is determined by

$$\frac{dM_1}{dt} = v_1 - v_2 \tag{2.8}$$

$$\frac{dM_2}{dt} = v_2 \tag{2.9}$$

The relative concentrations of adsorbed S1 on actin filaments are

$$\frac{m_1}{dt} = \frac{M_1}{A} \tag{2.10}$$

$$\frac{m_2}{dt} = \frac{M_2}{A} \tag{2.11}$$

and normalized concentration of free S1 in solution is given by:

$$\frac{m_f}{dt} = K_1 M_F \tag{2.12}$$

Then, the set of adsorption equations that determine the changes of the number of myosin molecules in state 1 and 2 are

$$\frac{dm_1}{dt} = k_{-1} m_F (1 - m_1 - 2m_2) - k_{-1} m_1 - \frac{k_{+2} m_1 (1 - m_1 - 2m_2)}{1 - m_2} + k_{-2} m_2 \tag{2.13}$$

$$\frac{dm_2}{dt} = \frac{k_{+2} m_1 (1 - m_1 - 2m_2)}{1 - m_2} + k_{-2} m_2 \tag{2.14}$$

In experiment we record changes of fluorescence (F) of pyrene conjugated to actin when S1 binds to the actin. The fluorescence signal measured in experiments is a sum of fluorescence of free actin (f_0), actin occupied by S1 in state 1 (f_1) and in state 2 (f_2):

$$F = f_0 A_F + f_1 A_1 + f_2 A_2 \tag{2.15}$$

$$A_F = A - A_1 - A_2 = A(1 - m_1 - 2m_2)$$

$$\frac{F}{A} = f_0 - (f_0 - f_1) \tag{2.16}$$

$F_0 = A f_0$ is the initial fluorescence signal at $t = 0$. During fitting we allow small (5%) variations for deviations in actin and S1 concentrations upon repetitive loading of actin and S1 in syringes. We applied nonlinear least squares fitting procedures using Levenberg-Marquardt algorithm to find the best fit of experimental

kinetics curve measured for different S1/actin ratios. The variable parameters were equilibrium constants (K_1 and K_2) and backward kinetic constants (k_{-1} and k_{-2}) (the forward kinetic constants then could be calculated using equations 1 and 2), and the normalized fluorescence of actin alone, bound to S1 in state 1 and state 2 (f_0, f_1 and f_2). We allowed just slight variation of normalized fluorescence of actin in three different states, since from the additional experiments we had knowledge about the approximate values of these parameters. The calculations were carried out in MatLab7.0.

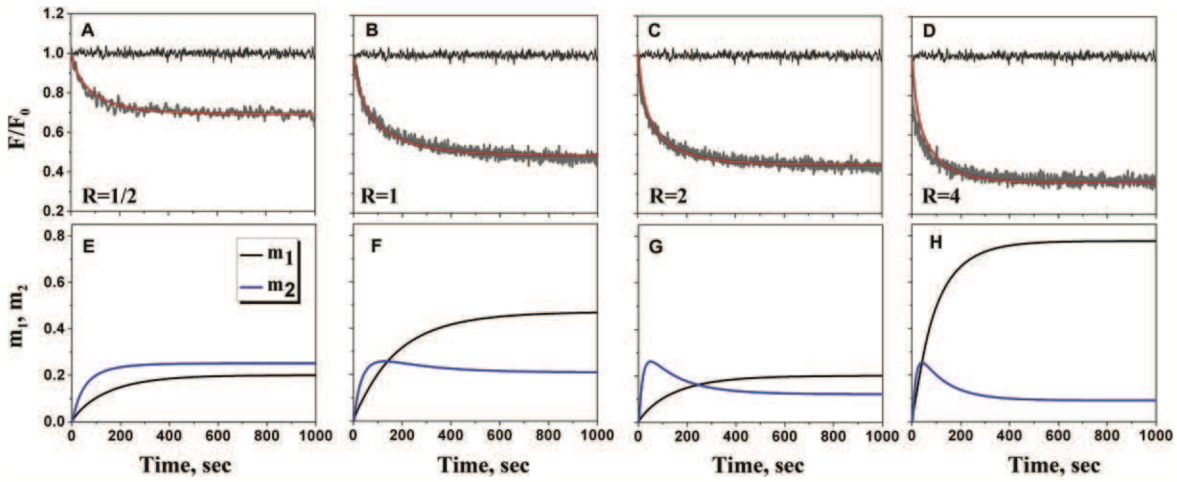

Figure 2.2. Kinetics of binding of S1 to F-actin at different molar ratios. The time course of the quenching of fluorescence of F-actin-pyrene by binding of S1 is shown (a-d). The signal of F-actin-pyrene in absence of S1 is shown in dark grey)on a-d. The data were fitted by using of the two-state binding model, theoretical curves are presented in red. Time dependences of fractional numbers of bound S1 in state 1 (m_1) and state 2 (m_2) predicted by the model for the experimental data obtained at different molar ratios are presented on e-h. The equilibrium and kinetic constants obtained in the result of the fitting are given in Table 2.1

3. Results and Discussion

We studied binding of S1 with F-actin-pyrene at different molar ratios ($R = 0.5$,

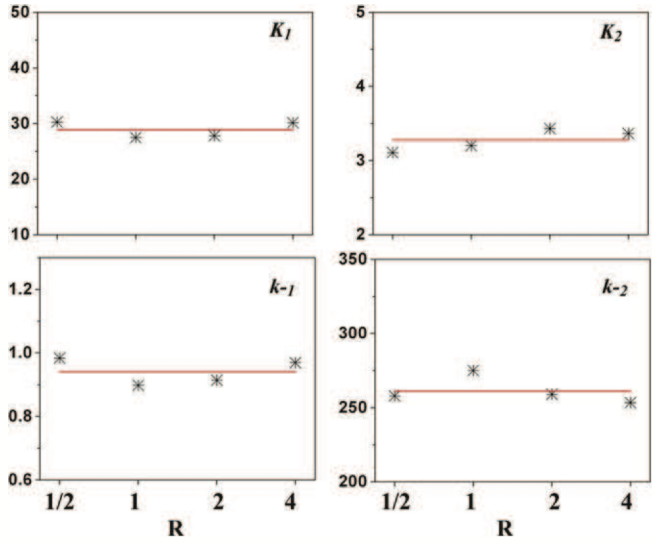

Figure 2.3. The plots show values of the fitting parameters, such as equilibrium (K_1 and K_2) and kinetic (k_{-1} and k_{-2}) constants obtained in the result of fitting of each kinetic curve measured at various ratios of S1 to actin. The red line represents mean values, which are given in Table 2.1

1, 2 and 4, where R =[S1]/[Actin]). The interaction of S1 with F-actin-pyr induces quenching of pyrene fluorescence [14, 15]. The averaged kinetics plots for S1 binding to F-actin-pyr at various molar rations are shown on Figure 2.2. All data were normalized for fluorescence signal measured when F-actin-pyr was mixed with buffer solution at the same conditions ($R = 0$). The data were fitted by the two-state model (theoretical curves shown in red on Figure 2.2 a-d). Due to the experimental error in samples concentrations after dilutions each parameter might slightly vary for each kinetic curve. Therefore, we performed fit of individual kinetic curves separately. The equilibrium and rate constants for binding of S1 and actin at different molar ratios were established (Figure 3). There is just very slight difference between the same parameters obtained for various ratios of S1 and actin, the averaged values and standard deviations for each parameter are given in Table 2.1. Time dependences of fractional numbers of bound S1 in state 1 (m_1) and state 2 (m_2) predicted by the model for the experimental data obtained at different

K_1X10^6, M^{-1}	28.9 ± 1.5
$k_{+1}X10^6, M^{-1}s^{-1}$	27.3 ± 2.6
k_{-1}, s	0.9 ± 0.1
K_2	3.3 ± 0.2
k_{+2}, s^{-1}	856.3 ± 39.5
k_{-2}, s^{-1}	261.4 ± 9.5
$Fl_0, a.u$	0.99 ± 0.03
$Fl_1, a.u$	0.70 ± 0.03
$Fl_2, a.u$	0.34 ± 0.01

Table 2.1. The mean values and standard deviations of the equilibrium (K_1 and K_2) and kinetic (k_{+1}, k_{+2}, k_{-1} and k_{-2}) constants obtained in the result of the fitting of kinetics curves obtained at various S1/actin ratios. Fl_1, Fl_2 and Fl_3, which are values of fluorescence signal for actin alone (it equals to 1), respectively, were used as variable parameters with fixed limits.

molar ratios are presented on Figure 2.2 e-h.

At low molar ratio, $R = 0.5$, the binding fast and almost mono exponential. The rate is mostly determined by the diffusion and binding of S1 to actin in state 1 followed by rapid transition to state 2. At high molar ratio, R= 4.0, the binding is also fast and almost monophasic: S1 molecules rapidly occupied all actin sites predominately in state 1 since transition to state 2 is restricted by high density of S1 on actin filament. At intermediate ratios, $0.5 < R < 2.0$, we observed a bi-phasic kinetics: initial rapid change in signal followed by slow change.

Our model provides the explanation of this slow phase of kinetics as a replacement of some S1 in state 2 by S1 in state 1. The S1s that arrived first to the actin would quickly occupy 2 actin monomers per S1, however, later they would be competitively transformed to state 1 by other S1 to reach a thermodynamic equilibrium. The slower rate of phase II is determined by growing steric restriction for binding of S1 and re-distribution of bound S1 between state 1 and state 2. The simulation curve for m_2 concentration showed that it initially increases and

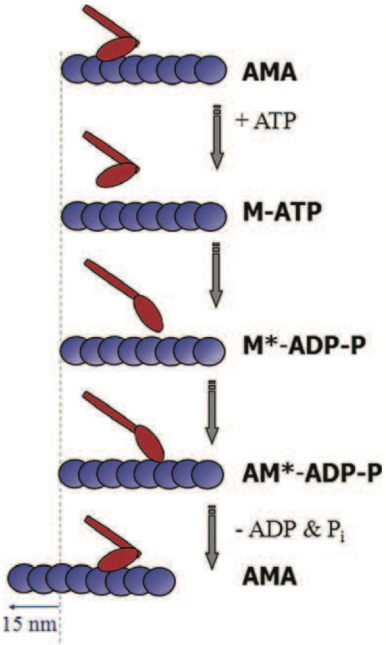

Figure 4. *Schematic presentation of the mechanism of force general and muscle contraction. Myosin head is bound to two actins, binding of ATP dissociates myosin from actin filament, following ATP hydrolysis is accompanied with unbending and extension of myosin (it is important to note that this step is occurred in absence of any mechanical load and energy loss), myosin binds to one actin (state 1) and then to two actins (state 2). Total sliding distance per one cycle is about 15 nm.*

then decreases, which means that at the beginning the probability to bind 2 actins was high but later it became less and an equilibrium between state 1 and state 2 shifted towards state 1 at higher R values (> 1.0). In kinetics studies carried out and published previously, most experiments were done at high molar ratio of S1 to actin and fitted by simple Michaelian equation [15, 16]. At these conditions a contribution of state 2 complexes was not significant and association constant between S1 and F-actin is lower than that at low ratios ($R < 0.5$).

There were several reports about observed negative cooperativity of S1 binding to F-actin, however the origin of anti-cooperativity was not identified. Blanchoin et al. [10] obtained experimental data similar to ours but they fitted their data using an artificial assumption that association constant decreased exponentially with increase of occupancy of F-actin. Despite that this assumption gave relatively good fit to experimental data it did not provide a molecular mechanism of anti-cooperativity. Our model based on experimental evidence that S1 can form two different complexes with stoichiometry 1:1 or 1:2 (S1:actin ratio). Kinetics of binding showed a strong dependence on the molar ratio of S1/actin, which can be explained by our model. Ability of myosin head to bind in sequence, initially, one actin and, then, two actins can have a fundamental significance since the transition from state 1 to state 2 might be associated with force generation and directed movement. Schematic presentation of a possible mechanism of force generation and muscle contraction is shown on Figure 4. In our previous work we showed that S1 indeed first binds one actin via loop 627-637 and then binds the second actin via loop 567-574 [13]. Such transition might provide about 10 nm (the size of two actins) sliding distance in muscle fiber. ATP binding and hydrolysis induce dissociation of myosin head from actin, followed by its unbending [17] and re-orientation for the binding to the next site further from the previous one, which can provide an additional 5 nm to total distance per myosin step.

Acknowledgements. The work was supported by URI Foundation.

4. References

[1] A.F. Huxley, R. Niedergerke, Structural changes in muscle during contraction;

interference microscopy of living muscle fibres, Nature, 173 (1954) 971-973.

[2] H. Huxley, J. Hanson, Changes in the cross-striations of muscle during contraction and stretch and their structural interpretation, Nature, 173 (1954) 973-976.

[3] R.W. Lymn, E.W. Taylor, Mechanism of adenosine triphosphate hydrolysis by actomyosin, Biochemistry, 10 (1971) 4617-4624.

[4] A.F. Huxley, R.M. Simmons, Proposed mechanism of force generation in striated muscle, Nature, 233 (1971) 533-538.

[5] O.A. Andreev, J. Borejdo, The myosin head can bind two actin monomers, Biochem Biophys Res Commun, 177 (1991) 350-356.

[6] O.A. Andreev, J. Borejdo, Two different acto-S1 complexes, J Muscle Res Cell Motil, 13 (1992) 523-533.

[7] O.A. Andreev, A.L. Andreeva, V.S. Markin, J. Borejdo, Two different rigor complexes of myosin subfragment 1 and actin, Biochemistry, 32 (1993) 12046-12053.

[8] O.A. Andreev, J. Borejdo, Binding of myosin cross-bridges to thin filaments of rabbit skeletal muscle, Biochem Biophys Res Commun, 258 (1999) 628-631.

[9] C. Valentin-Ranc, C. Combeau, M.F. Carlier, D. Pantaloni, Myosin subfragment-1 interacts with two G-actin molecules in the absence of ATP, J Biol Chem, 266 (1991) 17872-17879.

[10] L. Blanchoin, D. Didry, M.F. Carlier, D. Pantaloni, Kinetics of association of myosin subfragment-1 to unlabeled and pyrenyl-labeled actin, J Biol Chem, 271 (1996) 12380-12386.

[11] N. Bonafe, P. Chaussepied, A single myosin head can be cross-linked to the N termini of two adjacent actin monomers, Biophys J, 68 (1995) 35S-43S.

[12] J. Van Dijk, F. Celine, T. Barman, P. Chaussepied, Interaction of myosin

with F-actin: time-dependent changes at the interface are not slow, Biophys J, 78 (2000) 3093-3102.

[13] O.A. Andreev, Y.K. Reshetnyak, Mechanism of formation of actomyosin interface, J Mol Biol, 365 (2007) 551-554.

[14] O.A. Andreev, J. Borejdo, Binding of heavy-chain and essential light-chain 1 of S1 to actin depends on the degree of saturation of F-actin filaments with S1, Biochemistry, 34 (1995) 14829-14833.

[15] A.H. Criddle, M.A. Geeves, T. Jeffries, The use of actin labelled with N-(1-pyrenyl)iodoacetamide to study the interaction of actin with myosin subfragments and troponin/tropomyosin, Biochem J, 232 (1985) 343-349.

[16] E.W. Taylor, Kinetic studies on the association and dissociation of myosin subfragment 1 and actin, J Biol Chem, 266 (1991) 294-302.

[17] Y.K. Reshetnyak, O.A. Andreev, The interdomain motions in myosin subfragment 1, Biophys Chem, 94 (2001) 41-46.

MANUSCRIPT 3

Computational Modeling of Actomyosin Complexes

In preparation for submission to Journal of Theoretical Biology

Cynthia N. Prudence[1], Yana K. Reshetnyak[1], Oleg A. Andreev[1]

[1]Physics Department

University of Rhode Island

2 Lippitt Rd

Kingston, RI, 02881, USA

reshetnyak@mail.uri.edu or andreev@mail.uri.edu, http://biophys.phys.uri.edu

ABSTRACT

Force generation in muscle results from binding of myosin to F-actin. ATP binding to myosin provides energy to dissociate actomyosin complex, while the hydrolysis of ATP is needed for re-binding of myosin to F-actin. At the end of each cycle myosin and actin form a tight complex with a substantial interface area. We showed previously that myosin head (subfragment 1, S1) directly interacts with at least two monomers in actin filament. The stopped flow cross-linking experiments revealed that the initial contact (in presence or absence of nucleotides) occurs between loop 635-647 of S1 and 1-12 N-terminal residues of one actin and, then, the second contact forms between loop 567-574 of S1 and N-terminus of the second actin. Here we performed computational rigid body docking of atomic structures of S1 and F-actin with two fixed points corresponding to the crosslinking sites and two distinct conformations of S1 on F-actin surface. Stepwise translation and rotational movements were applied to S1, while F-actin was kept fixed. The minimization of the binding free energy of generated complexes calculated using the empirical method developed previously allowed us to select the best actomyosin models: where myosin neck was pointed toward the barbed (model A) or pointed (model B) ends of F-actin. The complex of the model B was twice less stable than complex of the model A, which led to the conclusion that the second actin, with which S1 interacts, is located in the same strand closer to the barbed end of F-actin. The formation of actomyosin complex proceeds in ordered sequence: i) S1 initially binds to one actin; ii) rotates toward the barbed end of F-actin and binds with the second actin. The sequential mechanism of formation of actomyosin interface starting from one end and developing towards the barbed end could play an important role in force generation and directional movement in actin-myosin system.

Keywords: myosin, actin, docking, protein-protein interaction

INTRODUCTION

A biological motility is one of the main functions of living systems. Using the chemical energy of the hydrolysis of ATP or GTP molecules, individual cells can change shape, divide and move. A mechanism of transduction of chemical energy into mechanical work is a one of the fundamental problems of modern biology. A muscle represents a highly effective biological machine, which can contract and develop a significant force in response to a nerve signal. A muscle contraction results from the sliding of actin and myosin filaments induced by the ATP-driven cyclic interaction of myosin with actin (Huxley and Niedergerke, 1954; Huxley and Hanson, 1954). Lymn and Taylor discovered that the binding of ATP to myosin induced its dissociation from F-actin followed by ATP hydrolysis in detached state and then the rebinding of myosin to F-actin and release of the hydrolysis products (Lymn and Taylor, 1971). The molecular mechanism of force generation by actin and myosin is still not completely elucidated. It has been suggested that the binding of myosin to F-actin occurred in several steps by transition from weak to strong contacts between myosin and actin (rolling of myosin on actin filament) (Huxley and Simmons, 1971). Later it was assumed that ATP hydrolysis might induce a rotation of the light chains containing the domain (lever arm) relative to the catalytic domain of myosin, and force would be produced by the reverse movement of lever arm, when myosin binds to actin and releases the hydrolysis products (Cooke, 1986; Rayment et al., 1993). The discovery of myosin VI, which moves in the opposite direction than the other myosins, led to the suggestion that the lever arm in this myosin rotates in opposite direction (Wells et al., 1999). This suggestion was challenged by demonstration that the directionality of movement

was determined by catalytic domain in genetically engineered chimera myosins with lever arm and a catalytic domain from different myosin types (Homma et al., 2001). As the alternative model to the lever arm hypothesis it was proposed that myosin might slide along actin surface (biased diffusion model) (Yanagida et al., 2000). A physical principle of biased diffusion of myosin on actin is not clear and moreover it is not supported by the observation that the genetically engineered myosin with the lever arm artificially extended and twisted on 180° moves in the opposite direction (Tsiavaliaris et al., 2004). However, the later results could be explained by the rotation of the lever arm or the rolling of myosin on actin as suggested by the Huxley-Simmons model. The important assumption of the Huxley-Simmons model is that myosin can make multiple contacts with actin and force is generated as a result of changes in actomyosin interface. In our previous work, we investigated the dynamics of actomyosin interface during complex formation in the presence and the absence of ATP by stopped flow cross-linking (Andreev and Reshetnyak, 2007). We found that the first contact occurs predominantly between myosin loop 635-647 and the N-terminus of one actin, then the second contact forms between loop 567-574 of S1 and N-terminus of the second actin. The distance between these two loops in S1 corresponds to the distance between the N-termini of two actins in the same strand (about 5 nm) but it is smaller than the distance between two actins from different strands (9-10 nm), which indicates that S1 binds the second actin on the same strand. The cross-linking experiments could not determine if the second actin located on the left or the right side from the first actin. Here we took the advantage that two contacts between S1 and F-actin were established experimentally (Andreev et al., 1993; Sutoh, 1983) and performed computational rigid body docking of S1 with F-actin with two fixed points for two opposite orientations of S1 on F-actin surface.

METHODS

In computational simulation we used atomic structures of chicken skeletal muscle S1 from the Protein Data Bank (PDB), 2MYS (Rayment et al., 1993) and reconstructed F-actin, 2ZWH (the authors provided to us atomic coordinates of a structure with 5 monomers) (Oda et al., 2009). In S1 structure the methylated lysine residues were replaced by lysine residues and the missing loops (a) 1-3, (b) 205 -215, (c) 572-574, (d) 627-646 and (e) 732-737 were reconstructed using the Modeler package of the Accelrys Discovery Studio using CHARMm forcefield (Brooks et al., 1983; Brooks et al., 2009). Ten models were generated for actin-non-binding loops (a), (b) and (e). The model with the lowest statistical potential (DOPE - Discrete Optimized Protein Energy score) for each loop was selected. Thirty models were generated for actin-binding loops (c) and (d). We choose only those models, which satisfy the criteria of having distance between two binding sites on S1 (site #1: Nζ atoms of Lys 636, or 637, or 640, 641, or 644 and site #2: Nζ atoms Lys 567, or 569, or 572, or 574) similar to the distance between two binding sites on the actin monomers, which is about 48-50 $\mathring{A}$x (distance between Oε1 atoms of Glu 4 on the first monomer of actin and Glu 2 on the second monomer of actin at the same strand). This yielded 5 options for each acin-binding loop, each of which was used to model potential formations of the actomyosin complex.

First, the coordinates of atomic structures of S1 were recalculated to have the respective binding sites at S1 and F-actin close to each (i.e. binding residues within 10-15 $\mathring{A}$) as a starting position for the fixed point docking. During the docking simulation using an algorithm developed by us, the structure of actin was kept fixed, while stepwise translational (0-10 $\mathring{A}$) and rotational (0-360°)

movements were applied to the S1 structure with reconstructed loops. All calculations were restricted by the requirement to keep the contacts revealed from the cross-linking experiments within 4 $\mathring{A}$, i.e., at least one of the Nζ atoms of Lys 636, 637, 640, 641, 644 on S1 was less than 4 $\mathring{A}$ from at least one of the Oδ1 and Oδ2 atoms of Asp 1, 3 or Oε1 and Oε2 atoms of Glu 2, 4 on the first actin monomer; and at least one of the Nζ atoms of Lys 567, 569, 572, 574 on S1 was less than 4 $\mathring{A}$ far from at least one of the Oδ1 and Oδ2 atoms of Asp 1, 3 or Oε1 and Oε2 atoms of Glu 2, 4 on the second actin monomer of the same strand.

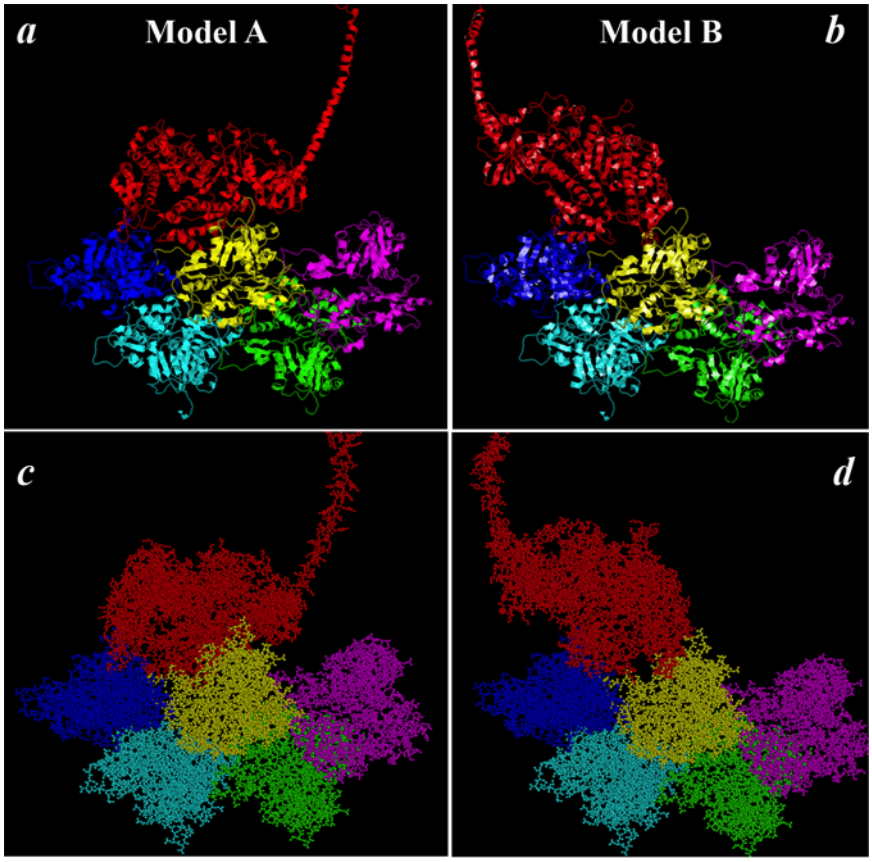

Figure 3.1. Atomic structures of acto-myosin complexes: model A (a, c) is the best selected structure, where myosin neck is pointed towards the barbed end of F-actin (actin monomer on the barbed end of F-actin shown in magenta); model B (b, d) is the best selected structure, where myosin neck is pointed towards the pointed end of F-actin (actin monomer on the pointed end of F-actin shown in blue).

The main criterion of the docking algorithm was a search for the maximum contact area between S1 and actin while keeping the atoms overlaps (clashes) at minimal value. For each model, we kept a count of all atoms (clashes) located within a distance (<1 and 1.8 $\mathring{A}$) smaller than van-der-Waals atoms radius. The binding free energy (ΔG) for actomyosin complexes was calculated using the equation (1) of the empirical method, which was introduced based on the analysis of more than 20 well known protein-protein complexes (Ma et al., 2002):

$$\Delta G = -0.87N_b - -0.35N_{pair} - 0.03\Delta ASA_{apol} + 0.92 \tag{3.1}$$

where, N_b is the side-chain accessible number; N_{pair} is the number of hydrophilic pairs; ΔASA_{apol} is the buried apolar solvent-accessible surface areas of complex interface. The solvent-accessible surface areas in S1, actin and acto-S1 complexes were calculated using program Surface Racer 1.0 (Tsodikov et al., 2002) http://pharmacy.mc.uky.edu/faculty/tsodikov/software.php) by setting of probe radius to be 1.4 $\mathring{A}$.

The buried apolar solvent-accessible surface areas were calculated according to the equation (2):

$$\Delta ASA_{apol} = \sum_{k=1}^{L}[(A^C - AM^C) + (M^C - AM^C)] \tag{3.2}$$

where (A^C, M^C and AM^C is the surface areas of carbon C-atoms of the non-polar residues in actin, myosin and acto-myosin structures, respectively. k = 1, L, where L is the number of non-polar residues, which are the following: Ala, Val, Leu, Phe, Ile, Pro, Trp, Met.

The side-chain accessible number (N_b) was calculated from the contacted residues

among the interface residues. The residue is defined as an interface residue, if the relative change of the solvent-accessible surface area ΔASA $> 20\%$:

$$\Delta ASA = 100 - \frac{AM_N}{MON_N}100 \tag{3.3}$$

MON_N is the surface area of the residue N in actin or myosin (called monomer) and AM_N is the surface area of the same residue N in acto-myosin complex. Next, the change in the side chain accessible surface area of residue in interface, δA_t, was calculated:

$$\delta A_t = 100 - MON_N^S - AM_N^S \tag{3.4}$$

MON_N^S and AM_N^S are the surface areas of the side chain of the interface residue N in actin or myosin (called monomer) and acto-myosin, respectively. The interface residue is taken as side chain accessible residue, if the effective accessibility of its side chain, $\Delta RA \geq 1$, where ΔRA is calculated by:

$$\Delta RA = \frac{DeltaA_t}{DeltaA_t^* * 60\%} * 100\% \tag{3.5}$$

where $DeltaA_t$ is the change of accessible surface area of side-chains calculated by equation (4) and $DeltaA_t^*$ is the standard side-chain surface area. The approximate value for 60% of the standard side-chain surface area in equation 5 was set to 80 $\mathring{A}^2$ in this work, as it was suggested in Ma et al (Ma et al., 2002). Thus, if $\delta RA \geq 1$, the side chain accessible number N_b is calculated.

Two atoms were treated as a hydrophilic pair, if the distance between the hydrophilic atoms (such as oxygen and nitrogen) was less than 2.8 $\mathring{A}$. The number of hydrophilic pairs will give the value of N_{pair}.

Parameters	**Model A**	**Model B**
ΔG**, kcal/mol**	-74.3	-36.2
N_b	19	9
N_{pair}	53	22
$\Delta ASA_{apol}, \AA^2$	1339	719
Number of clashes ¡ 1 $\AA$	24	27
Number of clashes ¡ 1.8 $\AA$	1335	161

Table 3.1. The main parameters of docking for the selected actomyosin complexes (models A and B) for two distinct orientations of S1.

A Java program was written to calculate each of these parameters by reading the PDB files and the results of the surface area calculations. There were in total 5 sets of models for each loop set. Each set was docked in two orientations: myosin tail was pointing to the barbered or pointed end of actin filament. The final models were selected based on the results of the energy minimization (minimal value of ΔG) with minimum number of clashes.

RESULTS

We performed a computational rigid body docking of the atomic structures of the skeletal S1 of myosin and the 5-monomers of F-actin with two fixed points. The previous cross-linking studies provided us with knowledge about interactions between the N-terminal negatively charged Asp/Glu residues on different actin monomers of F-actin with positively charged residues of loops 567-574 and 635-647 of S1 (Andreev and Reshetnyak, 2007; Andreev et al., 1993; Sutoh, 1983). First, the missing loops in the myosin S1 structure (residues 1-3, 205 -215, 572-574, 627-646 and 732-737) were reconstructed and optimized using the Modeler package of the Accelrys Discovery Studio. Several models were generated for each loop, the selected models had the lowest DOPE (Discrete Optimized Protein Energy) score, which is an atomic distance-dependent statistical potential calculated from a sample of native protein

	Model A	
Myosin	Actin, Monomer 1	Actin, Monomer 2
textbfALA 183	textbfALA 7	LEU 8
textbfVAL 408	textbfALA 22	VAL 17
textbfALA 412	PRO 102	**ALA 19**
ILE 535	LEU 104	PHE 21
MET 541	MET 132	ALA 29
PHE 542	textbfALA 144	MET 44
textbfPRO 543	TRP 340	**VAL 45**
textbfALA 545	ILE 345	**MET 47**
LEU 559	textbfALA 347	VAL 54
PRO 568	LEU 349	**TRP 86**
textbfPRO 570	PHE 352	PHE 90
textbfALA 571	TRP 356	LEU 94
textbfALA 577		VAL 96
VAL 582		**ALA 97**
textbfALA 624		PRO 98
textbfALA 630		LEU 105
textbfPHE 646		MET 123
		PHE 127
		VAL 129

Table 3.2. Interface residues for acto-myosin complexes of the model A. The contacted residues in the interface are shown in bold. These residues are counted to calculate the side-chain accessible number, N_b.

structures (Shen and Sali, 2006). For two loops in actin-binding sites (binding site #2: 572-574 and binding site #1: 627-646) we selected several models with the lowest DOPE score and the proper distances between these sites corresponding to the distance between the N-terminal Glu residues of two actin monomers, which was calculated to be 48-49 $\AA$ using 2ZWH atomic structure of F-actin. The empirical equation was used for the calculation of free energy (equation 1) at each step of rotation/translational movement of S1 around the fixed F-actin structure. In contrast to other docking procedures (see references within (Vajda and Kozakov, 2009)) (Popov et al., 2013), two points were fixed (just minimal translation freedom within a couple of angstroms was allowed). We performed docking for 2 distinct orientations of S1 molecule relative to the asymmetric F-actin structure, when myosin neck is pointed to the i) barbed or ii) pointed ends of F-actin. A number of acto-myosin structures were generated for different loops orientations in S1 and different positioning of S1 relative to F-actin. The criteria for the selection of the best structures were: i) the minimum of the free energy and ii) the minimal number of clashes, when atoms were found at distances smaller than their corresponding van-der-Waals radii. The reduction of free energy was associated with increasing number of clashes. Since the mobility of atoms in aqueous solution is significant, we assume that some clashes could easily be avoided in solution.

Both have similar number of clashes (model B has slightly more), however the binding free energy (ΔG) is twice less (-36.2 kcal/mol) for model B compared to model A (-74.3 kcal/mol). The protein-protein interface was twice less in complex of model B compare to model A. As a result, the number of interacting residues including polar pairs was twice less for model B compare to model A (see Tables 3.1-3.5 and Figure 3.1). Thus, our calculations show that the minimal free energy corresponds to the complex, where myosin neck is pointed towards the barbed end

	Model B	
Myosin	Actin, Monomer 1	Actin, Monomer 2
PRO 377	**ALA 26**	LEU 8
ALA 395	ILE 330	VAL 17
LEU 398	**PRO 333**	**ALA 19**
PRO 404	ILE 345	**PHE 21**
VAL 413	LEU 349	**PRO 27**
LEU 559		**ALA 29**
		VAL 30
		TRP 86
		PHE 90
		LEU 94
		VAL 96
		ALA 97
		PRO 98
		LEU 105
		ILE 122
		MET 123
		PHE 127
		VAL 129

Table 3.3. Interface residues for acto-myosin complexes of the model B. The contacted residues in interface are shown in bold. These residues are counted to calculate side-chain accessible number, N_b.

(model A).

DISCUSSION

In this work we combined cross-linking data with computational docking of atomic structures of skeletal myosin subfragment 1 and F-actin to obtain a structure of actomyosin complex. The cross-linking experiments allowed identifying the residues that are in close contact in actomyosin complex. The zero-length cross-linking method revealed that myosin subfragment 1 has two sites (loops

Model A	
GLY 407 - GLY 23	
GLY 409 - SER 344	
ASN 410 - SER 344	
GLU 411 - ASP 24	
ALA 412 - ASP 25	
PRO 543 - VAL 45	Second Actin
LYS 544 - ARG 147	
THR 548 - LYS 50	Second Actin
SER 549 - GLN 49	Second Actin
SER 549 - LYS 50	Second Actin
PHE 550 - GLN 49	Second Actin
ASP 556 - SER 52	Second Actin
ASP 556 - LYS 84	Second Actin
GLU 557 - ASP 51	Second Actin
GLU 557 - SER 52	Second Actin
LEU 559 - LYS 84	Second Actin
GLY 560 - GLU 83	Second Actin
SER 562 - GLU 83	Second Actin
SER 562 - GLU 83	Second Actin
ASN 563 - THR 126	Second Actin
LYS 567 - PHE 90	Second Actin
LYS 567 - TYR 91	Second Actin
LYS 567 - THR 89	Second Actin
PRO 568 - PHE 90	Second Actin
PRO 568 - TYR 91	Second Actin
LYS 569 - VAL 96	Second Actin
PRO 570 - LEU 94	Second Actin
ALA 571 - LEU 94	Second Actin
LYS 572 - GLU 2	Second Actin
ALA 577 - ASN 92	Second Actin
GLY 628 - GLU 4	
GLY 628 - THR 5	
GLY 629 - GLU 4	
GLY 629 - THR 5	
ALA 630 - GLU 4	
ALA 630 - THR 5	
ALA 630 - THR 6	
GLU 631 - GLU 4	
GLU 631 - THR 6	
GLY 632 - GLU 4	
LYS 637 - ASP 3	
SER 645 - THR 351	

Table 3.4. Polar pairs of myosin (first residue) and actin (second residue) for Model A, for which distances between their polar atoms are < 2.8 $\AA$

Model B	
LYS 369 - GLU 99	Second Actin
GLN 370 - PHE 127	Second Actin
GLN 374 - TYR 91	Second Actin
GLN 374 - TYR 91	Second Actin
GLN 396 - GLU 2	Second Actin
LEU 397 - GLU 2	Second Actin
LYS 399 - ASP 1	Second Actin
TYR 403 - LEU 94	Second Actin
TYR 403 - ARG 95	Second Actin
TYR 403 - ALA 97	Second Actin
TYR 403 - PRO 98	Second Actin
PRO 414 - LEU 94	Second Actin
LYS 415 - LEU 94	Second Actin
LYS 415 - ARG 95	Second Actin
GLY 416 - THR 89	Second Actin
GLY 416 - PHE 90	Second Actin
GLY 416 - ASN 92	Second Actin
GLU 417 - TYR 91	Second Actin
ASN 552 - GLY 146	
ASN 552 -ARG 147	
PRO 570 - GLY 23	

Table 3.5. Polar pairs of myosin (first residue) and actin (second residue) for Model B, for which distances between their polar atoms are ¡ 2.8 $\mathring{A}$

627-637 and 567-574 residues), each of which interacts with one site (1-12 residues) on the N-terminus of actin (Sutoh, 1983). We showed that S1 can be cross-linked simultaneously to two actins via each site (Andreev et al., 1993). The results of time-resolved cross-linking of S1 and F-actin upon their rapid mixing in stopped flow apparatus directly demonstrated that myosin head initially binds through the loop 635-647 (binding site #1) to the N-terminus of one actin and, then through the loop 567-574 (binding site #2) to the N-terminus of the second actin (Andreev and Reshetnyak, 2007). From cross-linking data, however, it is impossible to determine the relative location of S1 on the F-actin surface, which was the main goal of the conducted computational study. We reconstructed the missing loops in atomic structure of chicken S1 and found that the average distance between binding sites 1 and 2 is about 5 nm, which matches a distance between N-termini of two actin monomers located in the same strand. At the same time, this distance is much shorter than distance of about 9 nm between N-termini of two monomers located on different strands. In this work we investigated two possible pathways of S1 binding with two actin monomers in F-actin. The docking procedure based on minimization of binding free energy allowed us to reveal that the complex of S1 with two actins has much lower energy when S1 neck is oriented towards the barbed end (model A) rather than to the pointed end (model B). This finding and our previous time-resolved cross-linking data provide a base for the model of force generation as a result of the sequential formation of the actomyosin interface toward the barbed end of F-actin. It can also explain the directionality of myosin movement on the actin filament. Previously published actomyosin complex models were based on the fitting of the atomic structures of S1 and actin into the envelope of F-actin decorated with S1 reconstructed from electron microscopy (EM) data. EM reconstruction was done with actin filament

saturated with S1, where S1 interacts mostly only with one actin. The binding with two actins is possible only at partial decoration of F-actin with S1, where S1 molecules do not interfere with each other. Attempts to directly fit S1 structures into actin filament fully decorated with S1 were not successful and revealed a significant structural collisions (Rayment et al., 1993). To get satisfactory fit it was necessary to artificially alter a shape of S1 by closing a cleft in the 50 kDa domain of S1 (Holmes, 2010; Liu et al., 2006; Rayment et al., 1993). The obtained complex does not represent a physiological state of rigor complex, when density of the bound myosin heads on the actin filament is low. Our model presents a more physiological state, where there is no interference between neighboring S1 molecules, and it does not require altering the shape or closing the cleft in S1. We found that in the model of the S1 structure with the closed cleft published by Liu (Liu et al., 2006) the distance between loops 1 and 2 is too short (3.8 nm) for S1 to be connected to two N-termini of two actin monomers in the filament. The structure with the closed cleft is probably realized in complexes at full decoration of F-actin with S1. The cross-linking data demonstrated that at full decoration, S1 cross-links only with one monomer predominately via loop of binding site #1, while at low degree of decoration S1 cross-links with two actins via both loops of binding sites #1 and #2. The computational docking explained this difference in cross-linking products at fully and partially decoration of F-actin with S1.

The sequential development of actomyosin interface in one direction might play an important role in force generation and unidirectional movement of actin filament relative to the myosin molecule. The formation of the actomyosin interface is accompanied with a release of significant amount of energy and it seems unlikely that this energy would be wasted during muscle contraction. Indeed, it was demonstrated recently that the force exerted by the muscle

cross-bridge depends directly on the strength of actomyosin bond (Liu et al., 2004).

References

Andreev, O. A., Reshetnyak, Y. K., 2007. Mechanism of formation of actomyosin interface. J Mol Biol 365, 551-4, doi:S0022-2836(06)01353-2 [pii]10.1016/j.jmb.2006.10.014.

Andreev, O. A., Andreeva, A. L., Markin, V. S., Borejdo, J., 1993. Two different rigor complexes of myosin subfragment 1 and actin. Biochemistry 32, 12046-53.

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S., Karplus, M., 1983. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. Journal of Computational Chemistry 4, 187-217, doi:10.1002/jcc.540040211.

Brooks, B. R., Brooks, C. L., 3rd, Mackerell, A. D., Jr., Nilsson, L., Petrella, R. J., Roux, B., Won, Y., Archontis, G., Bartels, C., Boresch, S., Caflisch, A., Caves, L., Cui, Q., Dinner, A. R., Feig, M., Fischer, S., Gao, J., Hodoscek, M., Im, W., Kuczera, K., Lazaridis, T., Ma, J., Ovchinnikov, V., Paci, E., Pastor, R. W., Post, C. B., Pu, J. Z., Schaefer, M., Tidor, B., Venable, R. M., Woodcock, H. L., Wu, X., Yang, W., York, D. M., Karplus, M., 2009. CHARMM: the biomolecular simulation program. J Comput Chem 30, 1545-614, doi:10.1002/jcc.21287.

Cooke, R., 1986. The mechanism of muscle contraction. CRC Crit Rev Biochem 21, 53-118.

Holmes, K. C., 2010. 50 years of fiber diffraction. J Struct Biol 170, 184-91, doi:10.1016/j.jsb.2010.01.004S1047-8477(10)00016-X [pii]. Homma, K., Yoshimura, M., Saito, J., Ikebe, R., Ikebe, M., 2001. The core of the motor domain determines the direction of myosin movement. Nature 412, 831-4, doi:10.1038/3509059735090597 [pii].

Huxley, A. F., Niedergerke, R., 1954. Structural changes in muscle during contraction; interference microscopy of living muscle fibres. Nature 173, 971-3.

Huxley, A. F., Simmons, R. M., 1971. Proposed mechanism of force generation in striated muscle. Nature 233, 533-8.

Huxley, H., Hanson, J., 1954. Changes in the cross-striations of muscle during contraction and stretch and their structural interpretation. Nature 173, 973-6.

Liu, J., Reedy, M. C., Goldman, Y. E., Franzini-Armstrong, C., Sasaki, H., Tregear, R. T., Lucaveche, C., Winkler, H., Baumann, B. A., Squire, J. M., Irving, T. C., Reedy, M. K., Taylor, K. A., 2004. Electron tomography of fast frozen, stretched rigor fibers reveals elastic distortions in the myosin crossbridges. J Struct Biol 147, 268-82, doi:10.1016/j.jsb.2004.03.008S1047847704000668 [pii].

Liu, Y., Scolari, M., Im, W., Woo, H. J., 2006. Protein-protein interactions in actin-myosin binding and structural effects of R405Q mutation: a molecular dynamics study. Proteins 64, 156-66, doi:10.1002/prot.20993.

Lymn, R. W., Taylor, E. W., 1971. Mechanism of adenosine triphosphate hydrolysis by actomyosin. Biochemistry 10, 4617-24.

Ma, X. H., Wang, C. X., Li, C. H., Chen, W. Z., 2002. A fast empirical approach to binding free energy calculations based on protein interface information. Protein Eng 15, 677-81.

Oda, T., Iwasa, M., Aihara, T., Maeda, Y., Narita, A., 2009. The nature of the globular- to fibrous-actin transition. Nature 457, 441-5, doi:10.1038/nature07685nature07685 [pii].

Popov, P., Ritchie, D. W., Grudinin, S., 2013. DockTrina: Docking triangular protein trimers. Proteins, doi:10.1002/prot.24344. 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.

Shen, M. Y., Sali, A., 2006. Statistical potential for assessment and prediction of protein structures. Protein Sci 15, 2507-24, doi:15/11/2507 [pii]10.1110/ps.062416606.

Sutoh, K., 1983. Mapping of actin-binding sites on the heavy chain of myosin subfragment 1. Biochemistry 22, 1579-85.

Tsiavaliaris, G., Fujita-Becker, S., Manstein, D. J., 2004. Molecular engineering of a backwards-moving myosin motor. Nature 427, 558-61, doi:10.1038/nature02303nature02303 [pii].

Tsodikov, O. V., Record, M. T., Jr., Sergeev, Y. V., 2002. Novel computer program for fast exact calculation of accessible and molecular surface areas and average surface curvature. J Comput Chem 23, 600-9, doi:10.1002/jcc.10061.

Vajda, S., Kozakov, D., 2009. Convergence and combination of methods in protein-protein docking. Curr Opin Struct Biol 19, 164-70, doi:10.1016/j.sbi.2009.02.008S0959 440X(09)00028-1 [pii].

Wells, A. L., Lin, A. W., Chen, L. Q., Safer, D., Cain, S. M., Hasson, T., Carragher, B. O., Milligan, R. A., Sweeney, H. L., 1999. Myosin VI is an actin-based motor that moves backwards. Nature 401, 505-8, doi:10.1038/46835.

Yanagida, T., Kitamura, K., Tanaka, H., Hikikoshi Iwane, A., Esaki, S., 2000. Single molecule analysis of the actomyosin motor. Curr Opin Cell Biol 12, 20-5, doi:S0955-0674(99)00052-6 [pii].

Selected Code from Rigid Body Docking Program

```
public class Clashes
{
    public void calculateClashes(Protein actin,
    Protein myosin, String myosinInputFilename, String actinInputFilename)
    {
        OutputFileWriter FileWriter = new OutputFileWriter("/Users/..."
        + getDateTime() + ".txt");


        for(int k = 0; k < lenthMyosin; k++)
        {
            for(int c = 0; c < lenthActin; c++)
            {
                String actinType =
                actin.getAtomType(c);
                double actinX =
                actin.getXcoordinate(c);
                double actinY =
                actin.getYcoordinate(c);
                double actinZ =
                actin.getZcoordinate(c);

                String myosinType =
                myosin.getAtomType(k);
                double myosinX =
                myosin.getXcoordinate(k);
                double myosinY =
                myosin.getYcoordinate(k);
                double myosinZ =
                myosin.getZcoordinate(k);

                double temp =
                calculateDistance
                (myosinX, myosinY, myosinZ,
                actinX, actinY, actinZ);



                if(temp <= 1.8)
                {
                    if(actinType == null)
```

```
{
    System.out.println
    ("c = " + c + " k = "
     + k + " actin is null");
}//end if(actinType ==null)
else if(myosinType == null)
{
    System.out.println("c = "
    + c + " k = " + k + " myosin is null");
}//end else if(myosinType ==null)

if (temp <= 1)
{
    //numClashes++;
    interactionONE++;
    //FileWriter.
    writeToFile
    ("Clash#: " + numClashes);
    FileWriter.writeToFile
    ("ClashONE#: " + interactionONE);
    FileWriter.writeToFile
    ("Distance less then 1");
    FileWriter.writeToFile
    ("Atom   " + myosin.getAtomNum(k)
     + " " + myosin.getAtomName(k)
     + " " +  myosin.getResNum(k)
     + " " + myosin.getXcoordinate(k)
      + " " + myosin.getYcoordinate(k)
       + " " + myosin.getZcoordinate(k)
        + " " + myosin.getAtomType(k));
    FileWriter.writeToFile("Atom   " +
    actin.getAtomNum(c) + " " +
    actin.getAtomName(c) + " " +
    actin.getResNum(c) + " " +
    actin.getXcoordinate(c) + " "
    + actin.getYcoordinate(c)
    + " " + actin.getZcoordinate(c)
     + " " + actin.getAtomType(c));
    FileWriter.
    writeToFile("distance: " +
    Double.toString(temp));
    FileWriter.writeToFile(" ");
}//end else if (temp <= 1)
else if((actinType.startsWith("H")
```

```
== true) &&
(myosinType.startsWith("H") == true))
{
    numClashes++;
    interactionHH++;
    FileWriter.writeToFile
    ("Clash#: " + numClashes);
    FileWriter.writeToFile
    ("ClashHH#: " + interactionHH);
    FileWriter.writeToFile
    ("H_H Interaction");
    FileWriter.writeToFile
    ("Atom   " + myosin.getAtomNum(k)
     + " " + myosin.getAtomName(k)
     + " " +  myosin.getResNum(k)
      + " " + myosin.getXcoordinate(k)
       + " " + myosin.getYcoordinate(k)
        + " " + myosin.getZcoordinate(k)
         + " " + myosin.getAtomType(k));
    FileWriter.writeToFile("Atom   " +
     actin.getAtomNum(c) + " " +
     actin.getAtomName(c) + " " +
     actin.getResNum(c) + " " +
      actin.getXcoordinate(c) +
      " " + actin.getYcoordinate(c)
       + " " + actin.getZcoordinate(c)
        + " " + actin.getAtomType(c));
    FileWriter.writeToFile("distance: "
     + Double.toString(temp));
    FileWriter.writeToFile(" ");
}//end if((actinType.equals("H")
 == true) && (myosinType.equals("H")
 == true))
else if((actinType.startsWith("H")
 == true) &&
 (myosinType.startsWith("C") == true))
{
    numClashes++;
    interactionHC++;
    FileWriter.writeToFile
    ("Clash#: " + numClashes);
    FileWriter.writeToFile
    ("ClashHC#: " + interactionHC);
    FileWriter.writeToFile
```

```
    ("H_C Interaction");
    FileWriter.writeToFile
    ("Atom   " + myosin.getAtomNum(k)
     + " " + myosin.getAtomName(k)
      + " " +  myosin.getResNum(k)
      + " " + myosin.getXcoordinate(k)
      + " " + myosin.getYcoordinate(k)
       + " " + myosin.getZcoordinate(k)
       + " " + myosin.getAtomType(k));
    FileWriter.writeToFile("Atom   "
    + actin.getAtomNum(c) + " "
     + actin.getAtomName(c) + " " +
      actin.getResNum(c) + " " +
       actin.getXcoordinate(c) +
       " " + actin.getYcoordinate(c)
       + " " + actin.getZcoordinate(c)
        + " " + actin.getAtomType(c));
    FileWriter.writeToFile("distance: "
     + Double.toString(temp));
    FileWriter.writeToFile(" ");
}//end else if((actinType.equals("H")
 == true) && (myosinType.equals("C")
  == true))
else if((actinType.startsWith("C")
 == true) && (myosinType.startsWith("H")
  == true))
{
    numClashes++;
    interactionHC++;
    FileWriter.writeToFile
    ("Clash#: " + numClashes);
    FileWriter.writeToFile
    ("ClashHC#: " + interactionHC);
    FileWriter.writeToFile
    ("C_H Interaction");
    FileWriter.writeToFile
    ("Atom   " + myosin.getAtomNum(k)
     + " " + myosin.getAtomName(k)
      + " " +  myosin.getResNum(k)
       + " " + myosin.getXcoordinate(k)
        + " " + myosin.getYcoordinate(k)
        + " " + myosin.getZcoordinate(k)
         + " " + myosin.getAtomType(k));
    FileWriter.w
```

```
    riteToFile("Atom   " +
    actin.getAtomNum(c) + " "
    + actin.getAtomName(c) + " "
    +  actin.getResNum(c) + " " +
     actin.getXcoordinate(c)
     + " " + actin.getYcoordinate(c)
     + " " + actin.getZcoordinate(c)
     + " " + actin.getAtomType(c));
    FileWriter.
    writeToFile("distance: "
    + Double.toString(temp));
    FileWriter.writeToFile(" ");
}//end else if((actinType.equals("C")
 == true) && (myosinType.equals("H")
 == true))
else if((actinType.startsWith("C")
== true) &&
(myosinType.startsWith("C")
 == true))
{
    numClashes++;
    interactionCC++;
    FileWriter.writeToFile
    ("Clash#: " + numClashes);
    FileWriter.writeToFile
    ("ClashCC#: " + interactionCC);
    FileWriter.writeToFile
    ("C_C Interaction");
    FileWriter.writeToFile
    ("Atom   " + myosin.getAtomNum(k)
     + " " + myosin.getAtomName(k)
      + " " +  myosin.getResNum(k)
      + " " + myosin.getXcoordinate(k)
       + " " + myosin.getYcoordinate(k)
        + " " + myosin.getZcoordinate(k)
        + " " + myosin.getAtomType(k));
    FileWriter.writeToFile("Atom   "
    + actin.getAtomNum(c) + " "
     + actin.getAtomName(c) +
     " " +  actin.getResNum(c)
     + " " + actin.getXcoordinate(c)
      + " " + actin.getYcoordinate(c)
       + " " + actin.getZcoordinate(c)
       + " " + actin.getAtomType(c));
```

```
    FileWriter.writeToFile
    ("distance: " + Double.toString(temp));
    FileWriter.writeToFile(" ");
}//end else if((actinType.equals("C")...
else if((actinType.startsWith("O")
 == true) &&
 (myosinType.startsWith("O")
  == true))
{
    numClashes++;
    interactionOO++;
    FileWriter.writeToFile
    ("Clash#: " + numClashes);
    FileWriter.writeToFile
    ("ClashOO#: " + interactionOO);
    FileWriter.writeToFile
    ("O_O Interaction");
    FileWriter.writeToFile
    ("Atom   " + myosin.getAtomNum(k)
     + " " + myosin.getAtomName(k)
      + " " +  myosin.getResNum(k)
       + " " + myosin.getXcoordinate(k)
        + " " + myosin.getYcoordinate(k)
         + " " + myosin.getZcoordinate(k)
          + " " + myosin.getAtomType(k));
    FileWriter.
    writeToFile("Atom   "
     + actin.getAtomNum(c) + " "
     + actin.getAtomName(c) + " "
     +  actin.getResNum(c) + " "
     + actin.getXcoordinate(c)
     + " " + actin.getYcoordinate(c)
      + " " + actin.getZcoordinate(c)
       + " " + actin.getAtomType(c));
    FileWriter.writeToFile
    ("distance: "
    + Double.toString(temp));
    FileWriter.
    writeToFile(" ");
}//end else if((actinType.equals("O")...
else if((actinType.startsWith("N"
== true)
 && (myosinType.startsWith("N")
 == true))
```

```
{
    numClashes++;
    interactionNN++;
    FileWriter.writeToFile
    ("Clash#: " + numClashes);
    FileWriter.writeToFile
    ("ClashNN#: " + interactionNN);
    FileWriter.writeToFile
    ("N_N Interaction");
    FileWriter.writeToFile
    ("Atom   " + myosin.getAtomNum(k)
    + " " + myosin.getAtomName(k)
    + " " +  myosin.getResNum(k)
    + " " + myosin.getXcoordinate(k)
    + " " + myosin.getYcoordinate(k)
    + " " + myosin.getZcoordinate(k)
    + " " + myosin.getAtomType(k));
    FileWriter.writeToFile("Atom   "
     + actin.getAtomNum(c) + " "
      + actin.getAtomName(c) + " "
       +  actin.getResNum(c) + " "
        + actin.getXcoordinate(c)
        + " " + actin.getYcoordinate(c)
         + " " + actin.getZcoordinate(c)
          + " " + actin.getAtomType(c));
    FileWriter.
    writeToFile("distance: "
     + Double.toString(temp));
    FileWriter.writeToFile(" ");
}//end else if((actinType.equals("N") ...
else
{
    interactionOTHER++;
    FileWriter.writeToFile
    ("ClashOTHER#: "
    + interactionOTHER);
    FileWriter.writeToFile
    ("Other " + "actin type: "
    + actin.getAtomType(c)
    + ", myosin type: "
     + myosin.getAtomType(k));
    FileWriter.writeToFile("Atom   "
    + myosin.getAtomNum(k) + " "
    + myosin.getAtomName(k) + " "
```

```
                +  myosin.getResNum(k) + " "
                + myosin.getXcoordinate(k)
                + " " + myosin.getYcoordinate(k)
                + " " + myosin.getZcoordinate(k)
                 + " " + myosin.getAtomType(k));
                FileWriter.writeToFile("Atom   "
                 + actin.getAtomNum(c) + " "
                 + actin.getAtomName(c) + " "
                 +  actin.getResNum(c) + " "
                 + actin.getXcoordinate(c)
                 + " " + actin.getYcoordinate(c)
                 + " " + actin.getZcoordinate(c)
                 + " " + actin.getAtomType(c));
                FileWriter.writeToFile("distance: "
                 + Double.toString(temp));
                FileWriter.writeToFile(" ");
            }//end else
        }//end if(temp <= 3)
    }//end for(int c =0; c < lenthActin; c++)
}//end for(int k = 0; k < lenthMyosin; k++)

FileWriter.writeToFile
(actinInputFilename);
FileWriter.writeToFile
(myosinInputFilename);
FileWriter.writeToFile
("Total # of clashes, x < 1 = "
+ interactionONE);
FileWriter.writeToFile
("Total # of clashes = "
+ numClashes);
FileWriter.writeToFile
("Total # of HH clashes = "
+ interactionHH);
FileWriter.writeToFile
("Total # of HC clashes = "
+ interactionHC);
FileWriter.writeToFile
("Total # of CC clashes = "
+ interactionCC);
FileWriter.writeToFile
("Total # of OO clashes = "
+ interactionOO);
FileWriter.writeToFile
```

```
    ("Total # of NN clashes = "
    + interactionNN);
    FileWriter.writeToFile
    ("Total # of other clashes, 1 < x < 1.8 = "
     + interactionOTHER);
    FileWriter.writeToFile
    ("Score: # of clashes
    + 2 * clashes < 1 = "
    + (numClashes +
    (2 * interactionONE)));
    FileWriter.writeToFile
    ("Score: # of clashes +
    # other clashes +
     2 * clashes < 1 = "
     + (numClashes +
     interactionOTHER +
      (2 * interactionONE)));
}//end public void calculateClashes()

private  final static String getDateTime()
{
    DateFormat df =
    new SimpleDateFormat
    ("yyyy-MM-dd_hh:mm:ss");
    df.setTimeZone
    (TimeZone.getTimeZone("PST"));
    return df.format(new Date());
}

private static double
calculateDistance(double x1,
double y1, double z1, double x2,
 double y2,  double z2)
{
    double x = Math.sqrt
    (Math.pow((x1-x2),2)+
    Math.pow((y1-y2),2)+
    Math.pow((z1-z2),2));

     x = Math.abs(x); //
      > 0 ? Math.floor(x * 10)
      / 10.0 : Math.ceil(x * 10)
       / 10.0);
```

```
        return x;
    }//end double calculateDistance...

    /*private double H_H = 2.0;
    private double H_C = 2.5;
    private double C_C = 3.0;
    private double O_O = 2.7;
    private double N_N = 2.7;*/

    int numClashes = 0;
    int interactionONE = 0;
    int interactionHH = 0;
    int interactionHC = 0;
    int interactionCC = 0;
    int interactionOO = 0;
    int interactionNN = 0;
    int interactionOTHER = 0;
    private int lenthMyosin = 6722;
    private int lenthActin = 14648;
}//public class Clashes
```

```
public class Protein
{
    public Protein(int arraySize)
    {
        proteinInfo = new proteinResidueData[arraySize];

        for(int c = 0; c < arraySize; c++)
        {
            proteinInfo[c] = new proteinResidueData();
        }//end for(int c = 0; c < arraySize: c++)
    }//end public Protein(int arraySize)

    public String getWholeLine(int position)
    {
        return proteinInfo[position].getWholeLine();
    }//end public String getwholeLine()

    public String getAtomType(int position)
    {
        return proteinInfo[position].getAtomType();
    }//end public String getAtomType()

    public int getResNum(int position)
    {
        return proteinInfo[position].getResNum();
    }//end public int getAtomNum()

    public double getXcoordinate(int position)
    {
        return proteinInfo[position].getXcoordinate();
    }//end public String getXcoordinate()

    public double getYcoordinate(int position)
    {
        return proteinInfo[position].getYcoordinate();
    }//end public String getYcoordinate()

    public double getZcoordinate(int position)
    {
        return proteinInfo[position].getZcoordinate();
    }//end public String getZcoordinate()

    public double getASA(int position)
    {
```

```
    return proteinInfo[position].getASA();
}//end public String getZcoordinate()

public String getAtomName(int position)
{
    return proteinInfo[position].getAtomResName();
}//end public String getAtomName()

public int getAtomNum(int position)
{
    return proteinInfo[position].getAtomNum();
}//end public int getAtomNum()

public void setAtomType(String type, int position)
{
    proteinInfo[position].setAtomType(type.trim());
}//end public void setAtomType(String type, int position)

public void setResNum(int num, int position)
{
    proteinInfo[position].setResNum(num);
}//end public void setAtomNum(int num, int position)

public void setXcoordinate(double x, int position)
{
    proteinInfo[position].setXcoordinate(x);
}//end public void setXcoordiante(double x, int position)

public void setYcoordinate(double y, int position)
{
    proteinInfo[position].setYcoordinate(y);
}//end public void setYcoordiante(double y, int position)

public void setZcoordinate(double z, int position)
{
    proteinInfo[position].setZcoordinate(z);
}//end public void setZcoordiante(double z, int position)

public void setASA(double ASAtemp, int position)
{
    proteinInfo[position].setASA(ASAtemp);
}//end public void setZcoordiante(double ASAtemp, int position)

public void setAtomName(String name, int position)
```

```
    {
        proteinInfo[position].setAtomResName(name);
    }//end public void setAtomName(String name, int position)

    public void setAtomNum(int num, int position)
    {
        proteinInfo[position].setAtomNum(num);
    }//end public void setAtomNum(int num, int position)

    public void setWholeLine(String temp, int position)
    {
     proteinInfo[position].setWholeLine(temp);
    }//end public void setWholeLine(String temp)

    private proteinResidueData[] proteinInfo;
}//end public class Protein
```

```
public class proteinResidueData
{
public String getWholeLine()
    {
        return wholeLine;
    }//end public String getwholeLine()

public String getAtomType()
    {
        return atomType;
    }//end public String getAtomType()

public int getResNum()
    {
        return resNum;
    }//end public int getResNum()

public double getASA()
    {
        return ASA;
    }//end public double getASA()

    public double getXcoordinate()
    {
        return xCoordinate;
    }//end public double getXcoordinate()

    public double getYcoordinate()
    {
        return yCoordinate;
    }//end public double getYcoordinate()

    public double getZcoordinate()
    {
        return zCoordinate;
    }//end public double getZcoordinate()

    public String getAtomResName()
    {
        return atomResName;
    }//end public String getAtomName()

    public int getAtomNum()
    {
```

```
    return atomNum;
}//end public int getAtomNum()


public void setResNum(int num)
{
    resNum = num;
}//end public void setResNum(int num)


public void setXcoordinate(double x)
{
    xCoordinate = x;
}//end public void setXcoordinate(double x)

public void setYcoordinate(double y)
{
    yCoordinate = y;
}//end public void setXcoordinate(double y)

public void setZcoordinate(double z)
{
    zCoordinate = z;
}//end public void setXcoordniate(double z)

public void setASA(double ASAtemp)
{
    ASA = ASAtemp;
}//end public void setXcoordniate(double z)

public void setAtomResName(String name)
{
    atomResName = name.trim();
}//end public void setAtomName(String name)

public void setAtomNum(int num)
{
    atomNum = num;
}//end public void setAtomNum(int num)

public void setAtomType(String type)
{
    atomType = type;
}//end public setAtomType(String type)
```

```
    public void setWholeLine(String temp)
    {
        atomType = temp;
    }//end public void setWholeLine(String temp)

    private double ASA = 0;
    private double xCoordinate = 0;
    private double yCoordinate = 0;
    private double zCoordinate = 0;
    private int resNum = 0;
    private String atomResName = null;
    private int atomNum = 0;
    private String atomType = null;
    private String wholeLine = null;
}//end public class proteinResidue
```

```
public class ReadFromFile
{


public void readFile(String actinFile,
String myosinFile)
{
  FileWriter.writeToFile("");
        FileWriter.writeToFile("REMARK File name: "
        +"/Users/....txt");
        FileWriter.writeToFile(actinFile);
        FileWriter.writeToFile(myosinFile);
        FileWriter.writeToFile("REMARK --------
        -------------
        --------------
        ---------------
        ---------------------------");

         try
         {
            FileInputStream fstream =
            new FileInputStream(myosinFile);
            BufferedReader br =
            new BufferedReader(new InputStreamReader(fstream));
            String strLine = new String();

            while(((strLine =
            br.readLine())!= null))
            {
             if(strLine.substring(0,3).
             equals("ATO")
             || strLine.substring(0,3).
             equals("HET") )
             {
             String atomType =
             strLine.substring(13,16).trim();
             String temp =
             (strLine.substring(22,27)).replace(" ", "");
                int resNum =
                Integer.parseInt(temp);

             if(resNum == 567
             && atomType.equalsIgnoreCase("NZ"))
                {
```

```
x567 =
Double.parseDouble
(strLine.substring(30,38));
y567 =
Double.parseDouble
(strLine.substring(39,46));
z567 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("567: "
 + x567 + "," + y567
  + ", " + z567);
   }//end if(resNum ...
else if(resNum == 569
&& atomType.equalsIgnoreCase("NZ"))
   {
x569 =
Double.parseDouble
(strLine.substring(30,38));
y569 =
Double.parseDouble
(strLine.substring(39,46));
z569 =
Double.parseDouble
(strLine.substring(47,55));
System.out.
println("569: " + x569 + ","
+ y569 + ", " + z569);
   }//end if(resNum == 569
   && atomType.equalsIgnoreCase("NZ"))
else if(resNum == 572 &&
atomType.equalsIgnoreCase("NZ"))
   {
x572 =
Double.parseDouble
(strLine.substring(30,38));
y572 =
Double.parseDouble
(strLine.substring(39,46));
z572 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("572: " + x572 + ","
+ y572 + ", " + z572);
```

```
   }//end if(resNum == 572...
else if(resNum == 574 &&
atomType.equalsIgnoreCase("NZ"))
   {
x574 =
Double.parseDouble
(strLine.substring(30,38));
y574 =
Double.parseDouble
(strLine.substring(39,46));
z574 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("574: " + x574 + ","
+ y574 + ", " + z574);
   }//end if(resNum == 574...
else if(resNum == 636
&& atomType.equalsIgnoreCase("NZ"))
   {
x636 =
Double.parseDouble
(strLine.substring(30,38));
y636 =
Double.parseDouble
(strLine.substring(39,46));
z636 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("636: "
 + x636 + ","
+ y636 + ", "
+ z636);
   }//end if(resNum == 636...
else if(resNum == 637
 && atomType.equalsIgnoreCase("NZ"))
   {
x637 =
Double.parseDouble
(strLine.substring(30,38));
y637 =
Double.parseDouble
(strLine.substring(39,46));
z637 =
 Double.parseDouble
```

```
 (strLine.substring(47,55));
System.out.println("637: "
 + x637 + ","
+ y637 + ", "
+ z637);
   }//end if(resNum == 63...
else if(resNum == 640 &&
atomType.equalsIgnoreCase("NZ"))
   {
x640 =
Double.parseDouble
(strLine.substring(30,38));
y640 =
Double.parseDouble
(strLine.substring(39,46));
z640 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("640: "
 + x640 + ","
 + y640 + ", " + z640);
   }//end if(resNum == 640...
else if(resNum == 641
&& atomType.equalsIgnoreCase("NZ"))
   {
x641 =
 Double.parseDouble
 (strLine.substring(30,38));
y641 =
Double.parseDouble
(strLine.substring(39,46));
z641 =
Double.parseDouble
(strLine.substring(47,55));
System.out.
println("641: " + x641 + ","
+ y641 + ", " + z641);
   }//end if(resNum == 641 &&
   atomType.equalsIgnoreCase("NZ"))
else if(resNum == 642 &&
atomType.equalsIgnoreCase("NZ"))
   {
x642 =
Double.parseDouble
```

```
(strLine.substring(30,38));
y642 =
Double.parseDouble
(strLine.substring(39,46));
z642 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("642: "
 + x642 + ","
+ y642 + ", " + z642);
   }//end if(resNum == 642
   && atomType.equalsIgnoreCase("NZ"))
else if(resNum == 644
 && atomType.equalsIgnoreCase("NZ"))
   {
x644 =
Double.parseDouble
(strLine.substring(30,38));
y644 =
Double.parseDouble
(strLine.substring(39,46));
z644 =
 Double.parseDouble
 (strLine.substring(47,55));
System.out.println("644: "
 + x644 + ","
+ y644 + ", "
 + z644);
   }//end if(resNum == 644
    && atomType.equalsIgnoreCase("NZ"))
else if(resNum == 647
&& atomType.equalsIgnoreCase("OE1"))
   {
x647 =
Double.parseDouble
(strLine.substring(30,38));
y647 =
Double.parseDouble
(strLine.substring(39,46));
z647 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println
("647: " + x647 + ","
```

```
    + y647 + ", " + z647);
       }//end if(resNum == 644...
    }//end if(strLine.substring(0,3)...
   }//end while(((strLine = br...


 //Close the input stream
   fstream.close();
}//end try
//Catch exception if any
catch (Exception tryException)
{
 System.err.println("Error: "
  + tryException.getMessage());
}//end catch (Exception e)



try
{
 FileInputStream newfstream =
 new FileInputStream(actinFile);
    BufferedReader newbr =
    new BufferedReader
    (new InputStreamReader(newfstream));
    String strLine = new String();

   while(((strLine =
   newbr.readLine())!= null))
   {
    if(strLine.substring(0,3).
    equals("ATO"))
    {
    String atomType =
    strLine.substring(13,16).trim();
    String temp =
    (strLine.substring(22,27)).replace(" ", "");
       int resNum =
       Integer.parseInt(temp);
       System.out.println(atomType + ", " + resNum);

     //monomer e
    if(resNum == 1001 &&
    atomType.equalsIgnoreCase("OD1"))
```

```
                   {
                u1_OD1 =
                Double.parseDouble
                (strLine.substring(30,38));
                v1_OD1 =
                Double.parseDouble
                (strLine.substring(39,46));
                w1_OD1 =
                Double.parseDouble
                (strLine.substring(47,55));
                System.out.println
                ("1001-OD1: " + u1_OD1
                + "," + v1_OD1 +
                 ", " + w1_OD1);
}//end (resNum == 1001...
                else if(resNum == 1001 &&
                atomType.equalsIgnoreCase("OD2"))
                   {
                u1_OD2 =
                Double.parseDouble
                (strLine.substring(30,38));
                v1_OD2 =
                Double.parseDouble
                (strLine.substring(39,46));
                w1_OD2 =
                Double.parseDouble
                (strLine.substring(47,55));
                System.out.println("1001-OD2: "
                 + u1_OD2
                + "," + v1_OD2 + ", "
                + w1_OD2);

                   }//end (resNum == 1001 &&
                    atomType.equalsIgnoreCase("OD2"))
                else if(resNum == 1002 &&
                 atomType.equalsIgnoreCase("OE1"))
                   {
                u2_OE1 =
                Double.parseDouble
                (strLine.substring(30,38));
                v2_OE1 =
                Double.parseDouble
                (strLine.substring(39,46));
                w2_OE1 =
```

```
Double.parseDouble
(strLine.substring(47,55));
System.out.println
("1002-OE1: " + u2_OE1
+ "," + v2_OE1 + ", "
+ w2_OE1);
   }//end (resNum == 1002...)
else if(resNum == 1002 &&
atomType.equalsIgnoreCase("OE2"))
   {
u2_OE2 =
 Double.parseDouble
 (strLine.substring(30,38));
v2_OE2 =
Double.parseDouble
(strLine.substring(39,46));
w2_OE2 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("1002-OE2: "
 + u2_OE2
+ "," + v2_OE2 + ", "
 + w2_OE2);
   }//end (resNum == 1002 ...
else if(resNum == 1003 &&
 atomType.equalsIgnoreCase("OD1"))
   {
u3_OD1 =
Double.parseDouble
(strLine.substring(30,38));
v3_OD1 =
Double.parseDouble
(strLine.substring(39,46));
w3_OD1 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("1003-OD1: "
+ u3_OD1
+ "," + v3_OD1 + ", "
 + w3_OD1);
   }//end (resNum == 1003...
else if(resNum == 1003 &&
atomType.equalsIgnoreCase("OD2"))
   {
```

```
u3_OD2 =
Double.parseDouble
(strLine.substring(30,38));
v3_OD2 =
Double.parseDouble
(strLine.substring(39,46));
w3_OD2 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("1003-OD2: "
+ u3_OD2
+ "," + v3_OD2 + ", "
 + w3_OD2);
   }//end (resNum == 1003...
else if(resNum == 1004 &&
atomType.equalsIgnoreCase("OE1"))
   {
u4_OE1 =
 Double.parseDouble
 (strLine.substring(30,38));
v4_OE1 =
Double.parseDouble
(strLine.substring(39,46));
w4_OE1 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("1004-OE1: "
+ u4_OE1
+ "," + v4_OE1 + ",
" + w4_OE1);
   }//end (resNum == 1004 ...
else if(resNum == 1004 &&
atomType.equalsIgnoreCase("OE2"))
   {
u4_OE2 =
Double.parseDouble
(strLine.substring(30,38));
v4_OE2 =
Double.parseDouble
(strLine.substring(39,46));
w4_OE2 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("1004-OE2: "
```

```
 + u4_OE2
+ "," + v4_OE2 + ", "
+ w4_OE2);
   }//end (resNum == 1004 &&
   atomType.equalsIgnoreCase("OE2"))

   //monomer g
if(resNum == 3001 &&
atomType.equalsIgnoreCase("OD1"))
   {
x1_OD1 =
Double.parseDouble
(strLine.substring(30,38));
y1_OD1 =
Double.parseDouble
(strLine.substring(39,46));
z1_OD1 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("3001-OD1: "
+ x1_OD1
+ "," + y1_OD1 + ", "
 + z1_OD1);
   }//end (resNum == 3001...
else if(resNum == 3001 &&
 atomType.equalsIgnoreCase("OD2"))
   {
x1_OD2 =
 Double.parseDouble
 (strLine.substring(30,38));
y1_OD2 =
Double.parseDouble
(strLine.substring(39,46));
z1_OD2 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("3001-OD2: "
+ x1_OD2
+ "," + y1_OD2 + ", "
 + z1_OD2);
   }//end (resNum == 3001...
else if(resNum == 3002 &&
 atomType.equalsIgnoreCase("OE1"))
   {
```

```
x2_OE1 =
Double.parseDouble
(strLine.substring(30,38));
y2_OE1 =
Double.parseDouble
(strLine.substring(39,46));
z2_OE1 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("3002-OE1: " +
x2_OE1
+ "," + y2_OE1 + ", "
+ z2_OE1);
 }//end (resNum == 3002..
else if(resNum == 3002 &&
atomType.equalsIgnoreCase("OE2"))
   {
x2_OE2 =
Double.parseDouble
(strLine.substring(30,38));
y2_OE2 =
Double.parseDouble
(strLine.substring(39,46));
z2_OE2 =
 Double.parseDouble
 (strLine.substring(47,55));
System.out.println("3002-OE2: "
 + x2_OE2
+ "," + y2_OE2 + ", "
 + z2_OE2);
   }//end (resNum == 3002
    && atomType.equalsIgnoreCase("OE2"))
else if(resNum == 3003 &&
atomType.equalsIgnoreCase("OD1"))
   {
x3_OD1 =
Double.parseDouble
(strLine.substring(30,38));
y3_OD1 =
Double.parseDouble
(strLine.substring(39,46));
z3_OD1 =
 Double.parseDouble
 (strLine.substring(47,55));
```

```
System.out.println("3003-OD1: "
+ x3_OD1
+ "," + y3_OD1 + ", "
+ z3_OD1);
   }//end (resNum == 3003...
else if(resNum == 3003 &&
atomType.equalsIgnoreCase("OD2"))
   {
x3_OD2 =
Double.parseDouble
(strLine.substring(30,38));
y3_OD2 =
Double.parseDouble
(strLine.substring(39,46));
z3_OD2 =
 Double.parseDouble
 (strLine.substring(47,55));
System.out.println("3003-OD2: " +
 x3_OD2
+ "," + y3_OD2 + ", "
+ z3_OD2);
   }//end (resNum == 3003...
else if(resNum == 3004 &&
 atomType.equalsIgnoreCase("OE1"))
   {
x4_OE1 =
Double.parseDouble
(strLine.substring(30,38));
y4_OE1 =
Double.parseDouble
(strLine.substring(39,46));
z4_OE1 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("3004-OE1: "
 + x4_OE1
+ "," + y4_OE1 + ", "
+ z4_OE1);
   }//end (resNum == 3004...
else if(resNum == 3004 &&
 atomType.equalsIgnoreCase("OE2"))
   {
x4_OE2 =
Double.parseDouble
```

```
(strLine.substring(30,38));
y4_OE2 =
Double.parseDouble
(strLine.substring(39,46));
z4_OE2 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("3004-OE2: "
+ x4_OE2
+ "," + y4_OE2 + ", "
 + z4_OE2);
   }//end (resNum == 3004 &&
   atomType.equalsIgnoreCase("OE2"))

// monomer I
else if(resNum == 5001 &&
atomType.equalsIgnoreCase("OD1"))
   {
a1_OD1 =
 Double.parseDouble
 (strLine.substring(30,38));
b1_OD1 =
Double.parseDouble
(strLine.substring(39,46));
c1_OD1 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("5001-OD1: "
 + a1_OD1
+ "," + b1_OD1 + ", "
+ c1_OD1);;
   }//end (resNum == 5001...)
else if(resNum == 5001 &&
atomType.equalsIgnoreCase("OD2"))
   {
a1_OD2 =
Double.parseDouble
(strLine.substring(30,38));
b1_OD2 =
Double.parseDouble
(strLine.substring(39,46));
c1_OD2 =
Double.parseDouble
(strLine.substring(47,55));
```

```
System.out.println("5001-OD2: "
+ a1_OD2
+ "," + b1_OD2 + ", "
 + c1_OD2);
   }//end (resNum == 5001 &&
   atomType.equalsIgnoreCase("OD2"))
else if(resNum == 5002 &&
atomType.equalsIgnoreCase("OE1"))
   {
a2_OE1 = Double.parseDouble
(strLine.substring(30,38));
b2_OE1 = Double.parseDouble
(strLine.substring(39,46));
c2_OE1 = Double.parseDouble
(strLine.substring(47,55));
System.out.println("5002-OE1: "
 + a2_OE1
+ "," + b2_OE1 + ", "
 + c2_OE1);
   }//end (resNum == 5002...
else if(resNum == 5002 &&
atomType.equalsIgnoreCase("OE2"))
   {
a2_OE2 =
Double.parseDouble
(strLine.substring(30,38));
b2_OE2 =
Double.parseDouble
(strLine.substring(39,46));
c2_OE2 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("5002-OE2: "
 + a2_OE2
+ "," + b2_OE2 + ", "
+ c2_OE2);
   }//end (resNum == 5002...
else if(resNum == 5003 &&
 atomType.equalsIgnoreCase("OD1"))
   {
a3_OD1 =
Double.parseDouble
(strLine.substring(30,38));
b3_OD1 =
```

```
Double.parseDouble
(strLine.substring(39,46));
c3_OD1 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("5003-OD1: "
 + a3_OD1
+ "," + b3_OD1 + ", " + c3_OD1);
   }//end (resNum == 5003...
else if(resNum == 5003 &&
atomType.equalsIgnoreCase("OD2"))
   {
a3_OD2 =
 Double.parseDouble
 (strLine.substring(30,38));
b3_OD2 =
 Double.parseDouble
 (strLine.substring(39,46));
c3_OD2 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("5003-OD2: "
+ a3_OD2
+ "," + b3_OD2 + ", "
 + c3_OD2);
   }//end (resNum == 5003...
else if(resNum == 5004 &&
atomType.equalsIgnoreCase("OE1"))
   {
a4_OE1 =
Double.parseDouble
(strLine.substring(30,38));
b4_OE1 =
Double.parseDouble
(strLine.substring(39,46));
c4_OE1 =
Double.parseDouble
(strLine.substring(47,55));
System.out.println("3004-OE1: "
+ a4_OE1
+ "," + b4_OE1 + ", "
+ c4_OE1);
   }//end (resNum == 5004 &&
    atomType.equalsIgnoreCase("OE1"))
```

```
                else if(resNum == 5004 &&
                atomType.equalsIgnoreCase("OE2"))
                   {
                a4_OE2 =
                Double.parseDouble
                (strLine.substring(30,38));
                b4_OE2 =
                Double.parseDouble
                (strLine.substring(39,46));
                c4_OE2 =
                Double.parseDouble
                (strLine.substring(47,55));
                System.out.println("5004-OE2: " +
                 a4_OE2
                + "," + b4_OE2 + ", " +
                c4_OE2);
                   }//end (resNum == 5004...


                }//end if(strLine.substring(0,3).equals("ATO"))
               }//end while(((strLine = br.readLine())!= null))

               //Close the input stream
               newfstream.close();
            }//end try
            //Catch exception if any
            catch (Exception tryException)
            {
System.err.println("Error: "
+ tryException.getMessage());
            }//end catch (Exception e)
}//end readFile

public void calcuateDistancesActinMyosin()
{
/*
FileWriter.writeToFile
("636 - 1001-OD1: " +
calculateDistance
(x636, y636, z636, u1_OD1, v1_OD1, w1_OD1));
FileWriter.writeToFile
("636 - 1001-OD2: " +
calculateDistanc
(x636, y636, z636, u1_OD2, v1_OD2, w1_OD2));
```

```
FileWriter.writeToFile("636 - 1002-OE1: " +
calculateDistance
(x636, y636, z636, u2_OE1, v2_OE1, w2_OE1));
FileWriter.writeToFile
("636 - 1002-OE2: " +
calculateDistance
(x636, y636, z636, u2_OE2, v2_OE2, w2_OE2));
FileWriter.writeToFile
("636 - 1003-OD1: " +
calculateDistance
(x636, y636, z636, u3_OD1, v3_OD1, w3_OD1));
FileWriter.writeToFile
("636 - 1003-OD2: " +
calculateDistance
(x636, y636, z636, u3_OD2, v3_OD2, w3_OD2));
FileWriter.writeToFile
("636 - 1004-OE1: " +
calculateDistance
(x636, y636, z636, u4_OE1, v4_OE1, w4_OE1));
FileWriter.writeToFile
("636 - 1004-OE2: " +
calculateDistance
(x636, y636, z636, u4_OE2, v4_OE2, w4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("637 - 1001-OD1: " +
calculateDistance
(x637, y637, z637, u1_OD1, v1_OD1, w1_OD1));
FileWriter.writeToFile
("637 - 1001-OD2: " +
calculateDistance
(x637, y637, z637, u1_OD2, v1_OD2, w1_OD2));
FileWriter.writeToFile
("637 - 1002-OE1: " +
calculateDistance
(x637, y637, z637, u2_OE1, v2_OE1, w2_OE1));
FileWriter.writeToFile
("637 - 1002-OE2: " +
calculateDistance
(x637, y637, z637, u2_OE2, v2_OE2, w2_OE2));
FileWriter.writeToFile
("637 - 1003-OD1: " +
calculateDistance
(x637, y637, z637, u3_OD1, v3_OD1, w3_OD1));
```

```
FileWriter.writeToFile
("637 - 1003-OD2: " +
calculateDistance
(x637, y637, z637, u3_OD2, v3_OD2, w3_OD2));
FileWriter.writeToFile
("637 - 1004-OE1: " +
calculateDistance
(x637, y637, z637, u4_OE1, v4_OE1, w4_OE1));
FileWriter.writeToFile
("637 - 1004-OE2: " +
calculateDistance
(x637, y637, z637, u4_OE2, v4_OE2, w4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("640 - 1001-OD1: " +
calculateDistance
(x640, y640, z640, u1_OD1, v1_OD1, w1_OD1));
FileWriter.writeToFile
("640 - 1001-OD2: " +
calculateDistance
(x640, y640, z640, u1_OD2, v1_OD2, w1_OD2));
FileWriter.writeToFile
("640 - 1002-OE1: " +
calculateDistance
(x640, y640, z640, u2_OE1, v2_OE1, w2_OE1));
FileWriter.writeToFile
("640 - 1002-OE2: " +
calculateDistance
(x640, y640, z640, u2_OE2, v2_OE2, w2_OE2));
FileWriter.writeToFile
("640 - 1003-OD1: " +
calculateDistance
(x640, y640, z640, u3_OD1, v3_OD1, w3_OD1));
FileWriter.writeToFile
("640 - 1003-OD2: " +
calculateDistance
(x640, y640, z640, u3_OD2, v3_OD2, w3_OD2));
FileWriter.writeToFile
("640 - 1004-OE1: " +
calculateDistance
(x640, y640, z640, u4_OE1, v4_OE1, w4_OE1));
FileWriter.writeToFile
("640 - 1004-OE2: " +
calculateDistance
```

```
(x640, y640, z640, u4_OE2, v4_OE2, w4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("641 - 1001-OD1: " +
calculateDistance
(x641, y641, z641, u1_OD1, v1_OD1, w1_OD1));
FileWriter.writeToFile
("641 - 1001-OD2: " +
calculateDistance
(x641, y641, z641, u1_OD2, v1_OD2, w1_OD2));
FileWriter.writeToFile
("641 - 1002-OE1: " +
calculateDistance
(x641, y641, z641, u2_OE1, v2_OE1, w2_OE1));
FileWriter.writeToFile
("641 - 1002-OE2: " +
calculateDistance
(x641, y641, z641, u2_OE2, v2_OE2, w2_OE2));
FileWriter.writeToFile
("641 - 1003-OD1: " +
calculateDistance
(x641, y641, z641, u3_OD1, v3_OD1, w3_OD1));
FileWriter.writeToFile
("641 - 1003-OD2: " +
calculateDistance
(x641, y641, z641, u3_OD2, v3_OD2, w3_OD2));
FileWriter.writeToFile
("641 - 1004-OE1: " +
calculateDistance
(x641, y641, z641, u4_OE1, v4_OE1, w4_OE1));
FileWriter.writeToFile
("641 - 1004-OE2: " +
calculateDistance
(x641, y641, z641, u4_OE2, v4_OE2, w4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("642 - 1001-OD1: " +
calculateDistance
(x642, y642, z642, u1_OD1, v1_OD1, w1_OD1));
FileWriter.writeToFile
("642 - 1001-OD2: " +
calculateDistance
(x642, y642, z642, u1_OD2, v1_OD2, w1_OD2));
FileWriter.writeToFile
```

```
("642 - 1002-OE1: " +
calculateDistance
(x642, y642, z642, u2_OE1, v2_OE1, w2_OE1));
FileWriter.writeToFile
("642 - 1002-OE2: " +
calculateDistance
(x642, y642, z642, u2_OE2, v2_OE2, w2_OE2));
FileWriter.writeToFile
("642 - 1003-OD1: " +
calculateDistance
(x642, y642, z642, u3_OD1, v3_OD1, w3_OD1));
FileWriter.writeToFile
("642 - 1003-OD2: " +
calculateDistance
(x642, y642, z642, u3_OD2, v3_OD2, w3_OD2));
FileWriter.writeToFile
("642 - 1004-OE1: " +
calculateDistance
(x642, y642, z642, u4_OE1, v4_OE1, w4_OE1));
FileWriter.writeToFile
("642 - 1004-OE2: " +
calculateDistance
(x642, y642, z642, u4_OE2, v4_OE2, w4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("647 - 1001-OD1: " +
calculateDistance
(x647, y647, z647, u1_OD1, v1_OD1, w1_OD1));
FileWriter.writeToFile
("647 - 1001-OD2: " +
calculateDistance
(x647, y647, z647, u1_OD2, v1_OD2, w1_OD2));
FileWriter.writeToFile
("647 - 1002-OE1: " +
calculateDistance
(x647, y647, z647, u2_OE1, v2_OE1, w2_OE1));
FileWriter.writeToFile
("647 - 1002-OE2: " +
calculateDistance
(x647, y647, z647, u2_OE2, v2_OE2, w2_OE2));
FileWriter.writeToFile
("647 - 1003-OD1: " +
calculateDistance
(x647, y647, z647, u3_OD1, v3_OD1, w3_OD1));
```

```
FileWriter.writeToFile
("647 - 1003-OD2: " +
calculateDistance
(x647, y647, z647, u3_OD2, v3_OD2, w3_OD2));
FileWriter.writeToFile
("647 - 1004-OE1: " +
calculateDistance
(x647, y647, z647, u4_OE1, v4_OE1, w4_OE1));
FileWriter.writeToFile
("647 - 1004-OE2: " +
calculateDistance
(x647, y647, z647, u4_OE2, v4_OE2, w4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("567 - 3001-OD1: " +
calculateDistance
(x567, y567, z567, x1_OD1, y1_OD1, z1_OD1));
FileWriter.writeToFile
("567 - 3001-OD2: " +
calculateDistance
(x567, y567, z567, x1_OD2, y1_OD2, z1_OD2));
FileWriter.writeToFile
("567 - 3002-OE1: " +
calculateDistance
(x567, y567, z567, x2_OE1, y2_OE1, z2_OE1));
FileWriter.writeToFile
("567 - 3002-OE2: " +
calculateDistance
(x567, y567, z567, x2_OE2, y2_OE2, z2_OE2));
FileWriter.writeToFile
("567 - 3003-OD1: " +
calculateDistance
(x567, y567, z567, x3_OD1, y3_OD1, z3_OD1));
FileWriter.writeToFile
("567 - 3003-OD2: " +
calculateDistance
(x567, y567, z567, x3_OD2, y3_OD2, z3_OD2));
FileWriter.writeToFile
("567 - 3004-OE1: " +
calculateDistance
(x567, y567, z567, x4_OE1, y4_OE1, z4_OE1));
FileWriter.writeToFile
("567 - 3004-OE2: " +
calculateDistance
```

```
(x567, y567, z567, x4_OE2, y4_OE2, z4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("569 - 3001-OD1: " +
calculateDistance
(x569, y569, z569, x1_OD1, y1_OD1, z1_OD1));
FileWriter.writeToFile
("569 - 3001-OD2: " +
calculateDistanc
e(x569, y569, z569, x1_OD2, y1_OD2, z1_OD2));
FileWriter.writeToFile
("569 - 3002-OE1: " +
calculateDistance
(x569, y569, z569, x2_OE1, y2_OE1, z2_OE1));
FileWriter.writeToFile
("569 - 3002-OE2: " +
calculateDistance
(x569, y569, z569, x2_OE2, y2_OE2, z2_OE2));
FileWriter.writeToFile
("569 - 3003-OD1: " +
calculateDistance
(x569, y569, z569, x3_OD1, y3_OD1, z3_OD1));
FileWriter.writeToFile
("569 - 3003-OD2: " +
calculateDistance
(x569, y569, z569, x3_OD2, y3_OD2, z3_OD2));
FileWriter.writeToFile
("569 - 3004-OE1: " +
calculateDistance
(x569, y569, z569, x4_OE1, y4_OE1, z4_OE1));
FileWriter.writeToFile
("569 - 3004-OE2: " +
calculateDistance
(x569, y569, z569, x4_OE2, y4_OE2, z4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("572 - 3001-OD1: " +
calculateDistance
(x572, y572, z572, x1_OD1, y1_OD1, z1_OD1));
FileWriter.writeToFile
("572 - 3001-OD2: " +
calculateDistance
(x572, y572, z572, x1_OD2, y1_OD2, z1_OD2));
FileWriter.writeToFile
```

```
("572 - 3002-OE1: " +
calculateDistance
(x572, y572, z572, x2_OE1, y2_OE1, z2_OE1));
FileWriter.writeToFile
("572 - 3002-OE2: " +
calculateDistance
(x572, y572, z572, x2_OE2, y2_OE2, z2_OE2));
FileWriter.writeToFile
("572 - 3003-OD1: " +
calculateDistance
(x572, y572, z572, x3_OD1, y3_OD1, z3_OD1));
FileWriter.writeToFile
("572 - 3003-OD2: " +
calculateDistance
(x572, y572, z572, x3_OD2, y3_OD2, z3_OD2));
FileWriter.writeToFile
("572 - 3004-OE1: " +
calculateDistance
(x572, y572, z572, x4_OE1, y4_OE1, z4_OE1));
FileWriter.writeToFile
("572 - 3004-OE2: " +
calculateDistance
(x572, y572, z572, x4_OE2, y4_OE2, z4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("574 - 3001-OD1: " +
calculateDistance
(x574, y574, z574, x1_OD1, y1_OD1, z1_OD1));
FileWriter.writeToFile
("574 - 3001-OD2: " +
calculateDistance
(x574, y574, z574, x1_OD2, y1_OD2, z1_OD2));
FileWriter.writeToFile
("574 - 3002-OE1: " +
calculateDistance
(x574, y574, z574, x2_OE1, y2_OE1, z2_OE1));
FileWriter.writeToFile
("574 - 3002-OE2: " +
calculateDistance
(x574, y574, z574, x2_OE2, y2_OE2, z2_OE2));
FileWriter.writeToFile
("574 - 3003-OD1: " +
calculateDistance
(x574, y574, z574, x3_OD1, y3_OD1, z3_OD1));
```

```
FileWriter.writeToFile
("574 - 3003-OD2: " +
calculateDistance
(x574, y574, z574, x3_OD2, y3_OD2, z3_OD2));
FileWriter.writeToFile
("574 - 3004-OE1: " +
calculateDistanc
(x574, y574, z574, x4_OE1, y4_OE1, z4_OE1));
FileWriter.writeToFile
("574 - 3004-OE2: " +
calculateDistance
(x574, y574, z574, x4_OE2, y4_OE2, z4_OE2));
*/

FileWriter.writeToFile
("636 - 3001-OD1: " +
calculateDistance
(x636, y636, z636, x1_OD1, y1_OD1, z1_OD1));
FileWriter.writeToFile
("636 - 3001-OD2: " +
calculateDistance
(x636, y636, z636, x1_OD2, y1_OD2, z1_OD2));
FileWriter.writeToFile
("636 - 3002-OE1: " +
calculateDistance
(x636, y636, z636, x2_OE1, y2_OE1, z2_OE1));
FileWriter.writeToFile
("636 - 3002-OE2: " +
calculateDistance
(x636, y636, z636, x2_OE2, y2_OE2, z2_OE2));
FileWriter.writeToFile
("636 - 3003-OD1: " +
calculateDistance
(x636, y636, z636, x3_OD1, y3_OD1, z3_OD1));
FileWriter.writeToFile
("636 - 3003-OD2: " +
calculateDistance
(x636, y636, z636, x3_OD2, y3_OD2, z3_OD2));
FileWriter.writeToFile
("636 - 3004-OE1: " +
calculateDistance
(x636, y636, z636, x4_OE1, y4_OE1, z4_OE1));
FileWriter.writeToFile
("636 - 3004-OE2: " +
```

```
calculateDistance
(x636, y636, z636, x4_OE2, y4_OE2, z4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("637 - 3001-OD1: " +
calculateDistance
(x637, y637, z637, x1_OD1, y1_OD1, z1_OD1));
FileWriter.writeToFile
("637 - 3001-OD2: " +
calculateDistance
(x637, y637, z637, x1_OD2, y1_OD2, z1_OD2));
FileWriter.writeToFile
("637 - 3002-OE1: " +
calculateDistance
(x637, y637, z637, x2_OE1, y2_OE1, z2_OE1));
FileWriter.writeToFile
("637 - 3002-OE2: " +
calculateDistance
(x637, y637, z637, x2_OE2, y2_OE2, z2_OE2));
FileWriter.writeToFile
("637 - 3003-OD1: " +
calculateDistance
(x637, y637, z637, x3_OD1, y3_OD1, z3_OD1));
FileWriter.writeToFile
("637 - 3003-OD2: " +
calculateDistance
(x637, y637, z637, x3_OD2, y3_OD2, z3_OD2));
FileWriter.writeToFile
("637 - 3004-OE1: " +
calculateDistance
(x637, y637, z637, x4_OE1, y4_OE1, z4_OE1));
FileWriter.writeToFile
("637 - 3004-OE2: " +
calculateDistance
(x637, y637, z637, x4_OE2, y4_OE2, z4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("640 - 3001-OD1: " +
calculateDistance
(x640, y640, z640, x1_OD1, y1_OD1, z1_OD1));
FileWriter.writeToFile
("640 - 3001-OD2: " +
calculateDistance
(x640, y640, z640, x1_OD2, y1_OD2, z1_OD2));
```

```
FileWriter.writeToFile("640 - 3002-OE1: " +
calculateDistance(
x640, y640, z640, x2_OE1, y2_OE1, z2_OE1));
FileWriter.writeToFile
("640 - 3002-OE2: " +
calculateDistance
(x640, y640, z640, x2_OE2, y2_OE2, z2_OE2));
FileWriter.writeToFile
("640 - 3003-OD1: " +
calculateDistance
(x640, y640, z640, x3_OD1, y3_OD1, z3_OD1));
FileWriter.writeToFile
("640 - 3003-OD2: " +
calculateDistance
(x640, y640, z640, x3_OD2, y3_OD2, z3_OD2));
FileWriter.writeToFile
("640 - 3004-OE1: " +
calculateDistance
(x640, y640, z640, x4_OE1, y4_OE1, z4_OE1));
FileWriter.writeToFile("640 - 3004-OE2: " +
calculateDistance
(x640, y640, z640, x4_OE2, y4_OE2, z4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("641 - 3001-OD1: " +
calculateDistance
(x641, y641, z641, x1_OD1, y1_OD1, z1_OD1));
FileWriter.writeToFile
("641 - 3001-OD2: " +
calculateDistance
(x641, y641, z641, x1_OD2, y1_OD2, z1_OD2));
FileWriter.writeToFile
("641 - 3002-OE1: " +
calculateDistance
(x641, y641, z641, x2_OE1, y2_OE1, z2_OE1));
FileWriter.writeToFile
("641 - 3002-OE2: " +
calculateDistance
(x641, y641, z641, x2_OE2, y2_OE2, z2_OE2));
FileWriter.writeToFile
("641 - 3003-OD1: " +
calculateDistance
(x641, y641, z641, x3_OD1, y3_OD1, z3_OD1));
FileWriter.writeToFile("641 - 3003-OD2: " +
```

```
calculateDistance
(x641, y641, z641, x3_OD2, y3_OD2, z3_OD2));
FileWriter.writeToFile
("641 - 3004-OE1: " +
calculateDistance
(x641, y641, z641, x4_OE1, y4_OE1, z4_OE1));
FileWriter.writeToFile
("641 - 3004-OE2: " +
calculateDistance
(x641, y641, z641, x4_OE2, y4_OE2, z4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("642 - 3001-OD1: " +
calculateDistance
(x642, y642, z642, x1_OD1, y1_OD1, z1_OD1));
FileWriter.writeToFile
("642 - 3001-OD2: " +
calculateDistance
(x642, y642, z642, x1_OD2, y1_OD2, z1_OD2));
FileWriter.writeToFile
("642 - 3002-OE1: " +
calculateDistance
(x642, y642, z642, x2_OE1, y2_OE1, z2_OE1));
FileWriter.writeToFile
("642 - 3002-OE2: " +
calculateDistance
(x642, y642, z642, x2_OE2, y2_OE2, z2_OE2));
FileWriter.writeToFile
("642 - 3003-OD1: " +
calculateDistance
(x642, y642, z642, x3_OD1, y3_OD1, z3_OD1));
FileWriter.writeToFile
("642 - 3003-OD2: " +
calculateDistance
(x642, y642, z642, x3_OD2, y3_OD2, z3_OD2));
FileWriter.writeToFile
("642 - 3004-OE1: " +
calculateDistance
(x642, y642, z642, x4_OE1, y4_OE1, z4_OE1));
FileWriter.writeToFile
("642 - 3004-OE2: " +
calculateDistance
(x642, y642, z642, x4_OE2, y4_OE2, z4_OE2));
FileWriter.writeToFile
```

```
("");
FileWriter.writeToFile
("647 - 3001-OD1: " +
calculateDistance
(x647, y647, z647, x1_OD1, y1_OD1, z1_OD1));
FileWriter.writeToFile
("647 - 3001-OD2: " +
calculateDistance
(x647, y647, z647, x1_OD2, y1_OD2, z1_OD2));
FileWriter.writeToFile
("647 - 3002-OE1: " +
calculateDistance
(x647, y647, z647, x2_OE1, y2_OE1, z2_OE1));
FileWriter.writeToFile
("647 - 3002-OE2: " +
calculateDistance
(x647, y647, z647, x2_OE2, y2_OE2, z2_OE2));
FileWriter.writeToFile
("647 - 3003-OD1: " +
calculateDistance(
x647, y647, z647, x3_OD1, y3_OD1, z3_OD1));
FileWriter.writeToFile
("647 - 3003-OD2: " +
calculateDistance
(x647, y647, z647, x3_OD2, y3_OD2, z3_OD2));
FileWriter.writeToFile
("647 - 3004-OE1: " +
calculateDistance
(x647, y647, z647, x4_OE1, y4_OE1, z4_OE1));
FileWriter.writeToFile
("647 - 3004-OE2: " +
calculateDistance
(x647, y647, z647, x4_OE2, y4_OE2, z4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("567 - 1001-OD1: " +
calculateDistance
(x567, y567, z567, u1_OD1, v1_OD1, w1_OD1));
FileWriter.writeToFile
("567 - 1001-OD2: " +
calculateDistance
(x567, y567, z567, u1_OD2, v1_OD2, w1_OD2));
FileWriter.writeToFile
("567 - 1002-OE1: " +
```

```
calculateDistance
(x567, y567, z567, u2_OE1, v2_OE1, w2_OE1));
FileWriter.writeToFile
("567 - 1002-OE2: " +
calculateDistance
(x567, y567, z567, u2_OE2, v2_OE2, w2_OE2));
FileWriter.writeToFile
("567 - 1003-OD1: " +
calculateDistance
(x567, y567, z567, u3_OD1, v3_OD1, w3_OD1));
FileWriter.writeToFile
("567 - 1003-OD2: " +
calculateDistance
(x567, y567, z567, u3_OD2, v3_OD2, w3_OD2));
FileWriter.writeToFile
("567 - 1004-OE1: " +
calculateDistance
(x567, y567, z567, u4_OE1, v4_OE1, w4_OE1));
FileWriter.writeToFile("567 - 1004-OE2: " +
calculateDistance
(x567, y567, z567, u4_OE2, v4_OE2, w4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("569 - 1001-OD1: " +
calculateDistance(x569, y569, z569, u1_OD1, v1_OD1, w1_OD1));
FileWriter.writeToFile
("569 - 1001-OD2: " +
calculateDistance
(x569, y569, z569, u1_OD2, v1_OD2, w1_OD2));
FileWriter.writeToFile
("569 - 1002-OE1: " +
calculateDistance
(x569, y569, z569, u2_OE1, v2_OE1, w2_OE1));
FileWriter.writeToFile("569 - 1002-OE2: " +
calculateDistance
(x569, y569, z569, u2_OE2, v2_OE2, w2_OE2));
FileWriter.writeToFile("569 - 1003-OD1: " +
calculateDistance
(x569, y569, z569, u3_OD1, v3_OD1, w3_OD1));
FileWriter.writeToFile("569 - 1003-OD2: " +
calculateDistance
(x569, y569, z569, u3_OD2, v3_OD2, w3_OD2));
FileWriter.writeToFile("569 - 1004-OE1: " +
calculateDistance
```

```
(x569, y569, z569, u4_OE1, v4_OE1, w4_OE1));
FileWriter.writeToFile("569 - 1004-OE2: " +
calculateDistance
(x569, y569, z569, u4_OE2, v4_OE2, w4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("572 - 1001-OD1: " +
calculateDistance
(x572, y572, z572, u1_OD1, v1_OD1, w1_OD1));
FileWriter.writeToFile("572 - 1001-OD2: " +
calculateDistance
(x572, y572, z572, u1_OD2, v1_OD2, w1_OD2));
FileWriter.writeToFile("572 - 1002-OE1: " +
calculateDistance
(x572, y572, z572, u2_OE1, v2_OE1, w2_OE1));
FileWriter.writeToFile
("572 - 1002-OE2: " +
calculateDistance
(x572, y572, z572, u2_OE2, v2_OE2, w2_OE2));
FileWriter.writeToFile
("572 - 1003-OD1: " +
calculateDistance
(x572, y572, z572, u3_OD1, v3_OD1, w3_OD1));
FileWriter.writeToFile
("572 - 1003-OD2: " +
calculateDistance
(x572, y572, z572, u3_OD2, v3_OD2, w3_OD2));
FileWriter.writeToFile
("572 - 1004-OE1: " +
calculateDistance
(x572, y572, z572, u4_OE1, v4_OE1, w4_OE1));
FileWriter.writeToFile
("572 - 1004-OE2: " +
calculateDistance
(x572, y572, z572, u4_OE2, v4_OE2, w4_OE2));
FileWriter.writeToFile("");
FileWriter.writeToFile
("574 - 1001-OD1: " +
calculateDistance
(x574, y574, z574, u1_OD1, v1_OD1, w1_OD1));
FileWriter.writeToFile
("574 - 1001-OD2: " +
calculateDistance
(x574, y574, z574, u1_OD2, v1_OD2, w1_OD2));
```

```
FileWriter.writeToFile
("574 - 1002-OE1: " +
calculateDistance
(x574, y574, z574, u2_OE1, v2_OE1, w2_OE1));
FileWriter.writeToFile
("574 - 1002-OE2: " +
calculateDistance
(x574, y574, z574, u2_OE2, v2_OE2, w2_OE2));
FileWriter.writeToFile
("574 - 1003-OD1: " +
calculateDistance
(x574, y574, z574, u3_OD1, v3_OD1, w3_OD1));
FileWriter.writeToFile
("574 - 1003-OD2: " +
calculateDistance
(x574, y574, z574, u3_OD2, v3_OD2, w3_OD2));
FileWriter.writeToFile
("574 - 1004-OE1: " +
calculateDistance
(x574, y574, z574, u4_OE1, v4_OE1, w4_OE1));
FileWriter.writeToFile
("574 - 1004-OE2: " +
calculateDistance
(x574, y574, z574, u4_OE2, v4_OE2, w4_OE2));

}//end private void calcuateDistancesActinMyosin()

private double calculateDistance(double x1, double y1,
double z1, double x2, double y2, double z2)
{
    //System.out.println((x1-x2)
     + ", " + (y1-y2) + ", " + (z1-z2));
    //System.out.println
    (Math.pow((x1-x2),2)
    +Math.pow((y1-y2),2)
    +Math.pow((z1-z2),2));
    //System.out.println
    (x1 + " " + x2 + " "
    + y1 + " " + y2 + " "
    + z1 + " " + z2);



    double x = Math.sqrt
```

```
    (Math.pow((x1-x2),2)+
    Math.pow((y1-y2),2)+
    Math.pow((z1-z2),2));
    x = Math.abs(x > 0 ? Math.floor(x * 10) / 10.0 :
    Math.ceil(x * 10) / 10.0);

    return x;
}//end private double calculateDistance()

private  final static String getDateTime()
{
    DateFormat df =
    new SimpleDateFormat
    ("yyyy-MM-dd_hh:mm:ss");
    df.setTimeZone
    (TimeZone.getTimeZone("PST"));
    return df.format(new Date());
}

//(Nz atoms of Lys 636, 637, 640, 641, 644
//Od1 & Od2 atoms of Asp 1, 3 or Oe1
//& Oe2 atoms of Glu 2, 4 (1st monomer)
//Nz atoms of Lys 567, 569, 572, 574)
//Od1 & Od2 atoms of Asp 1, 3 or Oe1
//& Oe2 atoms of Glu 2, 4 (2nd monomer)

private double x567 = 0, y567 = 0, z567 = 0;
private double x569 = 0, y569 = 0, z569 = 0;
private double x572 = 0, y572 = 0, z572 = 0;
private double x574 = 0, y574 = 0, z574 = 0;
private double x636 = 0, y636 = 0, z636 = 0;
private double x637 = 0, y637 = 0, z637 = 0;
private double x640 = 0, y640 = 0, z640 = 0;
private double x641 = 0, y641 = 0, z641 = 0;
private double x642 = 0, y642 = 0, z642 = 0;
private double x644 = 0, y644 = 0, z644 = 0;

//ATOM   4941  OE1 GLN A 647
// 50.159  37.334  38.493  1.00100.00
private double x647 = 0, y647 = 0, z647 = 0;

private double u1_OD1 = 0, v1_OD1 = 0, w1_OD1 = 0;
private double u1_OD2 = 0, v1_OD2 = 0, w1_OD2 = 0;
```

```
private double u2_OE1 = 0, v2_OE1 = 0, w2_OE1 = 0;
private double u2_OE2 = 0, v2_OE2 = 0, w2_OE2 = 0;

private double u3_OD1 = 0, v3_OD1 = 0, w3_OD1 = 0;
private double u3_OD2 = 0, v3_OD2 = 0, w3_OD2 = 0;

private double u4_OE1 = 0, v4_OE1 = 0, w4_OE1 = 0;
private double u4_OE2 = 0, v4_OE2 = 0, w4_OE2 = 0;

private double x1_OD1 = 0, y1_OD1 = 0, z1_OD1 = 0;
private double x1_OD2 = 0, y1_OD2 = 0, z1_OD2 = 0;

private double x2_OE1 = 0, y2_OE1 = 0, z2_OE1 = 0;
private double x2_OE2 = 0, y2_OE2 = 0, z2_OE2 = 0;

private double x3_OD1 = 0, y3_OD1 = 0, z3_OD1 = 0;
private double x3_OD2 = 0, y3_OD2 = 0, z3_OD2 = 0;

private double x4_OE1 = 0, y4_OE1 = 0, z4_OE1 = 0;
private double x4_OE2 = 0, y4_OE2 = 0, z4_OE2 = 0;

private double a1_OD1 = 0, b1_OD1 = 0, c1_OD1 = 0;
private double a1_OD2 = 0, b1_OD2 = 0, c1_OD2 = 0;

private double a2_OE1 = 0, b2_OE1 = 0, c2_OE1 = 0;
private double a2_OE2 = 0, b2_OE2 = 0, c2_OE2 = 0;

private double a3_OD1 = 0, b3_OD1 = 0, c3_OD1 = 0;
private double a3_OD2 = 0, b3_OD2 = 0, c3_OD2 = 0;

private double a4_OE1 = 0, b4_OE1 = 0, c4_OE1 = 0;
private double a4_OE2 = 0, b4_OE2 = 0, c4_OE2 = 0;


private OutputFileWriter FileWriter =
new OutputFileWriter("/Users/..."
+ getDateTime() + ".txt");

}//end class ReadFromFile
```

```
public class Calculations
{
    public void veryoldrotatePoints()
    {
        rotatedX = x;
        int ix = (int)(rotatedX * 100.0); // scale it
        double roundedValue = ((double)ix)/100.0;
       rotatedX = roundedValue;

        rotatedY = ((b*((u*u)*(w*w)))+(v*((-a*u)+
        (-c*w)+(u*x)+(v*y)+(w*z)))+((-b*((u*u)*(w*w)))+
        v*((a*u)+(c*w)+(-u*x)+(-w*z))+(y*((u*u)*
        (w*w)))*Math.cos(theta))+
        ((Math.sqrt((u*u)*(v*v)*(w*w)))*((c*u)+(-a*w)+(w*x)
        +(-u*z))*Math.sin(theta)))/((u*u)+(v*v)+(w*w));
        ix = (int)(rotatedY * 100.0); // scale it
        roundedValue = ((double)ix)/100.0;
        rotatedY = roundedValue;

        rotatedZ = ((c*((u*u)*(v*v)))+(w*((-a*u)+
        (-b*v)+(u*x)+(v*y)+(w*z)))+((-c*((u*u)*(v*v)))
        +w*((a*u)+(b*v)+(-u*x)+(-v*y))+(z*((u*u)*(v*v)))
        *Math.cos(theta))+((Math.sqrt((u*u)*(v*v)*(w*w)))
        *((-b*u)+(a*v)+(-v*x)+(u*y))*Math.sin(theta)))
        /((u*u)+(v*v)+(w*w));
        ix = (int)(rotatedZ * 100.0); // scale it
        roundedValue = ((double)ix)/100.0;
        rotatedZ = roundedValue;
    }//end public void veryoldrotatePodouble()

    public void oldrotatePoints()
    {
        rotatedX = .392159822527373469*x;
        int ix = (int)(rotatedX * 1000.0); // scale it
        double roundedValue = ((double)ix)/1000.0;
       rotatedX = roundedValue;

        rotatedY = .454555453482173077*
        y-.184481317697601677*z;
        ix = (int)(rotatedY * 1000.0); // scale it
        roundedValue = ((double)ix)/1000.0;
        rotatedY = roundedValue;

        rotatedZ = -.184481317697601677*
```

```
    y+.937604369517422653*z;
    ix = (int)(rotatedZ * 1000.0); // scale it
    roundedValue = ((double)ix)/1000.0;
    rotatedZ = roundedValue;
}//end public void oldrotatePodouble()

public void calculatedrotatePoints()
{
    rotatedX =0.242563108597366739e-1*
    x-.370994399162125676
    *y-.124757353474171925*z;
    int ix = (int)(rotatedX * 1000.0); // scale it
    double roundedValue = ((double)ix)/1000.0;
   rotatedX = roundedValue;

    rotatedY = -.370556690005765299*
    x-.124417167015791488*
    y+.296867627901004127*z;
    ix = (int)(rotatedY * 1000.0); // scale it
    roundedValue = ((double)ix)/1000.0;
    rotatedY = roundedValue;

    rotatedZ = -.126051501171216718*
    x+.295414178922473569*
    y-.899837378515070729*z;
    ix = (int)(rotatedZ * 1000.0); // scale it
    roundedValue = ((double)ix)/1000.0;
    rotatedZ = roundedValue;
}//end public void rotatePodouble()

public void rotatePoints()
{
    rotatedX = (w*w);
    int ix = (int)(rotatedX * 100.0); // scale it
    double roundedValue = ((double)ix)/100.0;
   rotatedX = roundedValue;

    rotatedY = ((b*((u*u)*(w*w)))+(v*((-a*u)+
    (-c*w)+(u*x)+(v*y)+(w*z)))+((-b*((u*u)*(w*w)))
    +v*((a*u)+(c*w)+(-u*x)+(-w*z))+(y*((u*u)
    *(w*w)))*
    Math.cos(theta))+((Math.sqrt((u*u)*(v*v)*
    (w*w)))*((c*u)+(-a*w)+(w*x)+(-u*z))*
    Math.sin(theta)))
```

```
    /((u*u)+(v*v)+(w*w));
    ix = (int)(rotatedY * 100.0); // scale it
    roundedValue = ((double)ix)/100.0;
    rotatedY = roundedValue;

    rotatedZ = ((c*((u*u)*(v*v)))+(w*((-a*u)+(-b*v)+
    (u*x)+(v*y)+(w*z)))+((-c*((u*u)*(v*v)))+w*((a*u)
    +(b*v)+(-u*x)+(-v*y))+(z*((u*u)*(v*v)))
    *Math.cos(theta))+((Math.sqrt((u*u)*(v*v)*
    (w*w)))*((-b*u)+(a*v)+(-v*x)+(u*y))*
    Math.sin(theta)))/((u*u)+(v*v)+(w*w));
    ix = (int)(rotatedZ * 100.0); // scale it
    roundedValue = ((double)ix)/100.0;
    rotatedZ = roundedValue;
}//end public void veryoldrotatePodouble()

public void setPoints(double ax,double ay,double az,
double bx, double by,double bz, double angle)
{
    a = ax;
    b = ay;
    c = az;
    d = bx;
    e = by;
    f = bz;
    theta = angle;
    u = d-a;
    v = e-b;
    w = f-c;

}//end public void setPoints(double ax,double ay,double az,
double bx, double by,double bz)

public void setXYZ(double a, double b, double c)
{
    x = a;
    y = b;
    z = c;
}//end public void setXYZ(double a, double b, double c)

public double getRotatedX()
{
    return rotatedX;
}//end public double getRotatedX()
```

```
    public double getRotatedY()
    {
        return rotatedY;
    }//end public double getRotatedY()

    public double getRotatedZ()
    {
        return rotatedZ;
    }//end public double getRotatedZ()

    //points
    //pointe to be rotated
    double x = 0;
    double y = 0;
    double z = 0;
    //first contact on Myosin
    double a = 0;
    double b = 0;
    double c = 0;
    //Secodn contact on Myosin
    double d = 0;
    double e = 0;
    double f = 0;

    //vector of rotation
    double u = 0;
    double v = 0;
    double w = 0;

    double theta = 0;

    double rotatedX = 0;
    double rotatedY = 0;
    double rotatedZ = 0;
}
```

```
public class Npairs
{
    public void calculateNpairs(Protein actin, Protein myosin, String inputFilena
    {
     fillArrays();
        OutputFileWriter FileWriter = new OutputFileWriter("/Users/.."
        + getDateTime() + ".txt");
        OutputFileWriter FileWriterPolarPairs = new OutputFileWriter("/Users/..."
        + getDateTime() + ".txt");
        OutputFileWriter FileWriterNpairsExcel = new  OutputFileWriter("/Users/..
        + getDateTime() + ".txt");


        for(int k = 0; k < lenthMyosin; k++)
        {
            for(int c = 0; c < lenthActin; c++)
            {
                String actinType = actin.getAtomType(c);
                String actinResName = actin.getAtomName(c);
                int actinResNum = actin.getResNum(c);
                double actinX = actin.getXcoordinate(c);
                double actinY = actin.getYcoordinate(c);
                double actinZ = actin.getZcoordinate(c);

                String myosinType = myosin.getAtomType(k);
                String myosinResName = myosin.getAtomName(k);
                int myosinResNum = myosin.getResNum(k);
                double myosinX = myosin.getXcoordinate(k);
                double myosinY = myosin.getYcoordinate(k);
                double myosinZ = myosin.getZcoordinate(k);

                double temp = calculateDistance(myosinX, myosinY,
                myosinZ, actinX, actinY, actinZ);

                //Npairs
                if(temp <= NpairsDistance)
                {
                    if(actinType == null)
                    {
                        System.out.println("c = " + c + " k = "
                        + k + " actin is null");
                    }//end if(actinType ==null)
                    else if(myosinType == null)
                    {
```

```
            System.out.println("c = " + c + " k = "
            + k + " myosin is null");
        }//end else if(myosinType ==null)
        else
        {
          for(int i = 0; i < 84; i++)
 {
  if(actinResNum ==
  actinResiduesExcludeE[i])
  actinTest = false;
  if(actinResNum ==
  actinResiduesExcludeF[i])
  actinTest = false;
  if(actinResNum ==
  actinResiduesExcludeG[i])
  actinTest = false;
  if(actinResNum ==
  actinResiduesExcludeH[i])
  actinTest = false;
  if(actinResNum ==
  actinResiduesExcludeI[i])
  actinTest = false;
 }//end for(int i = 0; i < 84; i++)

 for(int i = 0; i < 84; i++)
 {
  if(myosinResNum ==
  myosinResiduesExcludeH[i])
  myosinTest = false;
 }//end for(int i = 0; i < 84; i++)

if(myosinTest == true &&
actinTest == true)
{
if((actinType.equals("N") == true) && (myosinType.startsWith("O") ==
         {
if(temp < 2.8)
{
numNpairsLessThen++;
FileWriterNpairsExcel.
writeToFile(myosinResName
+ ", " +  myosin.getResNum(k) + ", "
+ myosinType);
             FileWriterNpairsExcel.
```

```
                writeToFile(actinResName + ", "
                 +  actin.getResNum(c) + ", " + actinType);
                FileWriterNpairsExcel.writeToFile("");
}//end if(NpairsDistance < 2.8)
                numNpairs++;
                FileWriter.writeToFile("Npairs#: " + numNpairs);
                FileWriter.writeToFile("Pair: " + actinType +
                 " " + myosinType);
                FileWriter.writeToFile("Atom   "
                + myosin.getAtomNum(k) + " " + myosinResName
                 + " " +  myosin.getResNum(k) + " " + myosinX
                  + " " + myosinY + " " + myosinZ + " "
                   + myosinType);
                FileWriter.writeToFile("Atom   "
                + actin.getAtomNum(c) + " " + actinResName
                + " " +  actin.getResNum(c) + " " + actinX
                 + " " + actinY + " " + actinZ + " "
                 + actinType);
                FileWriter.writeToFile("distance: "
                + Double.toString(temp));
                FileWriter.writeToFile(" ");


            }//end if((actinType.startsWith("N") == true)
            && (myosinType.startsWith("O") == true))
            if((actinType.equals("NZ") == true)
            && (myosinType.startsWith("O") == true))
            {
            if(temp < 2.8)
{
numNpairsLessThen++;
FileWriterNpairsExcel.
writeToFile(myosinResName
+ ", " +  myosin.getResNum(k)
+ ", " + myosinType);
                FileWriterNpairsExcel.
                writeToFile(actinResName
                + ", " +  actin.getResNum(c) + ", "
                 + actinType);
                FileWriterNpairsExcel.writeToFile("");
}//end if(NpairsDistance < 2.8)
                numNpairs++;
                FileWriter.writeToFile("Npairs#: " + numNpairs);
                FileWriter.writeToFile("Pair: " + actinType
```

```
                + " " + myosinType);
                FileWriter.writeToFile("Atom   "
                + myosin.getAtomNum(k) + " " + myosinResName
                + " " +  myosin.getResNum(k) + " " +
                myosinX + " " + myosinY + " " + myosinZ
                + " " + myosinType);
                FileWriter.writeToFile("Atom   "
                 + actin.getAtomNum(c) + " "
                 + actinResName + " " +  actin.getResNum(c)
                 + " " + actinX + " " + actinY + " " + actinZ
                  + " " + actinType);
                FileWriter.writeToFile("distance: "
                + Double.toString(temp));
                FileWriter.writeToFile(" ");
            }//end if((actinType.startsWith("NZ") == true)
            && (myosinType.startsWith("O") == true))
            if((actinType.startsWith("O") == true)
            && (myosinType.equals("N") == true))
            {
            if(temp < 2.8)
{
numNpairsLessThen++;
FileWriterNpairsExcel.
writeToFile(myosinResName + ", "
+  myosin.getResNum(k) + ", " + myosinType);
                FileWriterNpairsExcel.
                writeToFile(actinResName + ", "
                +  actin.getResNum(c) + ", "
                + actinType);
                FileWriterNpairsExcel.writeToFile("");
}//end if(NpairsDistance < 2.8
            numNpairs++;
            FileWriter.writeToFile("Npairs#: " + numNpairs);
            FileWriter.writeToFile("Pair: " + actinType
             + " " + myosinType);
            FileWriter.writeToFile("Atom   "
            + myosin.getAtomNum(k) + " " + myosinResName
             + " " +  myosin.getResNum(k) + " "
              + myosinX + " " + myosinY + " " +
              myosinZ + " " + myosinType);
            FileWriter.writeToFile("Atom   "
            + actin.getAtomNum(c) + " " + actinResName
             + " " +  actin.getResNum(c) + " " +
              actinX + " " + actinY + " " +
```

```
            actinZ + " " + actinType);
          FileWriter.writeToFile("distance: "
          + Double.toString(temp));
              FileWriter.writeToFile(" ");
          }//end if((actinType.startsWith("O") == true)
          && (myosinType.startsWith("N") == true))
          if((actinType.startsWith("O") == true)
          && (myosinType.equals("NZ") == true))
          {
          if(temp < 2.8)
{
numNpairsLessThen++;
FileWriterNpairsExcel.
writeToFile(myosinResName + ", "
+  myosin.getResNum(k) + ", "
+ myosinType);
              FileWriterNpairsExcel.
              writeToFile(actinResName + ", "
              +  actin.getResNum(c) + ", "
              + actinType);
              FileWriterNpairsExcel.writeToFile("");
}//end if(NpairsDistance < 2.8
          numNpairs++;
          FileWriter.writeToFile("Npairs#: " + numNpairs);
          FileWriter.writeToFile("Pair: " + actinType
          + " " + myosinType);
          FileWriter.writeToFile("Atom   "
          + myosin.getAtomNum(k) + " " + myosinResName
          + " " +  myosin.getResNum(k) + " " + myosinX
          + " " + myosinY + " " + myosinZ + " "
           + myosinType);
          FileWriter.writeToFile("Atom   "
          + actin.getAtomNum(c) + " " +
          actinResName + " " +  actin.getResNum(c)
           + " " + actinX + " " + actinY + " "
           + actinZ + " " + actinType);
          FileWriter.writeToFile("distance: "
           + Double.toString(temp));
              FileWriter.writeToFile(" ");
          }//end if((actinType.startsWith("O") == true)
           && (myosinType.startsWith("NZ") == true))
}//end if(myosinTest == true &&
actinTest == true)
```

```
    }//end else
}//end if(NpairsDistance)

//System.out.println("Actin Residue is: "
+ actinResName + actin.getResNum(c));

actinTest = true;
myosinTest = true;

if(actinResName == null)
{
 actinResName="";
}

if(myosinResName == null)
{
 myosinResName="";
}


 //Polar Pairs
if(actinResName.equals("ARG") ||
actinResName.equals("ASN") ||
actinResName.equals("ASP") ||
actinResName.equals("GLU") ||
actinResName.equals("GLN") ||
actinResName.equals("HIS") ||
actinResName.equals("LYS") ||
actinResName.equals("SER") ||
actinResName.equals("THR") ||
actinResName.equals("TYR"))
 {

  if(myosinResName.equals("ARG") ||
  myosinResName.equals("ASN") ||
  myosinResName.equals("ASP") ||
  myosinResName.equals("GLU") ||
  myosinResName.equals("GLN") ||
  myosinResName.equals("HIS") ||
  myosinResName.equals("LYS") ||
  myosinResName.equals("SER") ||
  myosinResName.equals("THR") ||
  myosinResName.equals("TYR"))
```

```
   {
//System.out.println(myosinResName + " "
+ actinResName);
 if(temp <= polarPairsDistance)
 {
 for(int i = 0; i < 84; i++)
 {
  if(actinResNum ==
  actinResiduesExcludeE[i])
  actinTest = false;
  if(actinResNum ==
  actinResiduesExcludeF[i])
  actinTest = false;
  if(actinResNum ==
  actinResiduesExcludeG[i])
  actinTest = false;
  if(actinResNum ==
  actinResiduesExcludeH[i])
  actinTest = false;
  if(actinResNum ==
  actinResiduesExcludeI[i])
  actinTest = false;
 }//end for(int i = 0; i < 84; i++)

 for(int i = 0; i < 84; i++)
 {
  if(myosinResNum ==
   myosinResiduesExcludeH[i])
  myosinTest = false;
 }//end for(int i = 0; i < 84; i++)

 if(myosinTest == true &&
 actinTest == true)
 {
 if((actinType.startsWith("N")
 == true) &&
 (myosinType.startsWith("N") == true)
     ||
    (actinType.startsWith("N") == true)
    && (myosinType.startsWith("O") == true)
    || (actinType.startsWith("O") == true)
     && (myosinType.startsWith("O") == true)
    || (actinType.startsWith("O") == true)
    && (myosinType.startsWith("N") == true))
```

```
                {
                numPolarPairs++;
                        FileWriterPolarPairs.
                        writeToFile("PolarPairs#: "
                         + numPolarPairs);
                            //FileWriterPolarPairs.
                            writeToFile("Pair: " + actinType
                            + " " + myosinType);
                        FileWriterPolarPairs.
                        writeToFile("Atom   "
                        + myosin.getAtomNum(k) + " "
                         + myosinResName + " "
                         +  myosin.getResNum(k)
                         + " " + myosinX + " " +
                         myosinY + " " + myosinZ
                         + " " + myosinType);
                        FileWriterPolarPairs.
                        writeToFile("Atom   " +
                        actin.getAtomNum(c) + " " +
                         actinResName + " " +
                         actin.getResNum(c) + " " +
                          actinX + " " + actinY +
                           " " + actinZ + " " +
                           actinType);
                        FileWriterPolarPairs.
                        writeToFile("distance:
                        " + Double.toString(temp));
                        FileWriterPolarPairs.
                        writeToFile(" ");
                    }//end if((actinType.startsWith("N")...
                }//end if(myosinTest == true ...
                   }//end if(temp <= polarPairsDistance)
                  }//end if(myosinResName.equals
              }//end if(actinResName.equals

        }//end for(int c =0; c < lenthActin; c++)
    }//end for(int k = 0; k < lenthMyosin; k++)

    FileWriter.writeToFile(inputFilenameMyosin);
    FileWriter.writeToFile(inputFilenameActin);
    FileWriter.writeToFile
    ("Total # of Npairs < 3.8 = " + numNpairs);
    FileWriter.writeToFile
    ("Total # of Npairs < 2.8 = " + numNpairsLessThen);
```

```
    FileWriterPolarPairs.
    writeToFile(inputFilenameMyosin);
    FileWriterPolarPairs.
    writeToFile(inputFilenameActin);
    FileWriterPolarPairs.
    writeToFile
    ("Total # of PolarPairs = " + numPolarPairs);
}//end public void calculateClashes()

private static double calculateDistance
(double x1, double y1, double z1,
double x2, double y2,  double z2)
{
    double x = Math.sqrt(Math.pow((x1-x2),2)
    + Math.pow((y1-y2),2)+ Math.pow((z1-z2),2));

     x = Math.abs(x); // > 0 ? Math.floor(x * 10)
     / 10.0 : Math.ceil(x * 10) / 10.0);

    return x;
}//end private static double calculateDistance
(double x1, double x2, double
y1, double y2, double z1, double z2)

private final static String getDateTime()
{
    DateFormat df = new SimpleDateFormat
    ("yyyy-MM-dd_hh:mm:ss");
    df.setTimeZone(TimeZone.getTimeZone
    ("PST"));
    return df.format(new Date());
}

private int numNpairs = 0;
private int numNpairsLessThen = 0;
private int numPolarPairs = 0;
private double polarPairsDistance = 3.8;
private double NpairsDistance = 3.8;
private int lenthMyosin = 6722;
private int lenthActin = 14648;
boolean actinTest = true;
boolean myosinTest = true;
```

```
    private int[] actinResiduesExcludeE = new int[84];
    private int[] actinResiduesExcludeF = new int[84];
    private int[] actinResiduesExcludeG = new int[84];
    private int[] actinResiduesExcludeH = new int[84];
    private int[] actinResiduesExcludeI = new int[84];
    private int [] myosinResiduesExcludeH = new int[84];


}//end public class Npairs
```

```
public static void calculateASA
(Protein MYOstructure, Protein ACTstructure,
Protein myosinStructure,
Protein actinStructure)
{
OutputFileWriter FileWriter = new OutputFileWriter("/Users/..."
 + getDateTime() + ".txt");
    FileWriter.writeToFile
    ("REMARK File Names Before:");
    FileWriter.writeToFile
    ("REMARK " + inputFilename02);
    FileWriter.writeToFile
    ("REMARK " + inputFilename03);
    FileWriter.writeToFile
    ("REMARK File Names After:");
    FileWriter.writeToFile
    ("REMARK " + inputFilename00);
    FileWriter.writeToFile
    ("REMARK " + inputFilename01);
    FileWriter.writeToFile
    ("Residue" + "\t\t" + "Number"
    + "\t\t" + "Before" + "\t\t" + "After"
    + "\t\t\t" + "ASA" + "\t\t\t" + "%");
    FileWriter.writeToFile("--------------
    ------------------------------------
    ------------------------------------");

    //myosin
    double beforeASA = 0;
    double afterASA = 0;
    double roundedValueBeforeASA = 0;
    double roundedValueAfterASA = 0;
    double deltaASA = 0;
    double roundedValueDeltaASA = 0;
    double finalASA = 0;
    double totalASAbefore = 0;
    double totalASAafter = 0;

    int counter = 1;
    int size = myosinSize;

    for(int c = 0; c < size;c++)
    {
     String atomType =
```

```
myosinStructure.getAtomType(c);
String atomTypeMYO =
MYOstructure.getAtomType(c);
int resNum =
myosinStructure.getResNum(c);
String resName =
myosinStructure.getAtomName(c);
double myosinStructureASA =
myosinStructure.getASA(c);
double MYOstructureASA =
MYOstructure.getASA(c);

if(resNum == counter)
{
totalASAbefore += myosinStructureASA;
totalASAafter += MYOstructureASA;
// ALA, VAL, LEU, PHE, ILE, PRO, TRP, MET
if(resName.equals("ALA")
|| resName.equals("VAL")
|| resName.equals("LEU")
|| resName.equals("PHE")
|| resName.equals("ILE")
|| resName.equals("PRO")
|| resName.equals("TRP")
|| resName.equals("MET"))
{
//System.out.println
(resName + " " + beforeASA);
if(atomType.startsWith("C"))
{
beforeASA += myosinStructureASA;
    }//end if(resNum == counter)
if(atomTypeMYO.startsWith("C"))
{
    afterASA += MYOstructureASA;
}//end if(atomType.startsWith("C"))
}//endif(resName.equals(...
   }
   else
   {
int r = (int)(beforeASA * 1000);
roundedValueBeforeASA = r / 1000.0;

r = (int)(afterASA * 1000);
```

```
 roundedValueAfterASA = r / 1000.0;

 deltaASA = beforeASA - afterASA;
 r = (int)(deltaASA * 1000);
 roundedValueDeltaASA = r / 1000.0;

 finalASA = 100 -
 ((roundedValueAfterASA/
 roundedValueBeforeASA)*100);
 c--;
 if(resName != null)
 {
 if(resName.equals("ALA")
 || resName.equals("VAL")
 || resName.equals("LEU")
 || resName.equals("PHE")
 || resName.equals("ILE")
 || resName.equals("PRO")
 || resName.equals("TRP")
 || resName.equals("MET"))
 {
 if(finalASA >= 5)
 {
 FileWriter.
 writeToFile
 (myosinStructure.getAtomName(c)
 + "\t\t"
 + myosinStructure.getResNum(c)
  + "\t\t\t\t" +
  roundedValueBeforeASA
   + "\t\t" +
   roundedValueAfterASA
   + "\t\t" +
   roundedValueDeltaASA
   + "\t\t" +
   finalASA);
 }//end if(roundedValueDeltaASA
 }//end if(resName.equals("ALA")

 }//end if null

 beforeASA = 0;
afterASA = 0;
totalASAbefore = 0;
```

```
totalASAafter = 0;
 counter++;
 }//end if(resNum == counter)
}//end for(int c = 0; c < size;c++)

FileWriter.writeToFile("");

//actin
beforeASA = 0;
afterASA = 0;
roundedValueBeforeASA = 0;
roundedValueAfterASA = 0;
deltaASA = 0;
roundedValueDeltaASA = 0;
finalASA = 0;
totalASAbefore = 0;
totalASAafter = 0;
counter = 1001;
size = actinSize;

for(int c = 0; c < size;c++)
{
 String atomType =
 actinStructure.getAtomType(c);
 String atomTypeACT =
  ACTstructure.getAtomType(c);
 int resNum =
 actinStructure.getResNum(c);
 String resName =
 actinStructure.getAtomName(c);
 double actinStructureASA =
 actinStructure.getASA(c);
 double ACTstructureASA =
 ACTstructure.getASA(c);
 //System.out.println(resName);
 if(resNum == counter)
 {
 totalASAbefore += actinStructureASA;
 totalASAafter += ACTstructureASA;
 // ALA, VAL, LEU, PHE, ILE, PRO, TRP, MET
 if(resName.equals("ALA")
 || resName.equals("VAL")
  || resName.equals("LEU")
  || resName.equals("PHE")
```

```
|| resName.equals("ILE")
|| resName.equals("PRO")
|| resName.equals("TRP")
|| resName.equals("MET"))
{
//System.out.println(resName);
if(atomType.startsWith("C"))
    {
    beforeASA += actinStructureASA;
    }//end if(atomType.startsWith("C"))
if(atomTypeACT.startsWith("C"))
{
    afterASA += ACTstructureASA;
    }//end if(atomTypeACT.startsWith("C"))
}//end if(resName.equals....

   }
   else
   {
int r = (int)(beforeASA * 1000);
roundedValueBeforeASA = r / 1000.0;

r = (int)(afterASA * 1000);
roundedValueAfterASA = r / 1000.0;

deltaASA = beforeASA - afterASA;
r = (int)(deltaASA * 1000);
roundedValueDeltaASA = r / 1000.0;

finalASA = 100 - ((roundedValueAfterASA/
roundedValueBeforeASA)*100);
c--;
if(resName != null)
{
if(resName.equals("ALA")
 || resName.equals("VAL")
 || resName.equals("LEU")
 || resName.equals("PHE")
 || resName.equals("ILE")
 || resName.equals("PRO")
 || resName.equals("TRP")
 || resName.equals("MET"))
{
if(finalASA >= 5)
```

```
     {
     FileWriter.
     writeToFile(actinStructure.getAtomName(c)
      + "\t\t" + actinStructure.getResNum(c)
       + "\t\t\t" + roundedValueBeforeASA
        + "\t\t" + roundedValueAfterASA
        + "\t\t" + roundedValueDeltaASA
         + "\t\t" + finalASA);
     }//end if(roundedValueDeltaASA
        }//end if(resName.equals("ALA")
     }//end if null


     beforeASA = 0;
    afterASA = 0;
    totalASAbefore = 0;
    totalASAafter = 0;
     counter++;
     }//end if(resNum == counter)


     if(counter == 1376)
        {
        counter = 2001;
        }
        else if(counter == 2376)
        {
        counter = 3001;
        }
        else if(counter == 3376)
        {
        counter = 4001;
        }
        else if(counter == 4376)
        {
        counter = 5001;
        }
    }//end for(int c = 0; c < size;c++)
}//end public static void calculateASA...

public static void calculateRa()
{
double ala = 70.79;
double ile = 165.58*60;
```

```
double leu = 154.24*60;
double met = 167.87*60;
double phe = 180.97*60;
double pro = 130.76*60;
double trp = 223.23*60;
double val = 134.68*60;

Protein proteinStructureBefore =
 new Protein(numOfInterfaceResidues);
Protein proteinStructureAfter =
 new Protein(numOfInterfaceResidues);

ReadFromFile interfaceFiles =
 new ReadFromFile();

interfaceFiles.readFile
(proteinStructureBefore,
numOfInterfaceResidues,
interfaceResidueBefore);
interfaceFiles.readFile
(proteinStructureAfter,
numOfInterfaceResidues, interfaceResidueAfter);

OutputFileWriter FileWriter =
new OutputFileWriter("/Users/..."
+ getDateTime() + ".txt");
    FileWriter.writeToFile("REMARK File Names Before:");
    FileWriter.writeToFile("REMARK " +
    interfaceResidueBefore);
FileWriter.writeToFile("REMARK " +
interfaceResidueAfter);
FileWriter.writeToFile("Residue" +
"\t\t" + "Number" + "\t\t" + "Before"
 + "\t\t" + "After" + "\t\t\t" + "deltaAT"
  + "\t\t\t" + "deltaRA");
FileWriter.writeToFile("---------
----------------------");
FileWriter.writeToFile("");

    double beforeSCASA = 0;
    double afterSCASA = 0;
    double roundedValueBeforeSCASA = 0;
    double roundedValueAfterSCASA = 0;
    double deltaSCASA = 0;
```

```
    double roundedValueDeltaSCASA = 0;
double Ra = 0;

    for(int c = 0;
     c < (numOfInterfaceResidues-1);c++)
    {
     //String atomTypeBefore =
     proteinStructureBefore.getAtomType(c);
     //String atomTypeAfter =
      proteinStructureAfter.getAtomType(c);
     int resNumCurrent =
     proteinStructureBefore.getResNum(c);
     int resNumNext =
     proteinStructureBefore.getResNum(c+1);
     String resName =
      proteinStructureBefore.getAtomName(c);
     double asaBefore =
      proteinStructureBefore.getASA(c);
     double asaAfter =
     proteinStructureAfter.getASA(c);




     if(resNumCurrent == resNumNext)
     {
     beforeSCASA += asaBefore;
     afterSCASA += asaAfter;
        }
        else
        {
beforeSCASA += asaBefore;
     afterSCASA += asaAfter;

     int r = (int)(beforeSCASA
     * 1000);
     roundedValueBeforeSCASA =
     r / 1000.0;

     r = (int)(afterSCASA *
      1000);
     roundedValueAfterSCASA =
      r / 1000.0;
```

```
deltaSCASA = beforeSCASA
- afterSCASA ;
r = (int)(deltaSCASA *
 1000);
roundedValueDeltaSCASA =
r / 1000.0;

if(resName.equals("ALA"))
{
Ra =
roundedValueDeltaSCASA*100/ala;
}
else if(resName.equals("ILE"))
{
Ra =
roundedValueDeltaSCASA*100/ile;
}
else if(resName.equals("LEU"))
{
Ra =
roundedValueDeltaSCASA*100/leu;
}
else if(resName.equals("MET"))
{
Ra =
roundedValueDeltaSCASA*100/met;
}
else if(resName.equals("PHE"))
{
Ra =
roundedValueDeltaSCASA*100/phe;
}
else if(resName.equals("PRO"))
{
Ra =
roundedValueDeltaSCASA*100/pro;
}
else if(resName.equals("TRP"))
{
Ra =
roundedValueDeltaSCASA*100/trp;
}
else if(resName.equals("VAL"))
{
```

```
     Ra =
     roundedValueDeltaSCASA*100/val;
     }

    FileWriter.writeToFile
    (resName + "," +
    proteinStructureBefore.getResNum(c)
     + "," + roundedValueBeforeSCASA
     + "," + roundedValueAfterSCASA
     + "," + roundedValueDeltaSCASA
     + "," + Ra);

beforeSCASA = 0;
    afterSCASA = 0;
    }//end else


    }//end for(int c = 0; ...
}//end public static void calculateAT()

    private final static String getDateTime()
    {
        DateFormat df =
        new SimpleDateFormat
        ("yyyy-MM-dd_hh:mm:ss");
        df.setTimeZone
        (TimeZone.getTimeZone("PST"));
        return df.format(new Date());
    }
```

```
public static void findInterfaceResidues
(Protein myosinStructure,
Protein actinStructure)
{
OutputFileWriter FileWriter =
new OutputFileWriter("/Users/..."
 + getDateTime() + ".txt");
//OutputFileWriter FileWriter =
new OutputFileWriter("/Users...M"
+ getDateTime() + ".txt");
    //FileWriter.writeToFile
    ("REMARK File Names After:");
    FileWriter.writeToFile
    ("REMARK File Names Before:");
    FileWriter.writeToFile
    ("REMARK " + inputFilename00);
    FileWriter.writeToFile
    ("REMARK " + inputFilename01);
    FileWriter.writeToFile
    ("REMARK " + ASAfile);
    FileWriter.writeToFile("--------
    -------------------
    -------------------
    ------------------
    -------------");

    for(int c = 0; c < arraySize;c++)
    {
     for(int i = 0; i < myosinSize;i++)
     {
     int resNum =
      myosinStructure.getResNum(i);
     if((resNum ==
     interfaceList[c])
     && (interfaceList[c] != 0))
     {
     //System.out.println
     (interfaceList[c]);
     FileWriter.writeToFile
     (myosinStructure.getWholeLine(i));
     }//end if(resNum == counter)
     }//end for(int c = 0; c < size;c++)
    }//end for(int c = 0; c < arraySize;c++)
```

```
    for(int c = 0; c < arraySize;c++)
    {
     for(int i = 0; i < actinSize;i++)
     {
     int resNum =
     actinStructure.getResNum(i);

     if(resNum ==
     interfaceList[c] && interfaceList[c] != 0)
     {
     FileWriter.
     writeToFile(actinStructure.getWholeLine(i));
     }//end if(resNum == counter)
     }//end for(int c = 0; c < size;c++)
    }//end for(int c = 0; c < arraySize;c++)

    FileWriter.writeToFile("");
}//end public static void findInterfaceResidues...
```

Selected Code from PFAST

```
<?php
include("../includes/utility.php");
checkAdmin("/pfast/index.php");
?>

<html>
<head>
<title>PFAST: Bulk Database Insert Tool</title>
<link rel="stylesheet"
type="text/css" href="../includes/pfast.css">
</head>

<body>

<div id="centercontainer">
<?php include("../includes/header.php"); ?>

<div id="navigation">
<a href="../">Home</a> >> Bulk Database Insert Tool
</div>

<?php
include("../includes/footer.php");
echo "<div id='content'>";
//print_r($_POST);
if ( (isset($_POST['sqlfile'])) &&
($_POST['injectionAction']=='Insert') ) {
if (substr($_POST['sqlfile'],0,3)
!= "sql") header("Location: index.php");
if (!(file_exists($_POST['sqlfile'])))
header("Location: index.php");
echo "<div style='font-weight: bold;
color: red;'>
PLEASE DO NOT REFRESH THIS PAGE!</a>";
$sql_commands = array();
$file_handle =
fopen($_POST['sqlfile'], "r");
$first = true;
$command = "";
while (!feof($file_handle)) {
//(substr($line,0,6) == "INSERT")
```

```
$line = fgets($file_handle);
if ((!$first) &&
(substr($line,0,6) == "INSERT")) {
array_push($sql_commands,$command);
$command = "";
$command .= rtrim($line) . PHP_EOL;
}
else {
$command .= rtrim($line) . PHP_EOL;
$first = false;
}
}
fclose($file_handle);
// echo "<pre>";
// print_r($sql_commands);
// echo "</pre>";
OpenDB();
foreach ($sql_commands as $command)
{
if (mysql_query($command))
{ echo /*$command
.*/ "<br />Inserted successfuly.<br />
<br />"; }
else { echo "[WARNING!]
FAILED TO INSERT!<br />".mysql_error()."
<br /><br />"; }
}
echo "</div>";
//***UNCOMMENT THIS
//TO DELETE SQL STATEMENTS***
//unlink($_POST['sqlfile']);
//
//********
unlink("bulking.txt");
echo "<a href='index.php'>
Perform another insertion.</a>";
}
else if
 ( (isset($_POST['sqlfile']))
  &&
   ($_POST['injectionAction']
   =='delete') )
{
unlink($_POST['sqlfile']);
```

```
header
("Location: bulkInsert.php");
}
else {
header("Location: index.php");
}
echo "</div>";
?>

</div>

</body>
</html>

<-------------->

<?php
include("../includes/utility.php");

if (!isset($_GET['view']))
{
display_normal();
}
elseif ('input' == $_GET['view'])
{
// Connect to database
OpenDB();
// SQL query to open database and
// select a Protein Id
// and code from the table fcat
  //$query = "SELECT * FROM fcat WHERE
   Code='" . $_GET["Code"] .
   "'";
 $query = "SELECT * FROM
  fcat WHERE ProteinID='".
  mysql_real_escape_string
 ($_GET["Protein"])."' AND Code='" .
  mysql_real_escape_string
 ($_GET["Code"]) . "'";
 $result = mysql_query($query);
 $row = mysql_fetch_array($result);


if ($_REQUEST['dl'] == "yes")
```

```
{

header('Content-type: text/plain');
header('Content-Disposition:
 attachment; filename="'
 .$row["ProteinID"]. $row["Code"] .
  '-input-fluor.txt"');
echo $row["InputFile"];

}
else
{
echo "<pre style='font-family:
 monospace; white-space:
 pre-wrap; white-space: -moz-pre-wrap;
  white-space: -pre-wrap; white-space:
   -o-pre-wrap; font-size: 14px;
    word-wrap: break-word; width:
     100%;'>\n" . $row["InputFile"] . "\n</pre>";
}
}
elseif ('sum' == $_GET['view'])
{
// Connect to database
OpenDB();

// SQL query
$query = "SELECT * FROM fcat
WHERE ProteinID='".
mysql_real_escape_string
($_GET["Protein"])."' AND Code='" .
mysql_real_escape_string
($_GET["Code"]) . "'";
$result = mysql_query($query);
$row = mysql_fetch_array($result);

if ($_REQUEST['dl'] == "yes")
{
header('Content-type: text/plain');
header('Content-Disposition:
attachment; filename="' .
 $row["ProteinID"].$row["Code"] .
  '-summary-fluor.txt"');
echo $row["FluorSummary"];
```

```
}
else
{
echo "<pre style='font-family:
 monospace; white-space:
 pre-wrap; white-space:
  -moz-pre-wrap; white-space:
  -pre-wrap; white-space: -o-pre-wrap;
   font-size: 14px; word-wrap:
   break-word; width: 100%;'>\n" .
   $row["FluorSummary"] . "\n</pre>";
}
}
elseif ('res' == $_GET['view'])
{
// Connect to database
OpenDB();

// SQL query
$query = "SELECT * FROM fcat
WHERE ProteinID='".
mysql_real_escape_string
($_GET["Protein"])."'
AND Code='" .
mysql_real_escape_string
($_GET["Code"]) . "'";
$result = mysql_query($query);
$row = mysql_fetch_array($result);

if ($_REQUEST['dl'] == "yes") {
header
('Content-type: text/plain');
header
('Content-Disposition: attachment; filename="'
 . $row["ProteinID"].$row["Code"] .
 '-result-fluor.txt"');
echo $row["FluorResult"];
}
else {
echo "<pre style='font-family:
monospace; white-space: pre-wrap; white-space:
 -moz-pre-wrap; white-space: -pre-wrap;
  white-space: -o-pre-wrap; font-size:
   14px; word-wrap: break-word;
```

```
   width: 100%;'>\n" . $row["FluorResult"] .
   "\n</pre>";
}
}
elseif ('grf' == $_GET['view'])
{
// Connect to database
OpenDB();

// SQL query
$query = "SELECT * FROM fcat WHERE
 ProteinID='".mysql_real_escape_string
($_GET["Protein"])."'
 AND Code='" . mysql_real_escape_string
 ($_GET["Code"]) . "'";
$result = mysql_query($query);
$row = mysql_fetch_array($result);

if ($_REQUEST['dl'] == "yes")
{
header('Content-type: text/plain');
header('Content-Disposition:
 attachment; filename="'
 .$row["ProteinID"].$row["Code"] .
 '-graph-fluor.txt"');
echo $row["FluorGraph"];

}
else
{
echo "<pre style='font-family:
 monospace; white-space: pre-wrap;
 white-space: -moz-pre-wrap; white-space:
  -pre-wrap; white-space: -o-pre-wrap;
   font-size: 14px; word-wrap: break-word;
    width: 100%;'>\n" . $row["FluorGraph"]
    . "\n</pre>";
}
}
elseif ('vis' == $_GET['view'])
{
// Connect to database
OpenDB();
```

```
// SQL query
$query = "SELECT * FROM fcat WHERE ProteinID='"
.mysql_real_escape_string
($_GET["Protein"])."' AND Code='" .
mysql_real_escape_string
($_GET["Code"]) . "'";
$result = mysql_query($query);
$row = mysql_fetch_array($result);

$userid = $_SESSION["userid"];
$tmpfolder = "tmp" . $userid;

@mkdir("./" . $tmpfolder);
file_put_contents('./' .
$tmpfolder . '/Graph.txt',
$row['FluorGraph']);
file_put_contents('./' .
 $tmpfolder . '/Summary.txt',
  $row['FluorSummary']);
  chdir($tmpfolder);

//Clean up all the crap from the last procedure
@unlink("phreq-residuals.gif");
@unlink("phreq-spectra.gif");
@unlink("phreq-sternvolmer.gif");
@unlink("graphs.html");
@unlink("sims1-residuals.gif");
@unlink("sims1-spectra.gif");
@unlink("sims1-sternvolmer.gif");
@unlink("sims2-residuals.gif");
@unlink("sims2-spectra.gif");
@unlink("sims2-sternvolmer.gif");
@unlink("sims3-residuals.gif");
@unlink("sims3-spectra.gif");
@unlink("sims3-sternvolmer.gif");
$_SESSION['graphing'] = True;
include ("../../fat/genGraphs.php");


chdir("..");

//IF there is no code for eg. protein is
// ACY then dont include - after proteinID
if ($row['Code']!="") {
```

```
header("Location: showGraph.php?Protein=" .
 $row['ProteinID'].'-'.$row['Code'] .
  "&searchstring=" . $_REQUEST["searchstring"]);
} else {
header("Location: showGraph.php?Protein=" .
$row['ProteinID'] . "&searchstring=" .
 $_REQUEST["searchstring"]);
}

}
else
{
display_normal();
}




function display_normal()
{

// Connect to database
OpenDB();

// SQL query
//$query = "SELECT * FROM fcat WHERE Code='"
. $_GET["Code"] . "'";

$query = "SELECT * FROM fcat WHERE ProteinID='".
mysql_real_escape_string
($_GET["Protein"])."' AND Code='" .
mysql_real_escape_string($_GET["Code"]) . "'";
$result = mysql_query($query);

// Fetch a row into $row
$row = mysql_fetch_array($result);

//Display "None" if Comments
 field in the databse is empty.
if($row["Comments"] == "" ||
 $row["Comments"] == "-" )
$row["Comments"] = "None";

?>
```

```
<html>
<head>
<title>PFAST: Database</title>
<link rel="stylesheet" type="text/css"
 href="../includes/pfast.css">
</head>

<body>

<div id="centercontainer">

<?php include("../includes/header.php"); ?>

<div id="navigation">
  <div id="leftnav">
<a href="../">Home</a> >>
<a href="../database/database.php">
PFAST Database</a> >>
<a href="../database/database.php?pagemode
=Search&searchstring=<?php echo
$_REQUEST["searchstring"];
?>">Search Results</a> >>
Experiment Details
</div>
<div id="rightnav">
  <a href="databaseHelp.php" align="right">Help</a>
  </div>
</div>

<div id="content">
<?php
    //in case the code is empty , we
    // do not need to use the '-'
    //between ProteinID an Code
    if ($row["Code"]!="")
    {
    echo '<h1>'. trim(stripslashes
    ($row["ProteinID"])).'-'.
    stripslashes($row["Code"]).'</h1>';
    }
    else
    {
    echo '<h1>'.
```

```
    trim(stripslashes($row["ProteinID"])).'</h1>';
    }
?>
<p><b>Protein</b><br><?php echo
 stripslashes($row["Protein"]); ?></p>

<p><b>Submission Date
</b><br><?php echo stripslashes
($row["SubmissionDate"]); ?></p>

<p><b>Source</b><br><?php echo
stripslashes($row["Source"]); ?></p>

<p><b>Measurement
 Conditions</b><br><?php echo
 stripslashes($row["Conditions"]); ?></p>

<p><b>Spectrofluorometer
Settings</b><br><?php echo
 stripslashes($row["Settings"]); ?></p>

<p><b>Quencher</b><br><?php
echo stripslashes($row["Quencher"]); ?></p>
<?php
if ($row["Comments"]=="")
{
 ?>
<p><b>Comments</b><br><?php echo "None" ?></p>
<?php
}
?>
<p><b>Comments</b><br><?php
echo stripslashes($row
["Comments"]); ?></p>
<p><b>Authors</b><br><?php
echo stripslashes($row
["Authors"]); ?></p>
<p><b>Reference</b><br><?php
echo stripslashes($row
["Citation"]); ?></p>

<h2>Fluorescence Analysis Files</h2>

<?php
```

```
if (strlen($row["FluorSummary"])< 1000)
{
echo "None available.";
}

else
{
?>
<dl>
<dt>
Input (plain text) 
<a href="
<?php echo $_SERVER["PHP_SELF"] .
"?Protein=$row[ProteinID]
&Code=
$row[Code]&view=input";
?>">View</a> 
<a href="<?php echo
 $_SERVER["PHP_SELF"] .
  "?Protein=$row[ProteinID]&Code=
$row[Code]&view=input&dl=yes";?>
">Download</a>
</dt>

<dt>
Summary (plain text) 
<a href="<?php echo
$_SERVER["PHP_SELF"] .
"?Protein=$row[ProteinID]&
Code=$row[Code]&view=sum";
?>">View</a> 
<a href="<?php echo
$_SERVER["PHP_SELF"] .
"?Protein=$row[ProteinID]&
Code=$row[Code]&view=sum&dl=yes";
?>">Download</a>
</dt>

<dt>
Results (plain text) 
<a href="<?php echo
$_SERVER["PHP_SELF"] .
"?Protein=$row[ProteinID]&
```

```
Code=$row
[Code]&view=res";?>">View</a> 
<a href="<?php echo
$_SERVER["PHP_SELF"] .
"?Protein=$row[ProteinID]
&Code=
$row[Code]&view=res&dl=yes";?>">Download</a>
</dt>

<dt>
Graph Data (plain text) 
<a href="<?php echo $_SERVER["PHP_SELF"] .
"?Protein=$row[ProteinID]&Code=
$row[Code]&view=grf"; ?>">View</a> 
<a href="<?php echo
 $_SERVER["PHP_SELF"] .
 "?Protein=$row[ProteinID]
 &Code=$row[Code]
&view=grf&dl=yes";?>">Download</a>
</dt>

<dt><a href="<?php echo
$_SERVER["PHP_SELF"] ."?Protein=
$row[ProteinID]&Code=
$row[Code]&view=vis&searchstring=" .
 $_REQUEST["searchstring"];?>">
Visualize Graph</a></p>
</dl>
<?php
}// end of else loop
?>


<?
/*****************************
*****************************
******************/
?>

<h2> Related Experiments </h2>

<?php
// Check protein name in Title, SOURCE, Comments
     $query2 = "SELECT * FROM fcat WHERE
```

```
      instr(Protein, '$row[Protein]')!=0
      OR instr(Source,'$row[Protein]')!=0
       OR instr(Comments, '$row[Protein]')!=0
       ORDER BY SubmissionDate DESC";

$result_pname = mysql_query($query2);

// If no related experiments found,
//display appropriate message
if (mysql_num_rows($result_pname) == 0)
{
echo "No related experiments found for
 $row[Protein] in the PFAST database.";
return;
}

// Matches for Protein name
if (@mysql_num_rows($result_pname)>0)
{//1o
?>

<!-- First row - with headings.--!>

<table id="browsedbresults" cellspacing="0">
<tr>
<th scope="col"
class="leftmost">FCAT Code</th>
<th scope="col">Protein</th>
<th scope="col">Date</th>

<?php
} //1c
?>
</tr>

<!-- Second row onwards - records --!>

<?php
// variables for alternating row styles
$thclass="th_norm"; $tdclass="td_norm";

// Fetch a rows until there are no rows
// in the resultset
while ($row = mysql_fetch_array($result_pname))
```

```
{//2o
?>

<tr>
<?php
                //In case code is empty then dont show
                the '-' between ProteinID and Code
                if ($row["Code"]=="") {
                    ?>
                <th scope="row" class="<?php echo
                 $thclass; ?>"><a href="fcat_details.php?
                 searchstring=<?php echo $searchstring;
                  ?>&Protein=<?php echo stripslashes
                  ($row["ProteinID"]); ?>"><?php
                  echo stripslashes($row["ProteinID"]);
                  ?></a></th>
                <?php
                }
                else if ($row["Code"]!="") {
                ?>
                <th scope="row" class="<?php
                echo $thclass; ?>">
                <a href="fcat_details.
                php?searchstring=<?php echo
                 $searchstring; ?>&Protein=
                 <?php echo stripslashes
                 ($row["ProteinID"]); ?>&
                 Code=<?php echo addslashes
                 ($row["Code"]); ?>"><?php
                 echo trim(stripslashes($row
                 ["ProteinID"])).'-'.
                 stripslashes($row["Code"]);
                  ?></a></th>
                <?php
                }
                ?>


<td class="<?php echo $tdclass;
 ?>"><?php echo stripslashes
 ($row["Protein"]); ?></td>
<td class="<?php echo $tdclass;
 ?>"><?php echo stripslashes
 ($row["SubmissionDate"]);
```

```
  ?></td>
<?php

// some code for alternating row styles
if ($thclass=="th_norm")
{
$thclass="th_alt";
$tdclass="td_alt";
}
else
{
$thclass="th_norm";
$tdclass="td_norm";
}


  }//end while 2c
   ?>
        </table>


<?php

/*****************************
*******************
*****************************/


?>


</div>

<?php include("../includes/footer.php"); ?>

</div>
</body>
</html>

<?php
} // end function display_normal()
?>

<------------>
```

```
<?php
/*
SCAT System 2.0
Brad Boudreau
Fall2011
Now with added support for
 uploading custom PDB files.
*/
include("../includes/utility.php");
checkSignedIn("/pfast/index.php");
error_reporting(-1);

//--- Pre-defined Environment Variables ---
define(LINE_BREAK, "\r\n");
// Windows-style linebreak
define(LINE_BREAK_REGEXP, '\r\n');
// Windows-style linebreak
define(PURGE, "del ");
// Windows delete command

//define(LINE_BREAK, "\n");
// *nix linebreak
//define(LINE_BREAK_REGEXP, '\n');
// *nix linebreak
//define(PURGE, "rm ");
// *nix delete command
//--- Pre-defined Environment Variables ---


if (isset($_REQUEST['pagemode'])
&& ($_REQUEST['pagemode']
== "Finish >>")) {
// The second stage on the page
// Display the chain data
verify_stage2();
// stage3();

}
else if
(isset($_REQUEST['pagemode'])
 && ($_REQUEST['pagemode']
  == "Step 2 >>")) {
// The third stage on the page
```

```
// Display the results

verify_stage1();
stage2();
}
else if (isset($_REQUEST['pagemode']
) && ($_REQUEST['pagemode'] ==
"<< Step 1")) {
stage1();

}
else {
// The first time we've gone to the page.
// Display the first page,
//requesting the PDB code or PDB data.
clean_up();
$_SESSION['timestamp']
 = timestamp();
stage1();
}


// Erase old session variables
// Ensure there is no old data.
function clean_up() {
for ($i='A'; $i<='Z'; $i++)
 { unset($_SESSION["chain".$i]); }
    unset($_SESSION["timestamp"]);
    unset($_SESSION["chainKeys"]);
    unset($_SESSION["chainData"]);
    unset($_SESSION["PDBcode"]);
unset($_SESSION["PDBfile"]);
}

//--- Verification ---
function verify_stage1() {
// Which type of file
//are we provided with?
// Options from the
//previous page are "web" or "local"
$type = htmlentities
($_POST['filesource']);
$pdbfile = "";
if ($type == "web") {
```

```
// Load the PDB code from the
//$_POST data.
if (strlen($_POST['PDBcode'])
 < 4) {
stage1("Please enter
a PDBcode."); die();
}
$pdbcode = htmlentities
($_POST['PDBcode']);
$_SESSION['PDBcode']
= $_POST['PDBcode'];

// Construct URL and read PDB file from RCSB into string
$pdburl = 'http://www.rcsb.org/pdb/
//download/downloadFile.
//do?fileFormat=pdb&compression
//=NO&structureId=' . $pdbcode;
@$pdbfile =
file_get_contents($pdburl);
} elseif ($type == "local")
 {
// We want to parse the
//custom pdb text
if (!eregi('text/',$_FILES
['custompdb']['type'])) {
stage1("Please upload
only a PDB text file.")
; die();
}
else {
$pdbcode = "custom";
$_SESSION['PDBcode']
 = "custom";
move_uploaded_file(
$_FILES['custompdb']
['tmp_name'], $_SESSION
['timestamp'].'.txt');
@$pdbfile =
file_get_contents
($_SESSION['timestamp'].'.txt');
}
} else {
// Someone broke something.
// This is possibly caused by
```

```
someone messing with a DOM Editor
 or not clicking on an option.
stage1("Please choose one of
either web source of local
source.");
die();
}
$_SESSION['sourcetype'] = $type;

    // Check that we actually got something
if (is_this_a_pdbfile($pdbfile) == 0) {
if ($type == "web") stage1('The PDB
code you provided ['. $pdbcode .']
 could not be downloaded from the
  RCSB website.');
elseif ($type == "local")
 stage1('There was an error
 uploading the PDB file.
  Please try again.');
die();
    }

    // Make temporary directory, change to it
    $tempdir = $_SESSION['timestamp'];
    //if(!(mkdir("./" . $tempdir)))
    // { echo("Cannot make directory: ./$tempdir"); die(); }
mkdir("./" . $tempdir);
// $temppath = "./" . $tempdir . "/";
    chdir($tempdir);

    // Delete previous files
    // There should never be
    // any previous files.
//  @exec(PURGE ."*.ent");
//  @exec(PURGE ."*.txt");
// @exec(PURGE ."*.exe");
//  @exec(PURGE ."*.* /q");

    // Save the PDB file for use later
file_put_contents($pdbcode .
 ".txt", $pdbfile);
//$_SESSION['PDBfile'] =
serialize($pdbfile);
    chdir("..");
```

```
    unlink($_SESSION['timestamp'].'.txt');
$chainData = getChainData($pdbfile);

    // Save data to session
    $_SESSION["chainData"] =
    serialize($chainData);
}

function verify_stage2() {
set_time_limit(1800);

// Make sure that at
//least one chain was selected
$hasAtLeastOne = false;
$chainKeys =
$_SESSION["chainKeys"];
$chainData =
unserialize
($_SESSION["chainData"]);
$multiResList =
 unserialize
 ($_SESSION["multiResList"]);
$pdbfile =
file_get_contents($_SESSION['timestamp']."/".$_SESSION['PDBcode'].".txt");
$validChainKeys = array();
// Are there more than one
//chain in this PDB file?
if (sizeof($chainKeys) > 1) {

// For each chain,
foreach ($chainKeys as $chainKey)
{
// If the current chain
//was selected in the previous screen...
if (isset($_POST["chain" .
$chainKey])) {

// Note that at least
// one chain has been selected
$hasAtLeastOne = true;

// Add current chain
//to list of chains to process
$validChainKeys[] = $chainKey;
```

```
}
}
}
if (sizeof($chainKeys) == 1)
 {
 // This used to be an ELSE statement
// There is only one chain
$hasAtLeastOne = true;
$validChainKeys[] = $chainKeys[0];
}
if (!$hasAtLeastOne) {
stage2("Please select
 at least one chain.");
return;
}

$pdbcode = $_SESSION['PDBcode'];
// Change to tmp directory (it
//should have been already
//created in step 1)
$tempdir = $_SESSION['timestamp'];
chdir($tempdir);
// $pdbfile = unserialize
//($_SESSION['PDBfile']);
// Check that nothing went wrong
 with the PDBfile
if (is_this_a_pdbfile($pdbfile)
 == 0) {
stage2('Unexpected error
when reading PDB file.');
return;
}
////////////////////////////////////////
// Filter out some (maybe none) of the
// ATOM/HETATM lines in the PDB string.

$validResidueNames = explode(' ',
'GLY ALA VAL LEU ILE MET PHE TRP
 PRO SER THR CYS TYR ASN GLN ASP
  GLU LYS ARG HIS');

// Grab all the lines beginning with
// ATOM__ from the PDB file.
preg_match_all("/^(ATOM
```

```
 |HETATM|TER   ).[^" .
  LINE_BREAK_REGEXP . "]+/m",
   $pdbfile, $lines);
$atoms = $lines[0];
echo "Test9";
$pdbfile_orig = $pdbfile;
foreach ($atoms as $atom) {
// Extract information for each atom line
// From http://www.wwpdb.org/
//\//documentation/format23/sect9.html
//
//  Column
//  Range
//Description
// ---------
//-----------------------------------
//  01 - 06
//Record name (ATOM, HETATM, etc.)
//  07 - 11
//Atom serial number
//  13 - 16
//Atom name
//  17 - 17
//Alternate character indicator
//  18 - 20
//Residue name *
//  22 - 22
//Chain ID *
//  23 - 26
//Residue sequence number
//  27 - 27
//Code for insertion of residues *
//  31 - 38
//Orthogonal coordinates for X (angstroms)
//  39 - 46
//Orthogonal coordinates for Y (angstroms)
//  47 - 54
//Orthogonal coordinates for Z (angstroms)
//  55 - 60
//Occupancy
//  61 - 66
//Temperature factor
//  77 - 78
//Element symbol, right-justified
```

```
//  79 - 80
//Charge on the atom
//
//  * fields we need here

$recName = trim(substr($atom, 0, 6));
$altChar = trim(substr($atom, 16, 1));
$chainID = trim(substr($atom, 21, 1));
if ($chainID == '') $chainID = '1';
$chainKey = $chainID . '^' . $recName;
$resName = trim(substr($atom, 17, 3));
$resNmbr = trim(substr($atom, 22, 4));
$resCode = trim(substr($atom, 26, 1));
// Filter out atoms of chains
//that weren't selected
if (!in_array($chainKey,
$validChainKeys)) {
// Get rid of current line
$pdbfile =
str_replace
($atom . "\n", '', $pdbfile);
continue;
}
// Filter out lines that
//don't meet subset
// selection requirement
if ($recName != 'TER' &&
!inSubsetSelect($resNmbr,
 $_POST['subset' . $chainKey])) {
// Get rid of current line
$pdbfile = str_replace($atom
 . "\n", '', $pdbfile);
continue;
}

// Filter out lines that
//don't meet multi-residue
//selection requirement
$multiResKey = $chainKey . '-' . $resNmbr;
if (in_array($multiResKey, $multiResList))
 {
$selectedResData =
explode('|', $_POST[$multiResKey]);
$selectedAltChar =
```

```
 $selectedResData[0];
$selectedResName =
$selectedResData[1];
// If it's the wrong
// residue name, or the
//wrong alt char for the
//right residue name
if ($resName !=
$selectedResName ||
($altChar != '' && $altChar
!= $selectedAltChar)) {
// Get rid of current line
$pdbfile =
 str_replace
 ($atom . "\n", '', $pdbfile);
}
}
}

$chainData = getChainData($pdbfile);
//Here we check if the selected
//residues contain atleast one
// TRP, this is important
//otherwise the programs will crash
chdir('../');
$result = array();
foreach ($chainData as $key => $values) {
$result = array_merge($result,$values);
}
if(!in_array('|TRP',$result)) {
stage2("Please select at
least one TRP residue");
return;
}

chdir($tempdir);

// echo "<pre>\n" .
$pdbfile . "</pre>";
// die();

//////////////////
// LINE 1: Date
$output_date = date("Y-M-d H:i:s");
```

```
//////////////////////
// LINE 2: PDB code
$output_pdbcode = $pdbcode;

//////////////////////
// LINE 3: Compound
preg_match_all("/^COMPND....
(.{60})/m", $pdbfile, $lines);
foreach ($lines[1] as $line)
$output_compound .= trim($line) . " ";

//////////////////////
// LINE 4: Source
preg_match_all("/^SOURCE....
(.{60})/m", $pdbfile, $lines);
foreach ($lines[1] as $line)
$output_source .= trim($line) . " ";

//////////////////////
// LINE 5: Author
preg_match_all("/^AUTHOR....
(.{60})/m", $pdbfile, $lines);
foreach ($lines[1] as $line)
 $output_author .= trim($line) . " ";

//////////////////////////
// LINE 6: Resolution
preg_match_all("/RESOLUTION[^A\n]
+ANGSTROMS/", $pdbfile, $matches);
// In practice, there will only
// be one RESOLUTION in the PDB file.
// In any case, we always select the first.
$output_resolution =
str_replace
('RESOLUTION.', 'RESOLUTION', $matches[0][0]);
//User inserted comment about his/her analysis
$output_comment='Comment:  '.$_POST['comment'];

////////////////////////////////////
// LINE 7: Number of residues,
number of trp, trp positions
$numRes = 0;
$numTRP = 0;
```

```
$trpPositions = array();
foreach ($chainData as $chainKey => $residues) {
foreach ($residues as $resNum => $resKey) {
$numRes++;
$resData = explode('|', $resKey);
$resChar = $resData[0];
$resName = $resData[1];
if ($resName == 'TRP') {
$numTRP++;
$pieces = explode('^', $chainKey);
$chainID = $pieces[0];
$chain_offset = 2000 *
(array_search($chainID .
'^ATOM', $validChainKeys)/2);
$adjResNum =
$chain_offset +
$resNum;
$trpPositions[] =
$adjResNum;
}
}
}
$trpPositions =
implode('; ', $trpPositions);
$output_line7 =
"[" . $numRes . "] "
 . $numTRP . " TRP
  {" . $trpPositions . "}";
$output_trpPositions =
implode(LINE_BREAK,
 explode('; ', $trpPositions));
   // Line 9++

// Pass numTRP as session
//variable to save.php
$_SESSION["numTRP"] = $numTRP;

////////////////////////////////
// LINE 8: num atoms, num TRP
// counting all ATOM or HETATM,
// need to exclude unwanted chains
preg_match_all("/^(ATOM  |HETATM).{15}
[ " . implode($validChainKeys) . "]
[^" . LINE_BREAK_REGEXP . "]
```

```
+/m", $pdbfile, $lines);
$numAtoms = sizeof($lines[0]);
$output_line8 = $numAtoms . "," . $numTRP;

////////////////////////////////////
// ATOMs and HETATMs, TRP positions
// counting all ATOM or HETATM, need
//to exclude unwanted chains
preg_match_all("/^(ATOM  |HETATM).{15}
[ " . implode($validChainKeys) . "]
[^" . LINE_BREAK_REGEXP . "]+/m",
$pdbfile, $lines);
$atoms = array();
$atoms_nowater = array();
$nbatoms = array();
//myadd
$trpPositions = array();
foreach ($lines[0] as $line) {
$columns = array();
$columns[] = (substr($line, 0, 6));
$columns[] = (substr($line, 6, 5));
$columns[] = (substr($line, 11,6));
$columns[] = (substr($line, 17, 4));
 // residue
$columns[] = " ";
$chainKey = (substr($line, 21, 1))
 . '^ATOM'; // Chain Key
$chain_offset = 2000 *
(array_search($chainKey,
 $validChainKeys)/2);
$resNumber = sprintf
("%4d", $chain_offset +
(substr($line, 22, 4)));
// residue number
$columns[] = $resNumber;
$columns[] = (substr($line, 26, 5));
$columns[] = (substr($line, 31, 8));
$columns[] = (substr($line, 39, 8));
$columns[] = (substr($line, 47, 9));
$columns[] = (substr($line, 56, 6));
$columns[] = (substr($line, 62, 11));
$columns[] = (substr($line, 73, 9));
$atoms[] = implode('', $columns);
if (trim(substr
```

```
($line, 16, 4)) != "HOH")
$atoms_nowater[] =
implode('', $columns);
}
$output_atoms =
implode(LINE_BREAK, $atoms);
$output_atoms_nowater =
implode(LINE_BREAK, $atoms_nowater);

//extracting columns for enter-nb.txt
foreach ($lines[0] as $line) {
$columns = array();
$columns[] = str_pad
(substr($line, 0, 6), 14);
$columns[] =
(substr($line, 12, 5));
$columns[] =
(substr($line, 17, 4));
 // residue
//$columns[] = " ";
$chainKey = (substr($line, 21, 1))
 . '^ATOM'; // Chain Key
$chain_offset = 2000 *
(array_search($chainKey,
 $validChainKeys)/2);
$resNumber = sprintf
(" %4d", $chain_offset +
(substr($line, 22, 4)));
// residue number
$columns[] = $resNumber . " ";
$columns[] = (substr($line, 30, 8));
$columns[] = (substr($line, 38, 8));
$columns[] = (substr($line, 46, 8));
$columns[] = (substr($line, 61, 5));
$nbatoms[] = implode(',', $columns);
}

$output_atoms_nb = implode(LINE_BREAK, $nbatoms);


//////////////////////////////////////////////////
// Compose enter.ent, enter-wh.ent and enter-nb.txt
$common = '';
$common .= $output_date . LINE_BREAK;
```

```
$common .= $output_pdbcode . LINE_BREAK;
$common .= $output_compound . LINE_BREAK;
$common .= $output_source . LINE_BREAK;
$common .= $output_author . LINE_BREAK;
$common .= $output_comment . LINE_BREAK;
$common .= LINE_BREAK . LINE_BREAK;
$common .= $output_resolution . LINE_BREAK;
$common .= $output_line7 . LINE_BREAK;

//top of result-table contains
//data up to this line also summary file
$result_table_top= $common;

$common .= $output_line8 . LINE_BREAK;
$common .= $output_trpPositions . LINE_BREAK;

$enter = $common . $output_atoms;
$enterwh = $common . $output_atoms_nowater;
$enternb = $common . $output_atoms_nb;


//////////////////
// Save to files
file_put_contents('enter.ent', $enter);
file_put_contents('enter-wh.ent', $enterwh);
file_put_contents('enter-nb.txt', $enternb);

//////////////////////
// Copy the programs from
//storage to the directory for use.
copy("../RICHBC-H.exe", "./RICHBC-H.exe");
copy("../RICHB-WH.exe", "./RICHB-WH.exe");
copy("../NB1.exe", "./NB1.exe");
copy("../NB2.exe", "./NB2.exe");
copy("../NB3.exe", "./NB3.exe");
// These DLLs are required
//for the FORTRAN programs
copy("../sdbgdll.dll",
"./sdbgdll.dll");
copy("../salflibc.dll",
"./salflibc.dll");

// Execute RICHB-WH and
//RICHBC-H programs.
```

```
exec("RICHB-WH");
exec("RICHBC-H");


// For NB1, NB2, and NB3, if
//there is an error, the program
// will generate an error file.
//The PHP script will verify at
// each stage that the error file
//wasn't generated.  If the file
// was generated, the PHP script will
 //throw an error and revert to
// the previous step.
exec("NB1");
if (file_exists("NB1error.txt")) {
chdir('..');
stage2("Due to some unexpected
error in the input pdb file,the
 analysis was terminated !");
return;

}

exec("NB2");
if (file_exists("NB2error.txt")) {
chdir('..');
stage2("Due to some unexpected
 error in the input pdb file,the
 analysis was terminated !");
return;
}

exec("NB3");
if (file_exists("NB3error.txt")) {
chdir('..');
stage2("Due to some
 unexpected error in the
 input pdb file,the analysis
  was terminated !");
return;
}

//generating file Result-table
//from Energy and Result-2 files
```

```
$result_2 = file("Result-2")
 or die('Could not read file!');
$energy = file_get_contents("Energy");

//extracting all lines
// from line 12 of Result-2 file
$part_result2 =
array_slice($result_2,12);

$newfile = ("Result-table.txt");
$fp = fopen($newfile,"a");
$resultTable =
$result_table_top."\r\n".
$energy.join('', $part_result2);

//putting the entire contents of
 //energy file after 12 lines from
  //Result-2 file and then rest
  //of the result-2 file
file_put_contents($newfile, $resultTable);

//////////////////////
// The following code will extract
the TRP
positions and six parameters
// from the 16-parameters file
which will be used to generate the summary file.
//
$data = file_get_contents
("16-parameters");
preg_match_all("/([0-9]+)
 .[ ]*TRP +([^\n\r]+)/",$data,$trp);
//reading the value of
 "Ne1 atom (without water molecules)"
 for each TRP into an array
preg_match_all("/2\. Accessibility
 \(%\) of Ne1 atom \(without water
  molecules\) +([^\n\r]+)/",
  $data, $param2);
//reading the value of
 "CZ1 atom (without water molecules)"
 for each TRP into an array
preg_match_all("/4\.  Accessibility
 \(%\) of CZ2 atom \
```

```
 (without water molecules\)
 +([^\n\r]+)/", $data, $param4);
//reading the value of
"Trp atom (without water molecules)"
 for each TRP into an array
preg_match_all("/6\.
Accessibility \(%\) of Trp
\(without water molecules\)
+([^\n\r]+)/", $data, $param6);
//reading the value
//of packing density of TRP
preg_match_all("/8\.
Packing density in sphere 7.5A
 +([^\n\r]+)/", $data, $param8);
//reading $fp=fopen($newsum,"a");the
//value of Parameter B1 of TRP
preg_match_all("/9\.
Parameter B1 +([^\n\r]+)/", $data, $param9);
//reading the value of
//Parameter B2 of TRP
preg_match_all("/10\.
Parameter B2 +([^\n\r]+)/", $data, $param10);
//reading the value of
//Parameter R1 of TRP
preg_match_all("/11\.
Parameter R1 +([^\n\r]+)/", $data, $param11);
//reading the value of
// Parameter R2 of TRP
preg_match_all("/12\.
Parameter R2 +([^\n\r]+)/", $data, $param12);
//reading the value of
//Parameter A1 of TRP
preg_match_all("/13\.
Parameter A1 +([^\n\r]+)/", $data, $param13);
//reading the value of
//Parameter A2 of TRP
preg_match_all("/14\.
Parameter A2 +([^\n\r]+)/", $data, $param14);

//for each TRP calculating
//the six parameters
$sixparams = array();
foreach ($param2[1] as $key => $value) {
$params = array();
```

```
$fp = fopen($newsum,"a");$params[]
= 0.0 + $param6[1][$key];
$params[] = (0.0 + $param2[1][$key]
+ $param4[1][$key]) / 2.0;
$params[] = 0.0 + $param8[1][$key];
$params[] = (0.0 + $param9[1][$key]
+ $param10[1][$key]) / 2.0;
$params[] = (0.0 + $param11[1][$key]
+ $param12[1][$key]) / 2.0;
$params[] = (0.0 +$param13[1][$key]
 + $param14[1][$key]) / 2.0;
$sixparams[] = implode(" ", $params);
}
$str= implode("\r\n",$sixparams);
$trp = implode(",", $trp[0]);

//writing sixparameters in
 //sixpara.txt
//file and extracting TRP
// positions in trp.txt file
file_put_contents("sixpara.txt",$str );
file_put_contents("trp.txt",$trp);

@copy("../winpfastmatx.exe",
 "./winpfastmatx.exe");

//The file note.txt contains
//the notes which appear at
//the bottom of the summary file
copy("../note.txt", "./note.txt");

//executing the C# program
// to calculate Mahalanobis distance
// and Classification score
exec("winpfastmatx");

//writing first 15 common
//lines at the top of the final.txt
//file to generate the summary file'
//final.txt is the output
// of the winpfastmatx program
$sumdata = file_get_contents
("final.txt") or die('Could
 not read file!');
```

```
file_put_contents
("summary.txt",join
('', $result_2). "\r\n" . $sumdata);
file_put_contents
("summary.txt",$result_table_top.
 "\r\n" . $sumdata);
@exec(PURGE . "*.exe");
@exec(PURGE . "*.dll");
chdir('..');
unset($_SESSION['sourcetype']);
header("Location: result.php?dir=
" . $_SESSION['timestamp']);
}
//--- Verification ---

//--- User Front-end ---
function stage1($errormsg = "") {
echo '
<html>
<head>
<title>PFAST: Structural
Correlation Analysis Tool</title>
<link rel="stylesheet"
type="text/css" href=
"../includes/pfast.css">
</head>

<body>

<div id="centercontainer">
';
include("../includes/header.php");
echo '
<div id="navigation">
<div id="leftnav">
<a href="../">Home</a> >> Structural
Correlation Analysis Tool </div>
<div id="rightnav">
<a href="scathelp.php"
align="right">SCAT HELP</a>
</div>
</div>

<div id="content">
```

```
<h1>SCAT: Structural
Correlation Analysis Tool</h1>
<p>
This tool is designed
 for the analysis of
 structural properties of
  tryptophan environment
  in proteins.
As a result of the SCAT
 module, the structural
  properties of tryptophan
  residues can be calculated
and the tryptophan
residues can be assigned
 to one of the five
  spectral-structural classes.
</p>
<h2>Step 1: Enter PDB Code</h2>
<p>Please enter 4 character PDB code.<p>
';
ShowErrorMessage($errormsg);
echo '
<form action="';
echo $_SERVER['PHP_SELF'];
echo '" method="post"
enctype="multipart/form-data">
<table>
<tr>
<td><input type="radio"
name="filesource"
value="web">PDB Website<br></td>
<td><input type="text"
maxlength="4" size="5"
name="PDBcode" value="';
echo $_SESSION['PDBcode']; echo '">
</td>
</tr>

<tr>
<td><input type="radio"
 name="filesource"
 value="local">Custom PDB File</td>
<td><input
name="custompdb" type="file"></td>
```

```
</tr>
</table>
<input type="submit"
name="pagemode"
value="Step 2 >>">
</form>
</div>
';
include("../includes/footer.php");
echo '
</div>

</body>
</html>
';
}

function stage2
($errormsg = "") {
$chainData =
unserialize
($_SESSION["chainData"]);
$_SESSION["chainKeys"] =
array_keys($chainData);
$multiResList = array();

echo '
<html>
<head>
<title>PFAST: Structural
 Correlation Analysis Tool</title>
<link rel="stylesheet" type=
"text/css" href=
"../includes/pfast.css">
</head>

<body>
<div id="centercontainer">
';

include("../includes/header.php");

echo '
<div id="navigation">
```

```
<div id="leftnav">
<a href="../">Home</a> >>
 Structural Correlation
 Analysis Tool
</div>
<div id="rightnav">
<a href="scathelp.php"
align="right">SCAT HELP</a>
</div>
</div>

<div id="content">

<h1>SCAT: Structural Correlation
 Analysis Tool</h1>
<h2>Step 2: Select Chains and
Residues</h2>

<p>The following chains were found
in the PDB file.  If there is more
than one chain, please <b style="color:
 red;">use the checkboxes to indicate
 which chains</b> are to be used for
  analysis.</p>
<p> For each chain a subset
of the residues could be selected:
 enter the range of residues (inclusive)
  in the text box. If you leave the box
  blank, all residues in the chain will
  be used. Examples:</p>

<ul>
<li>1-400 (select residues #1
to #400)</li>
<li>10-100,105,110-200
(select residues #10 to #100,
 #105, and #110 to #200)</li>
<li><b style="color: red;">
Do not enter any spaces!</b></li>
<li><b style="color: red;">
Do not include chains, which
 contain only Calpha atoms!</b></li>
<li><b style="color: red;">
Do not include chains, which
```

```
contain Hydrogen atoms!</b></li>
</ul>
<hr>
';
if ($_SESSION['sourcetype']
 == "web") echo'<p>You have
 selected the pdb file  :'.
 '<a href= "http://www.rcsb.org/pdb/files/'
 .$_SESSION['PDBcode'].'.pdb"
 target= "_blank">'.$_SESSION['PDBcode'].
 '</a>' ;
if ($_SESSION['sourcetype'] == "local")
 echo '<p>You have submitted your
  own file</p>';
// if (isset($_SESSION['PDBcode']))
 echo '<p>You have selected the
 pdb file  :'.'<a href=
 "http://www.rcsb.org/pdb/files/'.
 $_SESSION['PDBcode'].'.pdb" target=
  "_blank">'.$_SESSION['PDBcode'].
  '</a>' ;
// else echo '<p>You have
submitted your own file</p>';

echo '<br>';
if (sizeof($chainData)>1) {
echo "<p>The ATOMs and HETATMs
in the specified PDB file are
listed below, grouped by chains
 if applicable.  You can use
 the following links to jump
 to their respective places on
 <b>this page</b>.</p><ul>";

foreach ($chainData as
$chainKey => $residues) {
$pieces = explode('^', $chainKey);
$chainID = $pieces[0];
$atomType = $pieces[1];
if (sizeof($chainData)>2
&& $chainID == '1') {
echo "<li><a href=
'#chain$chainKey'>Shared
 ${atomType}s</a></li>";
```

```
}
else {
echo "<li><a href=
'#chain$chainKey'>Chain
 $chainID ${atomType}s</a></li>";
}
}

echo "</ul><hr>";
}
ShowErrorMessage($errormsg);

render_chain_data($chainData);

echo '</div>';
include("../includes/footer.php");
echo '</div></body></html>';
}

function stage3() {

}
//--- User Front-end ---


//--- Utility Functions ---

//http://deposit.rcsb.org/
//adit/docs/pdb_atom_format.html
function is_this_a_pdbfile($pdbfile) {
if (strlen($pdbfile)<431) return 0;
return 1;

}

function timestamp() {
$stamp = $_SESSION['userid']
 ."-" .(mt_rand(1,99)) .
 (date("-Y-M-d-s"));
return $stamp;
}

function render_chain_data($chainData)
 {
```

```
// allows user to add
any comment about his analysis
echo "<form action='"
.$_SERVER['PHP_SELF']."' method='post'>";
echo "<b>Insert any comment
 below :</b>[Please enter here
 the chainIDs, range of residues selected for
analysis and other information]<br>";
echo "<textarea name='comment'
 cols=120 wrap='soft' value='" .
 trim($_POST["comment"]) ."''>
 </textarea><br>";
$sep = '';

foreach ($chainData as $chainKey
 => $residues) {
$pieces = explode('^',
 $chainKey);
$chainID = $pieces[0];
$atomType = $pieces[1];

echo "<a name='chain$chainKey'></a><h3>";
if (sizeof($chainData)>1) {
echo "<input type='checkbox'
 name='chain" . $chainKey . "' ";
if ($_POST['chain' .
$chainKey]) echo "CHECKED";
echo (sizeof($chainData)==1)
 ? ' DISABLED' : '';
echo ">";
}
if (sizeof($chainData)>2
&& $chainID == '1') {
echo "Shared
${atomType}s</h3>";
}
else {
echo "Chain $chainID
${atomType}s</h3>";
}

echo "Subset Selection:
<input type='textbox' size='30'
 name='subset" . $chainKey . "'
```

```
 value='" . trim($_POST["subset"
 . $chainKey]) ."''><br>";
$count = 1;
echo "<center><table
class='scatResidueTable'
cellspacing=0 cellpadding=0>";
$numRes = sizeof($residues);
foreach ($residues as
$resNum => $resKey) {
if ($count % 8 == 1)
 echo "<tr class='scatResidueTable'>";
echo "<td class=
'scatResidueTable resNum'>";
echo str_replace(' ',
' ', sprintf("%5d", $resNum));
echo "</td>";
echo "<td class='
scatResidueTable resName'>";
if (is_array($resKey))
{
$multiResList[] =
$chainKey . '-' . $resNum;
echo "<select class=
'scatResidueTable' name='" .
 $chainKey . '-' .
  $resNum . "'>";
foreach ($resKey as $resKeyItem)
 {
$resData = explode('|', $resKeyItem);
$resAlt = $resData[0];
$resName = $resData[1];
$resOptionLabel = $resName .
strtolower($resAlt);
$resOptionValue = $resKeyItem;
$isSelected =
($_POST[$chainKey . '-' .
 $resNum] == $resOptionValue) ?
  ' SELECTED' : '';
echo "<option value='" .
 $resOptionValue . "'"
 . $isSelected . ">" .
 $resOptionLabel . "</option>";
}
echo "</select>";
```

```
}
else {
$resData = explode('|', $resKey);
$resAlt = $resData[0];
$resName = $resData[1];
echo ' ' . $resName;
}
echo "</td>";
$count++;
if ($count % 8 == 1) echo "</tr>";
}
$_SESSION['multiResList'] =
serialize($multiResList);

while ($count % 8 != 1)
{
echo "<td> 
</td><td> </td>";
$count++;
if ($count % 8 == 1) echo "</tr>";
}
echo '
</table>
</center>
<hr>
';
}
echo '
<input type="submit"
 name="pagemode" value="<< Step 1">
<input type="submit"
name="pagemode" value="Finish >>">
</form>
';
}

function getChainData($pdbfile) {
    ////////////////////////////////////////////
    // Iterate through the ATOM entries to
    // 1. Determine how many chains there are
    // 2. Determine if any of the residue numbers
    //    have multiple entries
    //
    // Hold this information in $chainData, which
```

```
// maps chainIDs to a map of residue numbers
// mapping to the three-letter residue names.
// In the case where there are multiple
// residues for a particular residue number,
// that reside number will instead map to an
// array of residue names.
//
$chainData = array();

$validResidueNames =
explode(' ', 'GLY ALA VAL
 LEU ILE MET PHE TRP PRO
 SER THR CYS TYR ASN GLN
 ASP GLU LYS ARG HIS');

// Grab all the lines beginning
// with ATOM__ from the PDF file.
preg_match_all("/^(ATOM  |HETATM).[^" .
 LINE_BREAK_REGEXP . "]+/m",
  $pdbfile, $lines);
$atoms = $lines[0];

foreach ($atoms as $atom) {
    // Extract information
    //for each atom line

    // From http://www.wwpdb.org/
    //documentation/format23/sect9.html
    //
    //  Column
    //  Range
    //Description
    // ---------
    //-----------------------------------
    //  01 - 06
    //Record name (ATOM,
    //HETATM, etc.)
    //  07 - 11
    //Atom serial number
    //  13 - 16
    //Atom name
    //  17 - 17
    // Alternate character indicator
    //  18 - 20
```

```
// Residue name *
//  22 - 22
//Chain ID *
//  23 - 26
// Residue sequence number
//  27 - 27
//Code for insertion of residues
//  31 - 38
//Orthogonal coordinates
// for X (angstroms)
//  39 - 46
//Orthogonal coordinates
// for Y (angstroms)
//  47 - 54
//Orthogonal coordinates
//for Z (angstroms)
//  55 - 60
//Occupancy
//  61 - 66
//Temperature factor
//  77 - 78
//Element symbol,
// right-justified
//  79 - 80
// Charge on the atom
//
//  * fields we need here

$recName =
trim(substr($atom, 0, 6));
$chainID =
trim(substr($atom, 21, 1));
if ($chainID == '')
 $chainID = '1';
$chainKey = $chainID . '^' . $recName;

$altChar =
trim(substr($atom, 16, 1));
$resName =
trim(substr($atom, 17, 3));
$resNmbr =
trim(substr($atom, 22, 4));

$resKey = $altChar .
```

```
'|' . $resName;

// if (!in_array($resName,
 $validResidueNames)) continue;

if (!array_key_exists($chainKey,
$chainData))
$chainData[$chainKey]
= array();

/*
if ($resNmbr == '24') {
    echo "<br><br>[" .
    $resKey . "]<br>";
    print_r($chainData
    [$chainKey][$resNmbr]);
}
*/

// Do we have an entry for this
//residue number in this chain?
if (!array_key_exists($resNmbr,
 $chainData[$chainKey])) {

    // No, then just add
    //an entry mapping
    // the residue number
    //to the residue name.
    $chainData[$chainKey]
    [$resNmbr] = $resKey;
}
else {

    // Yes, then does the existing
    //entry for this residue
    // number map to something
    //different than the
    //current name?
    if ($chainData[$chainKey]
    [$resNmbr] != $resKey) {

        // Yes, then does the existing
        // entry for this
        // residue number map to an array?
```

```
if (is_array($chainData
[$chainKey][$resNmbr])) {

    // Yes, and if the array
    //doesn't already contain
    // the current name, then add it.
    if (!in_array($resKey,
    $chainData[$chainKey]
    [$resNmbr])) {
        $chainData[$chainKey]
        [$resNmbr][] = $resKey;
    }

    // If adding an entry
    // with an alternate
    //character, then we need
    //to remove the entry without
    //an alt char
    if ($altChar != '')
    {  //  && in_array('|' .
     $resName, $chainData
     [$chainKey][$resNmbr])
        foreach ($chainData
        [$chainKey][$resNmbr]
        as $key => $value) {
            // echo "<br>Checking
             [" . $key . "]
             vs [" . '|' .
             $resName . "]";
            if ($value == '|'
            . $resName)
             {
                unset($chainData
                [$chainKey]
                [$resNmbr][$key]);
            }
        }
    }

}
else {

    // No, then
    //convert the existing entry
```

```
                    // into an array,
                    //and add the current
                    // name to that array.
                    $chainData[$chainKey][$resNmbr]
                    = array($chainData[
                    $chainKey][$resNmbr], $resKey);
                }
            }
        }
    }

    return $chainData;
}

function inSubsetSelect
($candidate, $filter='') {
    if ($filter == '') return true;

    $validValues = array();

    $ranges =
    explode(',', $filter);
    foreach
    ($ranges as $range) {
        $ends = explode('-', $range);
        if (sizeof($ends) == 2) {
            for ($i=$ends[0];
            $i<=$ends[1]; $i++) {
                $validValues[] = '' . $i;
            }
        }
        else if (sizeof($ends) == 1)
        {
            $validValues[] =
            '' . $ends[0];
        }
    }
   return
    (in_array($candidate, $validValues));
}
//--- Utility Functions ---
?>
```