Applications of machine learning to problems in biomolecular function and dynamics

Ruth Veevers

School of Computing Sciences

University of East Anglia

This dissertation is submitted for the degree of

Doctor of Philosophy

School of Computing Sciences September 2020

Dedicated to Anthony.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Ruth Veevers

September 2020

Acknowledgements

I would like to acknowledge and thank my supervisors, Dr. Steven Hayward and Dr. Gavin Cawley for their support, advice and guidance. I would also like to thank Professor Akio Kitao and his lab group for hosting me for the molecular dynamics portion of this work.

I would like to thank the school of Computing Sciences for funding my studentship and the Japan Society for the Promotion of Science for funding my time at Tokyo Institute of Technology.

For supporting me for the last few years I would like to thank my kind and very patient partner Anthony, my family, particularly my mother, grandmother and brother, and my friends: Stacy, Dave, Alex, Ben, and Tarnia whose encouragement gave me the confidence to begin this path. Lastly I would like to thank Halo and Marley for keeping me company, despite their best efforts to prevent me from writing this thesis.

Abstract

Biomolecules such as proteins and nucleic acids are involved in all biological processes. As they take part in these processes, biomolecules often undergo motions and changes in their conformation that are related to their function. This thesis presents research into and development of methods to support the study of the dynamics of these changes.

Due to the scale of the structures and speed of the changes, common methods of determining (or "solving") the structures of biomolecules cannot capture the change in conformation. Detail of the changes must be extrapolated from changes observed between multiple solved states of the same structure. We present a novel method of visualising potential motions of atoms comprising biomolecules, estimated from solved structures at the start and end of the trajectory. Comparisons show that our method produces atomic coordinates that pass closer to known intermediates than those produced by similar existing methods.

Our visualisations treat each atom as an individual body, but the conformational changes of proteins can be broken down into the motions of "dynamic domains", which are sections of proteins that move semi-rigidly, controlled by flexible hinge bending regions. Tools such as the DynDom program identify and analyse the motions of these dynamic domains displayed between pairs of solved structures. We designed and developed DynDom6D, a new version of the DynDom program for very large macromolecules that assigns atoms to domains or hinge bending regions using 6-dimensional k-means clustering.

Several methods, including DynDom and DynDom6D, determine proteins' hinge bending regions from one or multiple solved structures. Previous attempts to identify the hinge

bending regions of a protein from its sequence of amino acids, without structural information, have proven less accurate. We used kernel logistic regression to train and test models on sequence information labelled by DynDom, with modest predictive results. We identified residues and patterns of amino acids that showed a significant propensity to be either present or absent in the hinge bending region.

Table of contents

List of figures

List of tables

Symbols

Acronyms / Abbreviations

AROC	Area under the Receiver Operating Characteristic
ATCase	Aspartate transcarbamoylase
cryo-EM	Cryo-electron microscopy
EM	Energy Minimisation
HI	Hinge Index
KLR	Kernel Logistic Regression
MD	Molecular Dynamics
MDS	Multi-Dimensional Scaling
NMA	Normal Mode Analysis
NMR	Nuclear magnetic resonance
PaCS-MD	Parallel Cascade Selection Molecular Dynamics
PD-1	Programmed death-1
RBF	Radial basis function

ROC	Receiver Operating Characteristic
SMACOF	Scaling by MAjorizing a COmplicated Function
SMD	Steered Molecular Dynamics
TMD	Targeted Molecular Dynamics

Chapter 1

Introduction

Biomolecules are an important part of biological processes of the cell. Proteins, formed of folded chains of amino acids, are biomolecules that act as natural molecular machines, physically and chemically controlling biological processes such as the Krebs cycle, and liver function. DNA and RNA are molecules comprised of nucleotides which encode and decode the sequence of amino acids that make up the proteins. As these biomolecules perform their tasks they change shape in ways that are directly linked to their function. For example, the protein aspartate transcarbamoylase (ATCase) is an enzyme that catalyses a reaction on a pathway which has the nucleotide CTP among its end products[57]. When the CTP structure binds to the ATCase, the enzyme undergoes a large conformational change which inhibits further catalysis. Understanding the mechanism of the conformational changes of biomolecules such as these grants us an understanding of biology at a molecular level.

Bioinformatics is the application of computer science to biology to provide tools and insight, often for problems where this would otherwise be difficult or impossible to obtain. One example of such a problem is the comparison of sequences of amino acids that make up the chain of a protein with others for the purposes of finding common patterns. To perform this alignment manually would be impractical and would require expert judgement, but

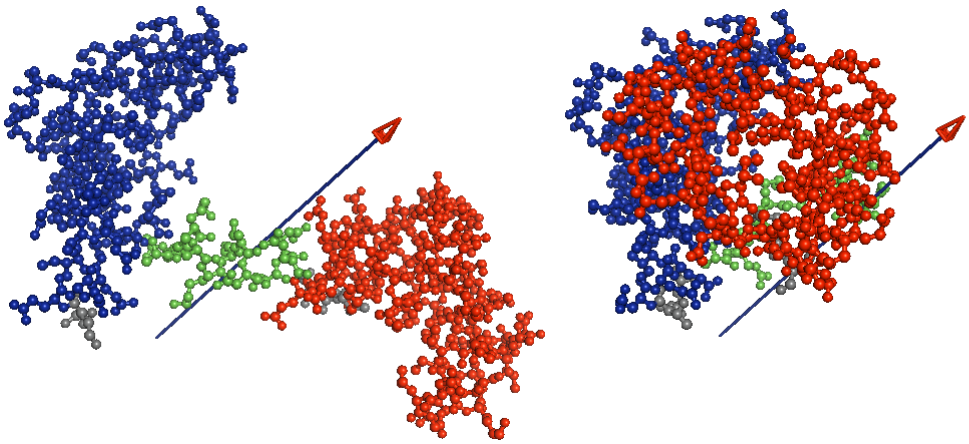


Fig. 1.1 The conformational change of calmodulin analysed by the DynDom program. The two dynamic domains are highlighted in red and blue, and the hinge-bending region is highlighted in green.

algorithms like the dynamic programming alignment methods Needleman-Wunsch [115] and Smith-Waterman [150] make this a fast and easy process.

Many individual proteins have been studied to determine their structure, sequence of amino acids, function, physical properties and family, building databases of experimentally derived knowledge. These provide potentially useful datasets for areas of computer science such as machine learning, where supervised classification and regression methods may learn from data labelled by observation to provide predictions for properties of proteins that have not yet been fully studied. This has had success in the prediction of secondary structure [134][165][46], disordered regions [100][108], and other structural properties.

Proteins undergo shape-changing movements during function. Sometimes, this conformational change can be described as the movements of "dynamic domains" [70], semi-rigid bodies that move relative to one another, facilitated by flexible hinge-bending regions that control the conformational change. These domains can be identified from experimentally-derived structures using methods such as HingeProt[44], FlexProt[145] and DynDom[95][69]. Figure 1.1 shows an example of calmodulin domains calculated by the DynDom program, where the green region acts as a hinge about which the red domain moves towards the blue domain.

This thesis presents the development of new bioinformatics methods and tools, many including implementations of machine learning algorithms, in order to derive new knowledge about the conformational changes that biomolecules undergo, and potentially to gain insight into mechanism and function.

1.1 Aims and objectives

This work describes the development of new methods and tools for the analysis of protein's structural changes.

- The first aim is to produce a new "protein morphing" method by which intermediate structures can be created that form a fluid transition from one solved structure to another. This allows the movements of specific atoms and amino acids to be noticed easier than by studying and comparing the static end-points. The method will be based on Multi-Dimensional Scaling (MDS), a family of techniques which attempt to construct a set of coordinates from a similarity or dissimilarity matrix such that similarity between items is represented by proximity in the created structure. In our method, the "dissimilarity" between atoms will be an ideal Euclidean inter-atomic distance created by interpolating the inter-atomic distances of atoms in the known structure.

  In working towards this aim, the thesis will attempt to achieve the following objectives:

  - The protein morphing method is intended to create an informative visualisation of a conformational change, so the animations created by viewing the frames sequentially should be smooth and easy to follow.
  - Conformational changes often accompany an interaction between biomolecules, but existing morph methods act on single structures. A novel version of morphing

will be developed that is tailored to the visualisation of the interaction between two biomolecules.

  - While the visualisation is unlikely to be a true representation of the exact movements of the atoms, the intermediate frames should be physically plausible with feasible bond lengths and torsion angles and no clashes between atoms.
  - It is possible to accurately predict the movements of atoms using Molecular Dynamics (MD), which simulates the physics underlying their behaviour. However, this method is extremely computationally expensive. As our method is more for visualisation rather than accurate prediction it should be considerably faster, preferably working within seconds or minutes.
  - It is assumed that the morphs will have more use as visualisations than as predictions of the movements of atoms, but the latter would give the morphing tool additional uses in studying proteins' movement. Where intermediate structures are known, the morphing tool would be particularly useful if it could be shown that its constructed intermediate frames pass as close as possible to the known intermediate conformation. A comparison with MD simulations may also indicate how well the visualisations reflect a plausible trajectory.

- The DynDom program for the identification of domains and hinge-bending regions described in the opening section operates on the backbone of a single chain of a protein, clustering segments of the backbone based on the similarity of the rotation they undergo. Macromolecules consisting of multiple chains may exhibit interesting domain movements to which several chains contribute, such as the ATCase example from the introduction. Advances in methods for solving proteins have provided high-resolution structures for larger macromolecules but these are not suitable for the original DynDom algorithm. A further method, DynDom3D[60], extended DynDom to identify domain movements in protein structures that contain multiple chains, but

required a computationally expensive connected-set algorithm that restricted the size of protein on which it could operate and used only 3 of the 6 degrees of freedom required to describe a rigid body movement to identify the domains. The second aim of this thesis is to develop a new DynDom program that can identify dynamic domains within large macromolecules by performing *k*-means clustering on all 6 dimensions required to describe the movement of a rigid body between two positions.

This aim will require the following objectives:

  - The results should be similar to the results of DynDom and DynDom3D where the comparison would be appropriate (that is, where the protein is a single chain or small enough to run in a reasonable time using DynDom3D).
  - Because the atoms are clustered on more information than previous iterations of the DynDom algorithm, the connected set algorithm may no longer be useful as additional features will separate distant sections that happen to have similar rotation vectors.
  - The method should perform well with very large macromolecules, and be able to accept mmCIF formatted files which are necessary for the latest large biomolecules.
  - The program should work when the input contains protein chains, DNA, RNA or other molecules, including a complex that contains a mix of protein and nucleic chains such as the ribosome.
  - DynDom identifies hinge-bending regions, but DynDom3D did not. DynDom6D should have a method for locating the hinge-bending regions as well as the rigid domain areas.

- Within the protein, domain movements are controlled by hinge-bending regions, which are short, flexible segments. Studies have shown that they are often positioned between

the termini of secondary structures[62]. A proposed predictor of the location of hinge-bending regions from sequence first predicts the protein's secondary structure from the provided sequence, and then predicts the location of the hinge-bending region given its predicted structural information [12]. There have been investigations into the composition of the sequences of amino acids that make up the hinge-bending region [52], but predictors based on this information do not show high accuracy. The final aim of this thesis is the application of kernel logistic regression (KLR) to this problem. KLR models the likelihood that a feature vector belongs to a given class; we will use it to predict the likelihood that a residue belongs to the hinge-bending region given which amino acid it is and its neighbours are. Sequence data and labelling of "hinge-bending region" and "not hinge-bending region" classes will be taken from the DynDom database.

This aim will require achievement of the following objectives:

- The predictor should be able to predict the location of hinge-bending regions using only the sequence of amino acids, with no additional information about the protein's solved structure.
- The predictor should have a high accuracy, as measured by training on one partition of our data and testing on a reserved set of non-redundant sequences.
- A benefit of the KLR algorithm is that trained models can be examined to identify which features had strong positive or negative discriminatory value. We will extract this information from successful models to look for patterns with potential biological insight into the composition of the hinge-bending region and its surrounding area.

1.2 Publications

The work included in this thesis was written about in the following publications:

- Veevers, Ruth, and Steven Hayward. "Morphing and docking visualisation of biomolecular structures using Multi-Dimensional Scaling." *Journal of Molecular Graphics and Modelling* 82 (2018): 108-116[160].
- Veevers, Ruth, and Steven Hayward. "Methodological improvements for the analysis of domain movements in large biomolecular complexes." *Biophysics and Physicobiology* 16 (2019): 328-336[161].
- Veevers, Ruth, Cawley, Gavin and Steven Hayward. "Investigation of sequence features of hinge-bending regions in proteins with domain movements using kernel logistic regression." *BMC Bioinformatics* 21 (2020): 1-18[159].

1.3 Thesis structure

Chapter 2 covers the background, motivation and relevant literature regarding the biological applications of the research. The properties and functions of biomolecules will be explored in more depth.

Chapter 3 is a review of literature relating to the computational processes required for understanding the research. The mathematical basis of the work, methods and comparable techniques will be described here. This chapter will describe existing methods of protein morphing, and explain the various MDS methods which our morphing method will use. A review of protein docking methods will provide background for our tailored docking morphs. An overview of MD will provide context for its comparison to morphing methods in later chapters. The machine learning methods used for domain identification and hinge-bending region prediction will be explored, with descriptions of the k-means clustering and KLR

algorithms. The tests that we will use to assess the performance of these methods will also be provided in Chapter 3.

The development of our MDS-based approach to protein morphing is described in **Chapter 4**. A test set of pairs of structures depicting protein domain movements was taken from the DynDom database, which covers a wide range of motions. These pairs were morphed using a number of versions of our method and assessed to find the MDS method that provided the best trade-off between speed and accuracy. The MolProbity[29] validation tool is used to compare how possible the constructed protein structures are in terms of the known limits of bond lengths and torsion angles, and timing experiments show the efficiency of the method and the improvement that can be gained from a GPU implementation. The methods were also compared against existing methods using a metric designed to measure how close a protein morph can come to a known intermediate structure. The extension of our method to allow all biomolecules and to provide visualisations of docking morphs is also included in Chapter 4. This technique is fast and while it performs well when there is a known intermediate structure, to know how possible it is to use the MDS morph as a prediction of the real path it should be compared to the true path of a conformational change. As recording is not possible, we turn to the gold standard, MD simulation. This chapter also describes the application of both the MDS morphing method and recent state-of-the-art MD technique PACS-MD to a test case, innovative and promising cancer drug nivolumab interacting with its target protein, programmed death-1 (PD-1). The relative timing of small internal movements are examined to compare the resulting paths.

Chapter 5 discusses the development of DynDom6D, a new generation of the domain-finding algorithm DynDom3D, which is aimed at macro-molecules for a problem domain where newly-solved proteins are increasingly huge machines with multiple moving parts. The previous versions of DynDom are described and details are provided for each step of the

DynDom6D process. Results are listed for four example biomolecules showing a range of domain movements.

Our investigation into applying Kernel Logistic Regression (KLR) to the prediction of hinge-bending regions from sequence data is described in **Chapter 6**. The chapter covers the creation of the dataset by filtering sequences from the DynDom database to find domain movement with strong hinge motions, and the effects of further filtering to reduce redundancy. The choice of window length to surround each residue is examined, and results are shown from a range of window lengths. A range of linear and non-linear kernels were used to map the input vector to feature space, and the results of varying this kernel are examined. Weights are extracted from the trained models and examined.

Chapter 7 summarises and discusses results from previous chapters, outlines potential directions for future work, and concludes the thesis.

Chapter 2

Biological Background

The following chapter describes the biological background and motivations underlying the subsequent work in this thesis. The composition and structure of proteins will be examined, as well as the way these structures can change shape in the course of a protein's function. This chapter will describe the physical constraints to which proteins must adhere as they go about these conformational changes, and which our method of visualisation (described in Chapter 4) should respect if it is to create plausible motions. It will also cover the ways in which these movements are described as the motions of semi-rigid bodies within the protein, driven by a flexible hinge-bending region, which will be detected in the method described in Chapter 5 and predicted in Chapter 6.

2.1 Protein Structure

Given that the mechanism by which a protein operates is known to be intimately linked with its structure, it is important to be able to formally describe these structures. Protein structure is relevant to the understanding of biological processes and for drug design[32]. There are four levels of structure which will be referenced in this work[122].

Amino acid name	Three-letter code	Single-letter code
Alanine	Ala	A
Cysteine	Cys	C
Aspartic acid	Asp	D
Glutamic acid	Glu	E
Phenylalanine	Phe	F
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Lysine	Lys	K
Leucine	Leu	L
Methionine	Met	M
Asparagine	Asn	N
Proline	Pro	P
Glutamine	Gln	Q
Arginine	Arg	R
Serine	Ser	S
Threonine	Thr	T
Valine	Val	V
Tryptophan	Trp	W
Tyrosine	Tyr	Y

Table 2.1 The names, three-letter codes and single-letter codes for each amino acid[122]

2.1.1 Primary Structure

The main chain of a protein is arranged as a "backbone" consisting of a repeating pattern of oxygen, carbon, and nitrogen atoms. The alpha carbon (or Cα) atom is located at the branch between the main chain and a side chain. The side chain, which typically takes one of twenty standard shapes, distinguishes the different amino acids, or residues, from which the protein is made. Table 2.1 lists the names and common abbreviations of the 20 amino acids. The body of this work will use the long names of amino acids, but some figures will use the single-letter code for clarity.

The atoms are bonded using several different interactions [122]. These bonds have standard ranges of distances depending on the type of bonds and the amino acids and atoms involved; studies of experimentally ascertained bond distances have created lists of expected

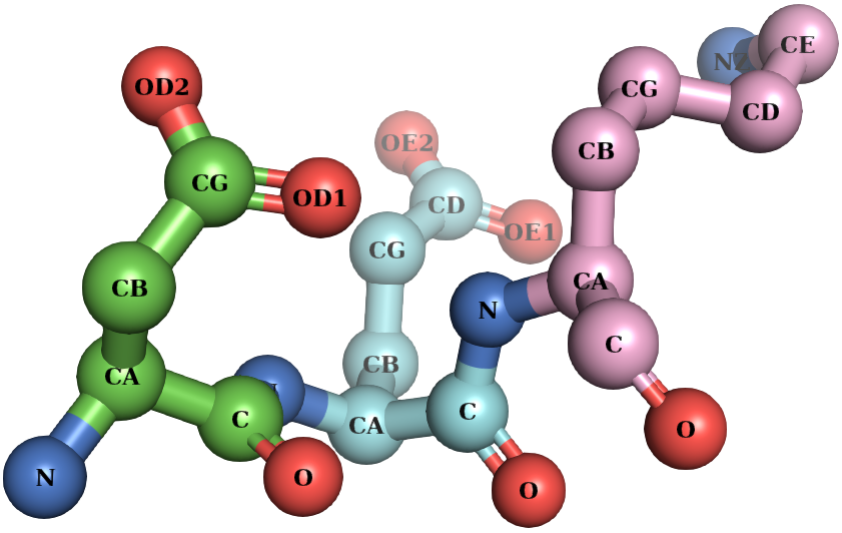


Fig. 2.1 A representation of a segment of protein chain consisting of three amino acids, labelled by atom name. Oxygen atoms are coloured red, nitrogen atoms are blue, and carbon atoms are coloured according to amino acid: carbon atoms are coloured green in the aspartic acid residue, cyan in the glutamic acid residue and pink in the lysine residue.

values [120][45]. The amino acids are joined with covalent "peptide" bonds, which have expected lengths of around 1.3Å[11]. Covalent disulphide bonds form bridges between a pair of cysteine amino acids[11] with a typical distance of 2.2Å. Van der Waals interactions are weak interactions that occur over distances of around 3.5Å, and hydrogen bonds are non-covalent interactions around 3Å[122].

Figure 2.1 shows an example of the arrangement of atoms in a segment of protein chain comprised of an aspartic acid, a glutamic acid and a lysine residue. Each atom is labelled with its name, which begins with its chemical element, sometimes followed by a code describing its position in the residue. The backbone runs along the bottom of the figure.

Each amino acid has a set of physical characteristics that determine its chemical and physical effect on the protein. A residue may be hydrophilic or hydrophobic, which would affect its behaviour in relation to the solvent surrounding the protein. These categories depend on the polarity and charge of the amino acid which dictates the interactions in which

it may take part. The size and shape of each residue will also influence its position in the protein.

The "primary structure" or "sequence" of a protein is a list of its amino acids in the order that they appear along the backbone. It is typically represented as a string of letters, where each letter represents an amino acid as in Table 2.1.

There are methods for predicting properties of the structure, function and family of a protein from its sequence of amino acids. Chapter 6 will describe an attempt to train models on sequence data to predict which parts will make up a flexible region that controls the protein's movements.

Primary sequences are stored in databases such as the UniProtKB database, which is annotated with functional information [17] [118].

2.1.2 Secondary Structure

Sections of proteins frequently consist of recurring structural "building blocks" that provide another way of describing a protein, referred to as the "secondary structure". These are formed by regular patterns of main chain hydrogen bonds, and are heavily influenced by the properties of the amino acids in the sequence.

Two of the most common types of secondary structure are the α-helix and the β-sheet. The α-helix is a cylindrical structure formed when the backbone of residue (at position n) is hydrogen bonded to the main chain of the amino acid that follows it four residues later in the chain ($n+4$). The pattern is repeated; the backbone of residue $n+1$ is hydrogen bonded to the main chain of residue $n+5$, continued until the end of the helix [122]. The β-sheet is formed when repeated hydrogen bonds along a segment of main chain connect it to the residues of another segment of the main chain that runs in either the same or the opposite direction to the original segment[122]. There are two types of β-sheet, parallel and anti-parallel, which refers to the relative directions of the strands of which it is composed.

The secondary structure is often represented as a sequence of characters of the same length as the primary sequence, where each character represents the type of secondary structure feature to which the corresponding residue belongs.

The prediction of secondary structure from sequence data alone is a common application of machine learning in the study of proteins. Previous techniques are covered in more detail in Chapter 3 as the problem involves similar experimental design as the prediction of the location of hinge-bending regions, which is the problem we approach in Chapter 5. These problems are potentially related, as hinge-bending regions have shown a preference to belonging to the termini of secondary structure features [62].

2.1.2.1 Torsion Angles

As well as the lengths of bonds, and the angles between bonds that share an atom, there are physical constraints on the torsion or dihedral angles that can be found along the backbone [88]. The ϕ-angle and ψ-angle describe the rotation of the Cα - N bond and the rotation of the Cα - C bond respectively[11]. These angles fall within allowed ranges, though the favoured ϕ and ψ angles are different in different secondary structures. These values are commonly plotted against one another in Ramachandran plots. Figure 2.2 shows an example Ramachandran plot where the ϕ-angles and ψ-angles for a solved calmodulin structure are shown. The allowed areas are outlined in orange, while the core regions for different secondary structures are outlined in red. The ω-angle is an additional torsion angle describing the rotation about the C - N bond, which is usually constrained at approximately 180°[122].

There are also constraints on the torsion angles between the backbone and the side chains, which fall into a range of values depending on which amino acid the side chain is[141].

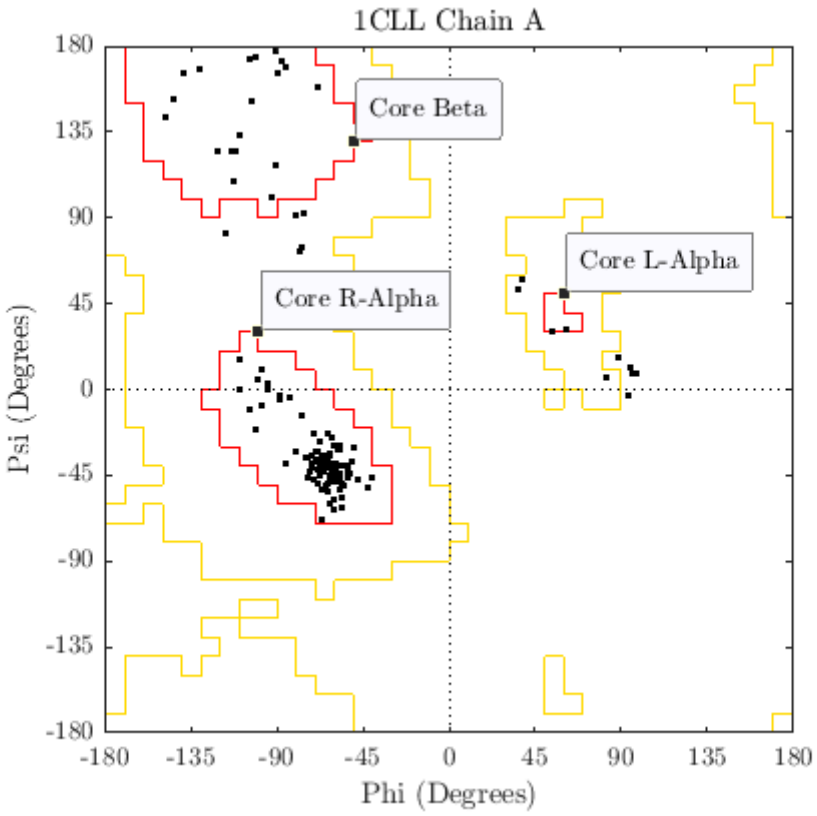


Fig. 2.2 Ramachandran plot for calmodulin (PDB code 1CLL)

2.1.3 Tertiary Structure

The tertiary structure of a protein is specified by the Cartesian coordinates of the atoms of which it consists. The tertiary structures can be stored in a file containing lists of the chains, residues and atoms which can be read and interpreted by molecular visualisation software such as Pymol [38] and Rasmol [140]. These files are stored in the Protein Data Bank [10], where they are assigned four-character codes. The tools developed using the methods we cover in Chapter 4 (MorphIt Pro) and Chapter 5 (DynDom6D) both accept tertiary structure files as input in the PDB format[9] which uses a column-based layout. Atoms are represented by a line of text, where the characters at set positions in a line correspond to a property of the atom. This format imposes a limit on the number of atoms and residues that the file can

contain, so the DynDom6D program also accepts mmCIF files, which are a dictionary-based format that was developed to store the details of macromolecules [16].

2.1.4 Quaternary Structure

Some macromolecules consist of multiple folded chains called "subunits". The position of chains within a macromolecule is referred to as its quaternary structure. The tool developed in Chapter 5 was designed to work on these large collections of chains.

2.2 Solving Protein Structure

The process of determining the tertiary structure of a given protein from some experimental data is known as "solving" it. Using x-ray crystallography, this experimental data is obtained through crystallising a protein and subjecting it to x-ray beams, and then studying the resulting diffraction patterns to get information about the crystal's contents, which are combined with pre-existing knowledge of its covalent structure to build a model of the structure[151][170]. Nuclear magnetic resonance (NMR) spectroscopy uses the magnetic properties of the protons and isotopes of nitrogen and carbon in the protein's atoms, which respond to different magnetic fields and eletromagnetic frequencies depending on their surroundings; by observing protons' reactions to variation in the magnetic field or applied electromagnetic frequencies deductions about those surroundings can be made[7][26]. Technological advances in the process of cryo-electron microscopy (cryo-EM), in which macromolecules are studied with electron microscopes after being frozen at very low temperatures, have resulted in the solution of structures of large macromolecules from small samples [5].

2.3 Conformational Changes in Proteins

As proteins perform their functions they undergo changes in shape, often in the course of an interaction with another molecule [61][6]. Understanding the mechanism by which they perform their function provides understanding of the biological processes to which they belong. The drug design process involves detailed study of the mechanism underlying biological processes in order to design drugs that can obstruct or assist the process. For example, research into the the interaction between the PD-1 protein and its PD-L1 or PD-L2 ligands has led to the development of targeted antibodies and small molecule drugs for the treatment of cancers[149]. This interaction triggers the inhibition of the immune system, preventing an immune response to tumours[30]. This interaction will be modelled in Chapter 4, so more information is provided below.

2.3.1 PD-1 and Nivolumab

The pathway by which PD-1 inhibits the immune system's response to tumours is a relatively recent discovery [80], which has introduced a new area for potential treatments of previously untreatable cancers[65]. The immune system eliminates disease within a host, but to do this it must distinguish what is part of the host itself in order to prevent damage[53]. The PD-1 protein forms a complex with its ligand, PD-L1, and this interaction has an inhibitory effect on the immune response[171]. The interaction between PD-1 and its other ligand, PD-L2, similarly inhibits the immune response though there are some differences in the mechanism of the response[174].

Positive expression of PD-1 and PD-L1 has been reported in patients with particular cancers, including advanced non-small cell lung cancers [65], particularly in current smokers and patients with certain mutations [40].

Drugs targeting the interaction between PD-1 and either PD-L1 [111] or PD-L2 [94] have been the focus of recent trials and studies. Five have been approved by the FDA for use in the USA[107]: pembrolizumab and nivolumab which target PD-1 itself, and atezolizumab, durvalumab and avelumab which target PD-L1. Research into small-body molecules is also ongoing [149][171]. While studies into the performance and safety of PD-1 targeting monoclonal antibodies is still ongoing, results indicate that they are potentially effective at combating cancers that were previously resistant to treatment such as advanced squamoous-cell carcinomas [109], advanced kidney cancers, and advanced non-small cell lung cancers.

Structures of human and murine PD-1 and its ligands have been solved; the human and murine ligands have been shown to interact differently with the human PD-1[162]. Investigations into the motion of the binding have found that two flexible loops in the PD-1 are used in binding with PD-L1 [146].

2.3.2 Domains

There are multiple definitions of "domains"[125] in proteins; the division can be based on structural features, evolutionary homology, function or dynamics. Domains form the basic structural element used in many of the databases of proteins' sequences and structures. The Pfam database contains sequences organised into "families", clusters of domains with similar sequences, classified by using hidden Markov models[152][43]. CATH is a database of structural families hierarchically grouped by shared structural properties and features [119][121]. SCOP[112] and SCOP2[3] are also structural databases grouped by similar structures and evolutionary homology, where entries are initially manually classified[148];

The following chapters will focus on dynamic domains, which are domains defined from conformational change. Dynamic domain information is stored in the DynDom database[95][69]. Residues in a dynamic domain will move rigidly or semi-rigidly rel-

ative to another domain. Domain movements have been classified according to this relative movement as "predominantly hinge", "predominantly shear", or not predominantly either [59], where hinge motions are also described as interface-creating, and shear motions as interface-preserving [154]. Hinge motions are a large movement about a flexible hinge, which brings the domains into contact, whereas shear motions are made of multiple smaller motions. These flexible hinge-bending regions control the conformational change; by maintaining rigid domains and allowing hinge-bending residues to flex, conformational changes can be reproduced [71].

In the technical background in Chapter 3, Algorithms for determining these dynamic domains will be covered further (Section 3.8). Chapter 5 will describe a method for identifying dynamic domains and hinge regions from a pair of structures, and Chapter 6 will cover the use of machine learning to predict the location of the hinge-bending region given only sequence data.

Chapter 3

Technical Background

The chapter provides background knowledge and a review of the literature regarding the key methods of the following chapters. Section 3.1 introduces a method for bringing pairs of sequences into alignment, which will be used in later methods to find corresponding atoms in biomolecular structures. Section 3.2 describes previous approaches to the problem of protein morphing. The development of our docking morph technique was inspired by methods of predicting the docked conformation of unbound proteins using methods described in Section 3.3. The gold standard of protein morphing is achieved by Molecular Dynamics, which is covered in Section 3.6. The next two sections relate to methods we use in our own protein morphing approach: Section 3.5 provides background on the linear algebra used in this and later methods; Section 3.6 is a detailed look at multi-dimensional scaling and its variants. Section 3.7 covers the two machine learning algorithms that are used in the following chapters. In Section 3.7.1, we examine the *k*-means clustering algorithm that will be used to identify dynamic domains in Chapter 5, while in 3.7.2 kernel logistic regression is explained. In Chapter 6 we will use kernel logistic regression to predict the location of hinge-bending regions from primary sequence alone; Section 3.8 discusses previous attempts to predict the hinge-bending region and Section 3.9 covers the specific case of using a sliding window to generate input for a machine learning model.

3.1 Sequence Alignment

There are procedures which require every residue or atom of a pair of proteins to be in alignment, so that they might be compared or processed. The morphing method in Chapter 4 requires a one-to-one correspondence of atoms in a pair of structures so that it can produce a visualisation of how each atom moves between the two states. The domain identification tool in Chapter 5 also requires two aligned structures so that it can calculate the movement required to take small subsets of atoms from their positions in one structure to their positions in the other. Sequences may be entirely distinct or almost identical, or they may contain some similar and some different segments, or have insertions and deletions such that a large part of two very closely related sequences may be unaligned if they were to be compared residue by residue.

Both Chapter 4 and Chapter 5 use approaches to sequence alignment that are based in dynamic programming. Dynamic programming uses an efficient matrix-based algorithm to find the optimum alignment between two sequences. There are various dynamic programming algorithms, but the method used in the following chapters is the Needleman-Wunsch [115] algorithm.

The alignment depends on the construction of a dynamic programming matrix, an example of which is shown in Table 3.1, which shows the alignment of two short dummy sequences. One sequence is place into the top row, and the other into the leftmost column. The first cell inside these headers is given a value of zero, and the algorithm constructs the cell from top-to-bottom and left-to-right using this zero as a starting point. When considering which value to put in the cell in row i and column j, $\mathbf{D}_{i,j}$, the three cells immediately above ($\mathbf{D}_{i-1,j}$), to the left ($\mathbf{D}_{i,j-1}$), and diagonally to the top-left ($\mathbf{D}_{i-1,j-1}$) are considered, and a score is calculated for each. The maximum of these three scores is then stored in the cell:

		L	**Q**	**E**	**L**	**L**	**Q**	**K**	**A**
	0	**-8**	-16	-24	-32	-40	-48	-56	-64
Q	-8	-2	**-3**	?					
D	-16								
L	-24								
Q	-32								
L	-40								
A	-48								
K	-56								
T	-64								

Table 3.1 An example of the dynamic programming matrix for an alignment of a pair of short sequences (sequences created for demonstration purposes)

$$\mathbf{D}_{i,j} = \max \begin{cases} \mathbf{D}_{i-1,j} + g \\ \mathbf{D}_{i-1,j-1} + \mathbf{M}_{x,y} \\ \mathbf{D}_{i,j-1} + g \end{cases} \tag{3.1}$$

The score from the top cell represents a deletion or insertion, and is calculated as the value of the cell above minus a gap penalty, g. The score from the left cell similarly represents a deletion or insertion from the other sequence, and is the value currently in the left cell minus the gap penalty. The score from the top-left cell represents an alignment between the i^{th} amino acid in the first sequence (x) and the j^{th} amino acid in second sequence (y), and is calculated as the sum of the value in the top-left cell and the match score between x and y looked up in a substitution matrix, $\mathbf{M}$. The methods in the following work use the BLOSUM62 [74] substitution matrix.

For an example, the shaded cell will consider three values. The score from the left will be -3 (from the cell directly to the left) minus the gap penalty score (8 in this example).

$$\text{Score}_{\text{left}} = -3 - 8 = -11 \tag{3.2}$$

The score from above is calculated in a similar way, subtracting the gap penalty from the value in the cell above:

$$\text{Score}_{\text{top}} = -24 - 8 = -32. \tag{3.3}$$

The score from the top-left will take the score from the diagonal cell and add the score given by BLOSUM62 for the match between a glutamine (Q) and a glutamic acid (E), 2:

$$\text{Score}_{\text{top-left}} = -16 + 2 = -14. \tag{3.4}$$

The maximum value of the three scores is given by the left, so -11 is placed in the cell. A record of the direction that give the cell its value is also stored. When the matrix has been completed, the matrix will be traversed using a pointer that starts in the bottom-right corner. The direction stored for the cell is retrieved, and the aligned sequence is built from the end to the beginning. If the score came from the left, a gap is inserted into the sequence on the left, and if the score came from above then a gap is inserted into the sequence on the top. If the score came from the top-left then the two corresponding residues are aligned. The pointer then moves following the stored direction.

3.1.1 Sequence Identity and Homology

Finding an optimal alignment between two proteins allows for a measure of similarity. Similar sequences are likely to belong to proteins with similar structure and functions, so it is useful to have a measure of how alike two proteins' sequences are. Given a pair of sequences aligned as best as possible (see Section 3.1), with n_{match} matching amino acids in aligned length n.

$$\text{SeqID} = \frac{n_{\text{match}}}{n} \times 100 \tag{3.5}$$

A pair of sequences with high sequence identity is likely to be homologous, having the same shape; Sander and Schneider[138] plots the relationship between sequence identity and structural similarity for sequences and structures from the EMBL/Swissprot and Protein Data Bank databases as a function of sequence length, showing that the threshold of sequence identity at which a pair of proteins are likely to be homologous decreases as the length increases. Rost[133] examines pairs whose sequence identity falls within the "twilight zone" in which proteins may be homologous but also difficult to align. While these findings also note that shorter sequences require a higher cut-off, Rost[133] places a threshold around 30%, above which pairs of proteins are very likely to be homologous, and another around 25%, below which pairs are very unlikely to be homologous. This is relevant when attempting to learn from sequence data as in Chapter 6.

3.2 History of Morphing Techniques

Section 2.2 discussed how static conformations of proteins can be determined through NMR spectroscopy or X-ray crystallography. Conformational changes accompany protein function, such as receptors changing shape during ligand binding[131] and changes between active and inactive enzyme states[143]. The solved structures known for such a protein will be snapshots of specific states, usually either the bound or the unbound state. Understanding the changes undergone during the course of a protein's function assists in understanding the mechanism by which the protein contributes to important biological processes. This understanding will benefit drug design, where a full view of a protein's mechanism may give insight into how it may best be targeted, or how a new drug may compete with a native ligand to disrupt a disease-causing interaction.

Experimental NMR techniques can monitor a number of properties that inform on the likely dynamics of the subject protein [81] but the complete structural pathway of a protein as it changes conformation can not yet be recorded. A simple before and after comparison of

the location of atoms may provide the viewer with a general idea of what a path between the two states may have looked like, but this requires an understanding of correspondence between atoms which may not be apparent, and tracking a single point from one structure to the next which may be difficult to parse, particularly in a large motion. The problem of constructing a potential pathway between known conformations is referred to as "protein morphing".

Computational protein morphing techniques require only the two known structures to construct a path between them which can assist in visualising the changes and obtaining potential intermediate structures. Presenting these paths as an animation aids the viewer in understanding the motion as individual atoms and residues can be followed as they move along their path.

The underlying forces that drive the changes can be modelled using Molecular Dynamics simulation packages such as GROMACS[1] and AMBER[25]. This process is described in more detail in Section 3.4. These results are physically plausible as they provide intermediate structures derived from a physically accurate energy function. However, the dynamics are so computationally expensive that MD is only a viable option for long-term in-depth research into a protein; it can take weeks or months to generate output. It also relies on pre-processing of the proteins to ensure that they have no missing sections. Additionally, rare events may not occur during the time frame of the simulation, so other techniques are required to induce the desired behaviour. To get fast results, it is necessary to use simpler techniques.

Simple Cartesian linear interpolation techniques such as the early methods used by Vonrhein et al.[163] incrementally move each atom of the protein from their start position to their end position. This grants a quick result but for many motions the intermediary structures are infeasible; bond lengths and angles are not considered and so atoms can overlap, causing clashes which would make the resulting energy improbably high. For example, a phenylalanine residue contains two side-chain single bonds leading to a benzyl

ring. The ring must remain planar but it may "flip over" between solved structures, rotating 180° about one of the single bonds. This would contain two pairs of atoms that appear to change places between the two structures. In a linear interpolation the atoms would attempt to pass through one another on their way to the opposite side of the residue, whereas a more realistic morph would show this as a rotation which maintains fixed inter-atomic distances within the residue.

Many methods are available; methods commonly involve elements of Cartesian interpolation, inter-atomic interpolation and energetics.

The MolMovDB web server, developed by Krebs and Gerstein [91], has been described as "probably the most popular online method"[167] of protein morphing. The process begins with a linear interpolation, but at each step the protein's atom locations are adjusted to minimise its energy. In their approach, one domain is held fixed, whilst the other moves.

The method proposed by Kim et al. [87] begins with an interpolation not of the Cartesian co-ordinates but of the inter-atomic distances between nearby Cα atoms in the two solved states. A cost function is minimised at each frame of the interpolation in order to construct a 3-dimensional structure that best fits these interpolated values. The cost function consists of the sum of squared distances between a linear interpolation of the start and end co-ordinates and the solved structures' Cα atoms with some displacement applied. Their technique for this minimisation is described as a form of Elastic Network Model where the distances between pairs of atoms are modelled as "springs". This method has been made available on the NOMAD-REF server. Similarly, MinActionPath[55] is an energetic Cα ENM-based approach that aims to construct a trajectory between conformations that minimizes the Onsager and Machlup action.

The CLIMBER[168] technique incrementally advances atoms towards their destination using a method that combines inter-atomic distance interpolation and energy minimisation. At each step, a structure is found by minimising the internal energy, with added restraints

based on gradually removing the difference between Cα distances in the current structure and the target structure.

The FATCAT [173] server contains, among other tools, FATCAT-pairwise, a morphing tool which brings the pair of structures into structural alignment and attempts to morph between the conformations in terms of the fewest possible rigid body rotations about flexible hinge regions. This provides a flexible superposition of the two structures as well as a visualisation of the path taken to create the superposition. As-Rigid-As-Possible (ARAP)[117] is another method which specifically attempts to create rigid motions between solved structures. It uses mesh distortion techniques originating from computer graphics. A mesh topology is created using information about which atoms are bonded; after interpolating the Cartesian coordinates of a single atom, the rest of the atoms are updated to minimise cell energy.

3.3 Protein Docking Visualisations

Conformational changes frequently accompany the process of protein interaction during the formation of complexes, wherein one protein (or other biomolecule), referred to as a ligand, is bound to a larger biomolecule known as a receptor. In these cases it may be informative to visualise the intra-molecular changes of each constituent in relation to the other. Unlike protein morphing, there are few available methods to perform this visualisation. Protein visualisation and animation software that currently exists such as UCSF Chimera [123] typically uses a rigid-body linear interpolation of start and end positions provided by the users, with any morphing happening in a separate process. The web server MovieMaker [105] automates the position and trajectory of the constituents from an input docked complex, but operates entirely rigidly, offering no input for the undocked constituents. To find relevant literature and explore the requirements of such a program, we looked to the related problem of docking.

Docking is the problem of predicting how two biomolecules will form a complex, given their unbound states. There are many protein docking methods available and in development. As with protein morphing, Molecular Dynamics (Section 3.4) provides a very realistic depiction of the interaction between biomolecules, provided they can be induced to perform the docking.

Frequently docking techniques contain two common stages: in the first, many (usually tens of thousands) potential docked complexes are generated, and in the second a chosen scoring system is applied to attempt to identify native or near-native structures [132][78]. Methods of the search for candidate complexes vary, but typically involve rigid-body scans wherein a moving ligand molecule is rotated and translated about the stationary receptor molecule. There are very many potential positions so docking methods accelerate this scan, often using similarity to known "templates" obtained from solved complexes [127], for example ClusPro [90] uses a Fast Fourier Transform approach. At this stage, some information-driven docking techniques such as HADDOCK [84][158] integrate data regarding shapes, orientations, interface restraints and other properties which are observed experimentally or predicted using bioinformatics methods.

While treating the constituents as rigid bodies is computationally convenient, flexibility of both backbone and side chain atoms has been shown to be important to the success of docking even when the conformational changes undergone are very small [127][42]. Therefore docking techniques attempt to incorporate flexibility in various ways [14]. For some techniques, such as ICM-DISCO[49] and ProPOSE[76], flexibility is applied to the side-chain or backbone atoms during a further refinement step after allowing some clashes during the rigid-body scan, sometimes referred to as “soft-docking”. Other methods incorporate flexibility by repeating the scan using various pre-generated conformations, referred to as “ensemble” docking[2], or by explicitly allowing flexibility at various stages during the search for candidates (as in RosettaDock’s fixed-backbone optimisation [142]).

However it is introduced to a docking methods' constituents, it is apparent that flexibility is a goal for recent docking prediction techniques. Candidate structures found by docking methods can therefore vary not only in the relative position of the ligand and receptor but also in their conformations, making a visualisation that clarifies the relation between atoms in the start and end structures particularly useful.

3.4 Molecular Dynamics

Molecular Dynamics is a computational technique for simulating the movements and interactions of molecules[85]. Trajectories plotting the motions of the atoms that make up a molecule are created by modelling approximations of their underlying physics[41]. The accuracy of the structural information that can be obtained using MD simulations makes them suitable for use in drug design, in the modelling of potential drugs, targets, and their interactions[37].

The simulations are typically created from experimental data and simulate a given period of time, split into discrete steps[137]. At each step, the forces currently acting on each atom are calculated using equations that approximate the effects of the inter-atomic interactions (section 2.1.1)[137]. Some *ab inito* MD methods calculate electron behaviour, but these calculations are computationally expensive[157]. This makes makes *ab initio* MD most suited for specific problems that require details on the making and breaking of bonds. More commonly, the atoms are considered to be solid bodies, with no modelling of the motion of individual electrons or other subatomic particles[41]. The effects of the electrons are modelled implicitly using force fields, which are sets of parameters and functions defining how different atoms will interact[85]. Each atom's position in the next step is then calculated from the approximated forces.

As well as the molecule itself and its constituent atoms, the environment surrounding it must also be modelled. The solvent surrounding the molecule, and the temperature of the solvent, have been shown to have a strong impact on the resulting simulation[85].

3.4.1 PaCS-MD

The method of classical MD as described requires a set time frame. In cases where MD is used to examine or simulate a specific conformational change or interaction, there is no guarantee that the desired behaviour will occur fully or at all within the set time frame, particularly when the desired event is rare. Initial attempts to overcome this involve applying external factors within the simulation: Steered MD (SMD) applies a force to an atom or atoms to induce a movement, and Targeted MD (TMD) applies holonomic (relating only to the time and coordinates) constraints based on the target structure. These approaches have drawbacks resulting from the external influences on the system, as both can result in large, unnatural distortions[67].

PaCS-MD [67] creates a transitional pathway between an initial and target structure without the application of external forces or constraints using cycles of parallel simulations. An initial short simulation is performed from the target structure, and then each step of the resulting trajectory is ranked against the target. The PaCS-MD paper [67] describes a ranking based on RMSD to a given target structure, but we use minimum distance between atoms of two docking molecules. The top ranged structures are then selected as the starting positions for a short cycle of multiple independent molecular dynamics. The trajectories of each independent run are also ranked according to the same ranking metric, top ranked structures again forming the initial structures of the next cycle. This continues until the RMSD to the target structure reaches an acceptable value. Trajectories between the initial structure and the final structure can be concatenated, creating one single path that the PaCS paper refers to as the "reactive" trajectory.

3.5 Linear Algebra

Methods in the following sections will rely on linear algebra; this section will briefly outline properties and methods that will be important for future methods.

3.5.1 Vectors

The manipulations of vectors will be relevant for methods used in DynDom6D and our hinge prediction investigations, where position vectors are used for the calculation of motion and feature vectors are used in the machine learning processes. These will involve the calculation of the inner product of two vectors, **a** and **b**, in a vector space. The inner product is a function to multiply two vectors that yields a scalar that we will denote $< \mathbf{a}, \mathbf{b} >$[166]. The standard inner product when **a** and **b** contain n real numbers is[18]:

$$< \mathbf{a}, \mathbf{b} >= \mathbf{a}^T \mathbf{b} = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i. \tag{3.6}$$

The cross product is a product of two three-dimensional vectors, $a = a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k}$ and $b = b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}$, which yields another vector that is perpendicular to both[147]:

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} \tag{3.7}$$

The magnitude, or Euclidean norm, of n-dimensional vector **a** can be calculated from[18]:

$$\|\mathbf{a}\| = \sqrt{\sum_{i=1}^{n} a_i^2} \tag{3.8}$$

3.5.2 Matrices

Our morphing method will require a square matrix, $\mathbf{A}$, with elements:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{3.9}$$

The transpose of $\mathbf{A}$ is a matrix where the elements of the rows and columns have [changed place]:

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \tag{3.10}$$

When $\mathbf{A}$ is equal to $\mathbf{A}^T$, the matrix is symmetric[156].

Some of our methods require the calculation of eigenvalues and eigenvectors, paired scalar λ and vector $\mathbf{v}$ values such that[129]:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \tag{3.11}$$

There are many methods for calculating eigenvalues and their corresponding eigenvectors given the matrix $\mathbf{A}$; we use the LAPACK implementation of the QR method integrated into MATLAB [63]. The version of the QR method implemented in LAPACK is more complicated, with shifts and matrix reductions selected to better find solutions for input matrices $\mathbf{A}$ with different properties[1], but a simple version of the QR method is as follows, as described in [129]. An orthogonal matrix $\mathbf{Q}$ is initialised as the identity matrix, and a matrix $\mathbf{T}$ is initialised as a copy of $\mathbf{A}$. An iterative process follows, where at step k, $\mathbf{T}^k$ is factorized into $\mathbf{Q}^k$ and a triangular matrix $\mathbf{R}^k$ such that:

[1] https://uk.mathworks.com/company/newsletters/articles/variants-of-the-qr-algorithm.html

$$\mathbf{T}^k - \mu\mathbf{I} = \mathbf{Q}^k\mathbf{R}^k, \tag{3.12}$$

where μ is a shift value and $\mathbf{I}$ is the identity matrix. The value of $\mathbf{T}^k$ is changed:

$$\mathbf{T}^k = \mathbf{R}^k\mathbf{Q}^k + \mu\mathbf{I}. \tag{3.13}$$

On convergence, the diagonal of $\mathbf{T}$ will contain the eigenvalues of $\mathbf{A}$ if real eigenvalues are possible[129].

A matrix of real numbers is described as positive definite if it adheres to the formula

$$(\mathbf{Ax}, \mathbf{x}) > 0 \; \forall \mathbf{x} \; \in \mathbb{R}^n, \; \mathbf{x} \neq 0 \tag{3.14}$$

or semi-definite if it instead adheres to the formula[129]:

$$(\mathbf{Ax}, \mathbf{x}) \geq 0 \; \forall \mathbf{x} \; \in \mathbb{R}^n, \; \mathbf{x} \neq 0 \tag{3.15}$$

3.6 MDS

Multi-Dimensional Scaling is the name for a family of techniques intended to construct co-ordinates for a set of objects, where similarity between the objects is represented as the proximity between their associated points in a space. Performing any MDS method requires a set of n objects and an $n \times n$ matrix D, where each element D_{ij} represents the dissimilarity between objects i and j. This similarity can be a real measure or perceived value; MDS has been used in psychological fields to plot relative judgements about topics. In the study of proteins, MDS has been used to reconstruct the physical structure of a protein given information about which residues are in contact[124].

3.6.1 Classical MDS

Classical MDS uses eigenvalue decomposition to construct the $3 \times n$ coordinate matrix $\mathbf{X}$ from the distance matrix $\mathbf{D}$[15]. Cox and Cox describe the classical MDS method as follows[33].

The inner product matrix B is first determined from the distance matrix through double centering.

$$\mathbf{A} = -\frac{1}{2}\mathbf{D}_t^2 \tag{3.16}$$

$$\mathbf{H} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T \tag{3.17}$$

$$\mathbf{B} = \mathbf{HAH} \tag{3.18}$$

Here, $\mathbf{I}$ is the identity matrix of size $n \times n$, and $\mathbf{1}$ is an $n \times n$ matrix where every element is 1.

The coordinate matrix is then derived from the eigenvalues of the inner product matrix.

$$\mathbf{B} = \mathbf{X}\mathbf{X}^T \tag{3.19}$$

From writing B as a product of its spectral decomposition:

$$\mathbf{B} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \tag{3.20}$$

The n-dimensional coordinate matrix $\mathbf{X}$ can be obtained from Equation 3.19:

$$\mathbf{X} = \mathbf{V}_1\Lambda_1^{\frac{1}{2}} \tag{3.21}$$

Where Λ_1 is the diagonal matrix of the first n eigenvalues and $\mathbf{V}_1$ is the first n eigenvectors.

We see how well the constructed coordinates fit with our ideal distances by calculating the strain, which is calculated by the sum of the first n eigenvalues divided by the sum of all non-zero eigenvalues. Where the strain is close to 0 the co-ordinates adhere closely to the ideal distances in **D**, but when the strain is non-zero, the distances between points in the co-ordinates contain more discrepancies with the target distances.

3.6.2 SMACOF MDS

Classical MDS is a reasonably fast method, but further MDS techniques been developed that better reproduce the ideal distances. These modern methods incrementally adjust a starting set of co-ordinates to minimise an error criterion, commonly the stress (σ) function. These coordinates are commonly the results of classical MDS, but can be obtained by any other means such as randomisation. Different choices of the stress function give different forms of modern MDS, which can be either metric MDS, which attempts to reconstruct the exact dissimilarities, or non-metric MDS, which focuses only on reconstructing the ranking of the dissimilarities. A common metric MDS method uses a simple weighted sum of squares stress function of the form:

$$\sigma_r(\mathbf{X}) = \sum_{i<j}^{N} w_{ij}(d_{ij}(\mathbf{X}) - \delta_{ij})^2 \tag{3.22}$$

Where δ_{ij} is the ideal distance between atoms i and j, d_{ij} is the distance between i and j in the constructed coordinates, and w_{ij} is the weight to apply to each pairwise distance. Weighting is frequently used to emphasise or negate the impact of pairings that have not been measured, for example, if the similarity between a pair of objects has not been measured.

The Scaling by MAjorizing a COmplicated Function (SMACOF) method, using de Leeuw's iterative majorization process, derives a monotonic function where the coordinate matrix **X** is repeatedly updated using a majorization process, until the resulting stress is sufficiently small or improves by a sufficiently small amount. The majorization process is a

means of finding the minimum of a complicated function by iteratively minimising a simpler one[33]. The following explanations are based on the descriptions in De Leeuw[34][35] and Cox and Cox[33].

The stress function in Equation 3.22 is expanded to multiply out the brackets:

$$\sigma_r(\mathbf{X}) = \sum_{i<j} w_{ij}\delta_{ij}^2 + \sum_{i<j} w_{ij}d_{ij}^2 - 2\sum_{i<j} w_{ij}\delta_{ij}d_{ij}(\mathbf{X}). \tag{3.23}$$

This is rearranged to highlight the three sum of squared distance terms:

$$\sigma(\mathbf{X}) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\delta_{ij}^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}d_{ij}^2(\mathbf{X}) - 2\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\delta_{ij}d_{ij}(\mathbf{X}), \tag{3.24}$$

The terms are given the notation:

$$\sigma(\mathbf{X}) = \eta_\delta^2 + \eta^2(\mathbf{X}) - 2\rho(\mathbf{X}). \tag{3.25}$$

The η_δ^2 term, obtained from the distance matrix and weighted matrix, will remain constant, as the initial similarity matrix and weighting will not change during the MDS process.

The second term, $\eta^2(\mathbf{X})$ can be rewritten as:

$$\eta^2(\mathbf{X}) = \mathbf{tr}\ \mathbf{X}^{\mathrm{T}}\mathbf{V}\mathbf{X} \tag{3.26}$$

The **tr** operation, called the "trace", is the sum of the elements on the diagonal of a matrix. **V** is defined as the following, where **A** is the row- and column-centred matrix:

$$\mathbf{V} = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\mathbf{A}_{ij}, \tag{3.27}$$

The final term, $\rho(\mathbf{X})$, can also be written:

$$\rho(\mathbf{X}) = \mathbf{tr}\ \mathbf{X}^T\mathbf{B}(\mathbf{X})\mathbf{X}, \tag{3.28}$$

where:

$$\mathbf{B}(\mathbf{X}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} s_{ij}(\mathbf{X}) \mathbf{A}_{ij}, \tag{3.29}$$

$$s_{ij}(\mathbf{X}) = \begin{cases} \frac{\delta_{ij}}{d_{ij}}, & \text{if } d_{ij}(\mathbf{X}) > 0 \\ 0, & \text{if } d_{ij}(\mathbf{X}) = 0. \end{cases} \tag{3.30}$$

When majorizing a complicated function, $f(x)$, a majorizing function $g(x,y)$ is chosen such that for all values of x, $g(x,y) \geq f(x)$, and $g(y,y) = f(y)$. At each step, i, of the iteration, y_{i+1} is set to the minimum of $g(x,y_i)$. This continues until convergence; the minimum of $g(x,y)$ at convergence is taken as the minimum of $f(x)$.

The SMACOF algorithm is minimizing the stress function, written using the above terms as:

$$\sigma(\mathbf{X}) = \eta_\delta^2 + \mathbf{tr}\ \mathbf{X}^{\mathrm{T}} \mathbf{V} \mathbf{X} - 2\mathbf{tr}\ \mathbf{X}^{\mathrm{T}} \mathbf{B}(\mathbf{X}) \mathbf{X}, \tag{3.31}$$

using the majorization function:

$$T(\mathbf{X}, \mathbf{Y}) = \eta_\delta^2 + \mathbf{tr}\ \mathbf{X}^{\mathrm{T}} \mathbf{V} \mathbf{X} - 2\mathbf{tr}\ \mathbf{X}^{\mathrm{T}} \mathbf{B}(\mathbf{Y}) \mathbf{Y}. \tag{3.32}$$

The minimum of $T(\mathbf{X}, \mathbf{Y})$ lies where:

$$\frac{\delta T}{\delta \mathbf{Y}} = 2\mathbf{V}\mathbf{X} - 2\mathbf{B}(\mathbf{Y})\mathbf{Y} = 0. \tag{3.33}$$

Using the Moore-Penrose inversion of $\mathbf{V}$, $\mathbf{V}^+$, the updated coordinate matrix $\mathbf{X}$ is calculated at each iteration from the previous coordinate matrix $\mathbf{Y}$ using the Guttman transform:

$$\mathbf{X}_u = \mathbf{V}^+ \mathbf{B}(\mathbf{Y}) \mathbf{Y}. \tag{3.34}$$

3.6.3 Multi-Grid Acceleration

The SMACOF method offers improved accuracy over classical MDS, but it does so at the expense of more computational complexity which leads to longer runtime. Multi-grid acceleration offers a method of improvement for the runtime of SMACOF MDS[21]. The multi-grid acceleration method achieves this by SMACOF-like optimisation of coarser representations of the set of objects.

From a set of all objects $\mathbf{A}$, a hierarchy is constructed where each level of the hierarchy is increasingly coarser. Level l_1 contains all objects in $\mathbf{A}$, layer l_2 is a subset of l_1, layer l_3 is a subset of l_2, and so on. Each object in layer l_r that is not represented in l_{r+1} will be assigned an object in l_{r+1} that "stands in" for it. These relationships will define how coarse results are extrapolated onto finer layers and how results from high-resolution levels are applied to coarser levels. They are established in a pair of matrices for each level referred to as the restriction operator and interpolation operator[22]. If level l_r contains n objects and level l_{r+1} contains m objects, the restriction operator $\mathbf{P}_r^{r+1}$ is an $n+1 \times m$ sparse matrix that defines which object in the next coarsest level that stands in for each object. The element (i, j) in $\mathbf{P}_r^{r+1}$ is 1 if the i^{th} object in l_{r+1} stands in for the j^{th} object in level l_r, and 0 otherwise. The interpolation operator $\mathbf{P}_{r+1}^{r}$, which defines how l_{r+1} is interpolated into l_r, is typically the transpose of $\mathbf{P}_r^{r+1}$.

The multi-grid MDS solution is found by applying a V-cycle algorithm to initial sets of coordinates representing the objects in each level, $\mathbf{X}_1$. At level l_r, the coordinate matrix $\mathbf{X}_r$ is optimized K times to minimize stress producing $\mathbf{X}_r^{'}$. The full solution, where the gradient of the stress is 0, is sought using the approximate solutions, where the gradient of the stress is a residual value T, on coarser sets of objects which are computationally less expensive to find[21]. This requires a new stress function which is a variation on the sum of squared errors stress in Equation 3.22

$$\sigma_r(\mathbf{X}) = \sum_{i<j}^{N} w_{ij}(d_{ij}(\mathbf{X}) - \delta_{ij})^2 + \lambda \sum_{j=1}^{m} \Big(\sum_{i=1}^{N} x_{ij} \Big)^2. \tag{3.35}$$

Bronstein et al.[22] define a single V-cycle recursively: if the current level has the lowest definition, then the minimization is performed using iterations of a SMACOF-like minimisation and returned. Otherwise, the stress on $\mathbf{X}_r$ is optimized to produce $\mathbf{X}'_r$. A coarser set of coordinates, $\mathbf{X}'_{r+1}$, is produced from $\mathbf{X}'_r$ by multiplying it with the restriction operator $\mathbf{P}^r_{r+1}$. The value of the residual is assigned from the gradients of the stress:

$$\mathrm{T}_{r+1} = \nabla \hat{s}_{r+1}(\mathbf{X}'_{r+1}) - \mathbf{P}^{r+1}_r \nabla \hat{s}_r(\mathbf{X}'_r). \tag{3.36}$$

The procedure is then recursively called passing in $\mathbf{X}'_{r+1}$ and parameters pertaining to level l_{r+1}, resulting in coordinates $\mathbf{X}''_{r+1}$. Errors are smoothed using the interpolation function and the resulting coordinates are optimized K' times and returned.

The entire method consists of multiple V-cycles.

Further acceleration has been achieved by moving the calculations required to perform multi-grid MDS onto GPU architecture. [79]

3.7 Machine Learning

The following sections cover the machine learning methods that will be used in the following chapters. Section 3.7.1 details the process of *k*-means clustering which we will use in the DynDom6D method in Chapter 5, and Section 3.7.2 describes the kernel logistic regression algorithm that we will apply to sequences in Chapter 6.

3.7.1 *k*-means Clustering

k-means clustering is a machine learning algorithm used for the task of separating objects into *k* groups of similar items according to their features [8] [68]. Given a set of *k* clusters

$C_1 ... C_k$, and a set of samples $x_1 ... x_n$, the samples are assigned to the clusters to minimise the distances between points and the mean (or "centroid", c_j) of the cluster to which they belong according to a chosen distance metric, here the sum of squared errors[8]:

$$\sum_{j=1:k} \sum_{x_i \in C_j} \|x_i - c_j\|^2 \tag{3.37}$$

The classical algorithm starts with k randomly chosen centroids, then assigns each item to the cluster containing the nearest centroid. After assigning each item to one of the k clusters, the centroids are moved to the new mean point calculated from all items in the cluster. The assignment of items to their nearest centroid and the re-calculation of mean points are continued until the clusters converge at a local minimum. This is an unsupervised method; no labelled training data is required.

The process of k-means clustering makes numerical data the most appropriate type of feature, but an input vector can contain data from different feature spaces. The scale and the range of values of these features may be very different, which may make features more influential over the mean. For example, if input vectors contained information about people including their height in miles and their weight in milligrams, then the heights' numerical values would be much less than zero and the weights' values would be in the order of 10^7; variation in the weight would cause greater differences in the mean than variation in the height. On the other hand, if irrelevant or redundant data is included, or if some features are more strongly correlated with a good separation, it may improve the clustering if some features are more influential than others[101][83]. Many methods of filtering the features to retain only the most informative have been proposed for k-means clustering and other machine learning problems [83]. For example, Lu et al. [103] uses PCA to reduce the large feature set extracted from videos of changing facial expressions. Modha and Spangler [110] applies varying weight factors to features from different feature spaces in a pre-clustering step, which attempts to optimise the similarity within the clusters and the separation between them.

Other methods take place during the normal operation of the machine learning algorithm[83]; Huang et al. [77] iteratively updates a similar weighting during the clustering process[101].

The classical algorithm requires a pre-selected set of k initial centroids, but the ideal choice for the number of clusters to use may not be clear. There have been many approaches to this selection, often focussing on repeating the clustering with different values of k and choosing the best results based on some criteria such as a measure of the variance within clusters, or a comparison between the similarity within a cluster and the similarity between items in different clusters [31]. The location of the initial centroids is also important to the method's results, as the classical algorithm is only guaranteed to find a local optimum which may change with different starting positions [8][86].

3.7.2 Kernel Logistic Regression

The KLR in this thesis was performed using the UEA's machine learning toolbox [27].

Kernel Logistic Regression takes as its input a matrix of input vectors $\mathbf{x}$, and performs some non-linear transformation mapping $\mathbf{x}$ onto a representation in feature space resulting in a feature space representation $\boldsymbol{\phi}(\boldsymbol{x})$. KLR aims to train a vector of primary model weights $\mathbf{w}$ and a scalar bias parameter b [175] [27] such that:

$$\text{logit}\{y(\mathbf{x})\} = \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) + b. \tag{3.38}$$

The logit link function,

$$\text{logit}\{p\} = \log\left\{\frac{p}{1-p}\right\} \tag{3.39}$$

is a standard link function that provides as an output an a-priori estimate of the likelihood of a given input vector belonging to the target class.

The mapping of $\boldsymbol{x}$ to $\boldsymbol{\phi}(\boldsymbol{x})$ does not need to be manually enumerated; by defining $< \mathbf{x}_1, \mathbf{x}_2 >$, the inner product of any two given input vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, the mapping to features space is implicitly defined. As long as the selected inner product function adheres to a set of criteria known as Mercer's conditions, the feature representation is implicitly defined by the calculation of the function. The condition for a valid function defining this implicit mapping, or kernel function, is that the resulting kernel matrix $\mathscr{K}$ for any set of input vectors $\mathbf{x}$ must be positive semi-definite (see 3.5). This transformation into a new space can map non-linear features into a representation in which they are linearly separable, which allows the training problem to be the simpler case of finding a linear separation of the feature space representations of the data.

$$\mathscr{K}(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}') \tag{3.40}$$

We will look at three specific kernels, which are the three that will be used in the KLR results chapter.

The linear kernel,

$$\mathscr{K}(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}' \tag{3.41}$$

is a straightforward implicit mapping where the input vector is the same as the representation in feature space, and hence a weight is associated with each input feature. This provides a linear separation of the original data.

The polynomial kernel,

$$\mathscr{K}(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + c)^d \tag{3.42}$$

incorporates both the linear terms and products of the primary input vectors, as well as scalar hyper-parameter c.

The radial basis function (RBF) provides a Gaussian separation of the data with Gaussian spread controlled by the hyper-parameter θ:

$$\mathscr{K}(\mathbf{x},\mathbf{x}') = \exp\{-\theta\|\mathbf{x}-\mathbf{x}'\|^2\}. \tag{3.43}$$

Given a set of l training examples, where $\mathbf{x}_i$ is the i^{th} training sample's input vector and t_i and y_i are its actual and predicted outcome respectively, the regularised "cross-entropy" loss function:

$$E = \frac{1}{2}||\mathbf{w}||^2 - \frac{\gamma}{2}\sum_{i=1}^{l}[t_i\log\{y_i\} + (1-t_i)\log\{1-y_i\}], \tag{3.44}$$

is minimised during the training process, which iteratively updates the applied weights until either a limit is reached or a minimum is found. The models in Chapter 6 are trained using an iteratively reweighted least squares training procedure [27] on the training data.

This minimisation is solved in the dual representation, where the dual model parameters are contained in the vector α:

$$\mathbf{w} = \sum_{i=1}^{l}\alpha_i\boldsymbol{\phi}(\mathbf{x}_i). \tag{3.45}$$

From these trained models, the expected outcome is calculated from an input vector using:

$$\text{logit}\{y(\mathbf{x})\} = \sum\alpha_i\mathscr{K}(\mathbf{x}_i,\mathbf{x}) + b, \tag{3.46}$$

where:

$$\|\mathbf{w}\|^2 = \alpha^{\mathbf{T}}\mathbf{K}\alpha. \tag{3.47}$$

The trained model is then applied to the reserved test data for the final assessment of its accuracy.

These weights can be extracted from the trained model in order to identify and examine the features that were most discriminative during training.

These kernels and processes require the selection of hyper-parameters such as the constant θ in the RBF kernel (Equation 3.43) or the regularization parameter γ in the loss function, Equation 3.44. The hyper-parameters were tuned using the Nelder-Mead simplex [116] using approximate leave-one-out cross validation [27].

3.7.3 Accuracy of Machine Learning Techniques

A typical method for assessing a machine learning model for classification or regression is partitioning of labelled data into separate sets for training and testing. After the model has learned from the training data, it is applied to the test data, so for each input vector of the test data $\mathbf{x}_i$, an expected value t_i and a predicted value y_i are obtained. There are various ways that the discrepancy between the ground truth and the output of the model can be quantified. The following will discuss performance using the Receiver Operating Characteristic (ROC) curve[48], which plots the rate of false positive results on the x axis against the rate of true positive results on the y axis. The area under the ROC (AROC) can be taken as a single measure of the performance of a model. In a random model assigning one of two classes by chance, the false positive and true positive rates would be expected to progress linearly and evenly, creating an ROC curve where the with an AROC of 0.5. A perfect model that accurately classifies every input would be reflected in an ROC that reaches a true positive result of 1 while still having a false positive rate of 0, which would result in an area under the ROC of 1.

3.7.3.1 Significance of AROC

The Mann-Whitney U-test is a non-parametric test of the significance of the difference between two populations[106]. It can be used to gauge the significance of a model's performance by studying the predicted values [113], divided into two sets. One set is built by taking the predicted values for all input vectors where the expected outcome is positive (the sample belongs to the target class), and another set is constructed from the predicted values of negatively labelled samples. If the model has no predictive power, with an AROC of 0.5, the two sets would be expected to be randomly drawn from the same population. If the model does exhibit predictive power then the mean of the positively labelled set should be higher than that of the negatively labelled set; a randomly selected sample from the positively labelled set is likely to be higher than a randomly selected sample from the negatively labelled set.

Samples from both sets are ranked according to their predicted value, and a U statistic is calculated for each set corresponding to the number of times its elements are greater than elements of the other set in a pairwise comparison[113]. For small sets, a p-value is retrieved from a set of tables based on the smallest U statistic and the number of elements in each set, and for larger sets the p-value comes from an approximation using the normal distribution [106].

The Mann-Whitney U-test is similar to the Wilcoxon rank-sum test [169].

3.7.3.2 Comparison of AROC

When the AROC is used as a measure of a model's performance, comparing the performance of two models on the same test set requires a comparison between AROC values. A higher AROC indicates more accurate results, but the significance of the difference between two AROCs must be calculated to identify the confidence with which the performance can meaningfully be described as better. The DeLong test[39] is a non-parametric indication of

statistical significance for the difference between AROCs achieved by different models that have been tested on the same samples, taking the paired relationship into account.

3.8 Algorithms for Identifying Dynamic Domains and Hinge-Bending Regions

We discussed in Section 2.3.2 how proteins undergoing conformational change can be divided into semi-rigid dynamic domains. Protein domain movements have been described as categorised as predominantly hinge and predominantly shear [59]. Hinge, or interface-creating [154], motions are characterised by the large, perpendicular relative movements of rigid domains, aided by a flexible hinge-bending region or regions. Both these hinge bending regions and the domains themselves can be identified algorithmically.

Dynamic domains can be determined through analysis of a pair of structures solved before and after the conformational change has taken place. If two structures are known, FlexProt [145] performs a flexible alignment by finding all possible alignments of sections and finding an optimal subset that best connects the segments. The subset is found from a directed weighted acyclic graph, where nodes are the possible segments and edges are the hinge regions [144].

DynDom [95] [69] also requires two structures to perform domain assignment. The structures are superposed and a window is slid over the main chain to create overlapping segments of backbone. The rotations required to bring the atoms comprising each segment in the first conformation into alignment with the corresponding atoms in the second conformation are calculated. The underlying principle behind DynDom is that as domains move semi-rigidly, parts of a protein that are in the same domain will display similar rotations between conformations, whereas the rotations of parts in different domains will be different. These calculated rotation vectors are treated as points in a 3-dimensional feature space, in

which groups of similar rotations are found using the *k*-means clustering algorithm (which is detailed in section 3.7.1). Residues are considered part of hinge bending regions if they are both located between a pair of domains along the backbone and not within the "ellipsoids of constant probability density" that surround the domain clusters [73]. The DynDom program also performs analysis of pairs of domains by describing the movement of a domain relative to another domain that is held in a fixed position between conformations, which yields the location of a screw axis, the rotation about it, and the translation along it that the moving domain undergoes. As the window is only slid along atoms belonging to the backbone, the method is restricted to a single chain of a protein and ignores atoms belonging to side chains. The DynDom method was extended by DynDom3D [126][60] which replaces the original sliding window with a sliding block which is placed over all atoms.

The Motion Tree method [89] also examines different structures of the same proteins. They construct distance matrices for each conformation of the protein then find distance difference matrices that compare pairs of distance matrices, which identify areas of the protein that have moved relative to other parts. These motions are hierarchically clustered and represented as dendograms; domain level motions are displayed as the parents of smaller motions to allow study of the details of a protein's motions.

Other methods predict the location of dynamic domains or hinge bending regions from only one structure. Methods based on Normal Mode Analysis (NMA) [72] investigate the flexibility of a protein's structure. While proteins have an extremely high number of potential conformations, their experimentally-derived conformations are usually small variations on a global energy minimum referred to as the "native" structure [4]. NMA investigates the potential fluctuations around this global minimum, constructing approximate representations of the motions of domains through calculation of the derivative of the potential energy function at points in the protein [24]. These studies have yielded techniques that can identify the low-frequency, high-amplitude motions that occur during domain motions [75].

HingeProt [44] applies Elastic Network Model analysis to fluctuations of residues calculated using NMA. If a pair of residues have correlated fluctuations, they are assumed to move in the same direction. The structure is divided into fragments of correlated residues, and the final rigid sections are found using "spatial fragment clustering". FlexOracle [50] repeatedly slices a chain along the backbone and assesses the energetic cost of separating the remaining segments, on the grounds that this will require the least energy when the slice occurs in a hinge-bending region rather than in a stable domain.

Fewer options are available for identifying a hinge bending region from a single conformation or from the sequence alone. One example of a sequence-based predictor is proposed in Flores, et al [52], in which the authors constructed annotated datasets of hinge bending regions, based on the Database of Macromolecular Movements [58]. From statistical analyses of their composition, they created a predictor that required sequence information alone. They calculated log-odds frequencies for a 17-residue-long sliding window, run across sequences in a training subset of their computer-annotated dataset. This assigned scores to the presence of given amino acids at each location in the window. They scored their test set, also taken from their computer-annotated dataset, by applying the same window to each residue and summing the score across each window, assigning the central residue to a hinge bending region if the summed score was above a given threshold. However, the results achieved in this test set did not appear to be significantly different to randomly assigning each residue to either the “hinge bending region” or “not hinge bending region” class. They incorporated information about secondary structure and active site location into a further predictor, HingeSeq, which improved the predictive power over sequence alone. This was incorporated into a later predictor, HingeMaster [51], which also used hinge predictions obtained through NMA and structural comparisons.

Boden and Bailey [12] also use the relationship with secondary structure for a predictor of hinge-bending regions. The process first predicts the secondary structure from a sequence

of amino acids, using a method that outputs a probability for each residue to belong to a secondary structure class, and therefore indicates segments of the sequence that are confidently and inflexibly within one type of secondary structure and segments that are likely to have varying secondary structure [13]. From these "continuum secondary structures" they then predict the location of flexible regions.

The FlexPred web server [93] uses support vector machines to predict “conformational switches”, which they describe as areas of flexibility that drive conformational changes [92]. As with HingeAtlas, they also incorporated structural and biological information, and found that this improved performance over sequence-based methods alone.

3.9 Applications of Machine Learning to Protein Sequences

Machine learning techniques have been applied to other questions related to classification and regression from protein sequences, including the problems of secondary structure prediction [46] [134] [172] and homology prediction [97]. The “sliding window” used by the approach of Flores, et al. [52] to input vectors, as described above, is used to build input vectors for many of these sequence based methods. The optimal length of the window varies from study to study but common values are between 9 and 17. MUFOLD-SS [46] predicts secondary structure using a full sequence of up to 700 residues, split into a deep “inception-inside-inception” neural network. Convolutional layers with windows of up to 9 are applied to combine local and global context. Fang, et al.[46] compared results with those reported in Wang et al. [165] and Busia and Jaitly[23], convolutional neural network methods that use fixed 11 and 17 residue windows, and reported that MUFOLD-SS outperformed them both. The window lengths used by the FlexPred conformational switch prediction server [92] are among the smallest, at 3 and 5 residues.

Chapter 4

Biomolecular Morphing using Multi-Dimensional Scaling

Chapter 2 contained a background on the motivation behind protein morphing: as proteins perform their functions, they change shape, and while structures at points along this trajectory may have been solved, it is easier to visually follow changes in an animation than identifying changes by examining before and after images. The link between protein function and conformational change makes understanding the dynamics of a protein's movements important, as through understanding this mechanism we gain insight into how a biological process works and, in the case of drug design, how this process may be helped or hindered. In Chapter 3, existing algorithms that constructed intermediate frames between solved structures were discussed. These incorporated three common elements: interpolation of Cartesian coordinates, interpolation of inter-atomic distances, and energetic approaches. Interpolation of inter-atomic distance derives from the principle that these distances in the solved structures are based on important properties of protein structure. Moving from one set of distances to the other, while keeping all distances between the two solved values, should then change the shape of the protein without allowing the structure to counter the physical constraints of

protein structure. These interpolated distances may not all be simultaneously achievable, so it is desirable to find a method for creating the best fit to the set of ideal distances.

The method of Kim et al. [87] is described as a form of Elastic Network Model where the distances between pairs of atoms are modelled as "springs". Each inter-atomic distance is interpolated at each frame, creating an $n \times n$ distance matrix. To create a set of coordinates this is reduced down to an $n \times 3$ matrix, with x, y and z coordinates for each atom, by minisiming a cost function. This process can be seen as a form of Multi-Dimensional Scaling (MDS). While Kim et al.'s concept offers a strong compromise between computational performance and quality of output, there are faster and more accurate approaches to multi-dimensional scaling which could offer improvements in both areas. Their implementation can also be improved on as structures which it cannot morph correctly result in ”exploding” morphs which are neither plausible nor easy to follow.

We present a method of morphing that starts with an inter-atomic distance interpolation and creates intermediate structures from these ideal distances using a blend of methods from the MDS family. The technical background in Chapter 3 describes some of the MDS methods by which a matrix of dissimilarity can be turned into a plot with a chosen number of dimensions. In our case, a matrix of inter-atomic Euclidean distances is built into a 3-dimensional co-ordinate set for each frame of the interpolation. This method is compared to similar protein morphing techniques in terms of the quality of the produced structures and of closeness to a known intermediate. In addition to proteins, the method has been extended for DNA and RNA morphing. A novel version focuses on producing visualisations of biomolecular docking to present a trajectory where the relative motions of each component can be observed.

However, MDS protein morphing is based primarily on geometry. The method uses knowledge about the nature of proteins as polypeptides to construct a hierarchy of atoms and residues for use in the multi-grid acceleration, and similarly the structure of DNA and RNA

used to build a hierarchy of nucleotides and atoms, but after this pre-processing the method relies only on information derived from the Cartesian co-ordinates. The only constraints applied to the structure after creating this hierarchy are holonomic, relating only to the inter-atomic distances and time in the path. There are no tests for physical constraints like bond length and bond angles. The docking version of our biomolecule morphing method was implemented to assist in tasks frequently required during the process of structure-based drug design: the study of mechanism of interaction between a protein and its ligand or the study of the interaction between a target protein and candidates for potential drugs[36]. For our method to be helpful in this context we need to study how accurately it performs against a method that is known to accurately model the true path a biomolecule might take.

Section 3.4 described the family of techniques referred to as Molecular Dynamics (MD), which simulate the behaviour of molecules by calculating the effects of atoms on one another by integration of interaction forces at a series of time intervals. This simulation is extremely accurate, though computationally very expensive. A simulation lasting 100 nano-seconds can take days to weeks to produce, depending on system size and hardware.

We will apply both the MDS morphing technique and the "gold standard" MD to an example of a particular interaction, the binding of cancer drug nivolumab to its target protein, programmed death protein-1 (PD-1), in order to examine the interaction in detail and to identify potential areas of improvement for the MDS protein morphing technique.

The results have been published in the Journal of Molecular Graphics and Modelling[160]. A web server for performing the developed method is available at http://morphit-pro.cmp.uea.ac.uk/.

4.1 Morphing Process

The motion from one configuration to the other is created by interpolating the distances between pairs of atoms at the start and end configuration and then creating structures that best fit these intermediary distances using MDS.

We interpolate for a series of m intervals where the time t is between 0 and 1 in increments of $\frac{1}{m}$. Distance matrices D_0 and D_1 are constructed from the $\frac{n}{2}(n-1)$ atom pairs in the start and end conformers respectively, such that $D_t ij$ is the distance between atom i and atom j at time t, where the distance metric used is the Euclidean distance,

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}. \tag{4.1}$$

At each interval between the start and end frames we take a value, λ, and calculate the interval's ideal interatomic distances from Equation 4.2.

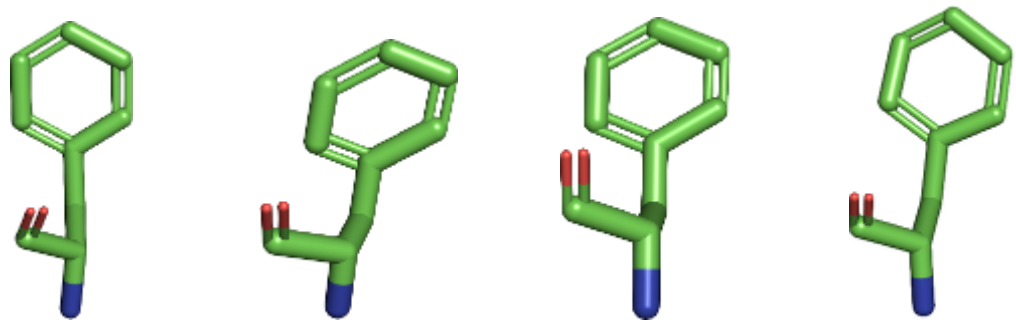
$$D_t = (1-\lambda) D_0 + \lambda D_1 \tag{4.2}$$

In reality, each pair of atoms will have a distinct value of λ at each time step. For the sake of computational complexity we use a consistent linear value of $\lambda = t$. This produces a morph that moves steadily from one conformation to the other in a consistent motion, where halfway through the morph the protein is halfway between the end states.

A value between 1 and 0 for λ means distances in our ideal distance matrix will be between the distances at the start and end of the morph. Provided that we can create atomic positions that maintain these pairwise distances these values should remain in the allowed ranges of bond lengths and bond angles.

This gives us a series of distance matrices which represent ideal distances between atoms at each position in the morph, but these distance pairs may not all be simultaneously possible. We construct a model of atoms by assigning positions that most closely fulfil the $\frac{n}{2}(n-1)$ distance pairs using MDS.

We use three of the MDS methods described in Section 3.6. Classical MDS is a fast eigenvalue-based method which we use to build our initial set of coordinates from the distance matrix. As the chapter will show, classical MDS alone does not result in plausible structures, so further improvements are gained by applying more modern MDS techniques. Figure 4.1(a)



(a) Solved structure (b) Classical MDS (c) SMACOF (d) Weighted

Fig. 4.1 A comparison of a single phenylalanine residue taken from a solved structure (pdb code 1CLL) and from the intermediate frames created by various MDS morphs from 1CLL to 1CM1: a) the original solved structure; b) classical MDS; c) SMACOF MDS with no weighting applied; d) SMACOF MDS with weighting applied up to a cut-off distance of 4Å.

shows the actual shape of a single phenylalanine in a solved calmodulin structure with PDB code 1CLL, and Figure 4.1(b) shows the results from an intermediate structure created by a classical MDS morph attempting to create frames between the 1CLL structure and another solved calmodulin structure, 1CM1. The ring should rotate 180° but maintain its shape. In the classical MDS version of the morph, the ring is skewed and bonds are clearly distorted; the two bonds parallel to the stalk have been compressed, and the other four bonds making up the ring have been stretched.

The Scaling by MAjorization of a COmplicated Function (SMACOF) method is an iterative method that repeatedly applies a monotonic update function to the set of coordinates, minimising a cost function (Equation 3.24) that incorporates a measure of distance between desired and observed results (stress) until either a limit or a local minimum is found. Because the method is not guaranteed to find the global minimum, the initial position of each atom has an effect on the results; we use the results of classical MDS as the input to SMACOF MDS. Figure 4.1(c) shows the results of classical MDS followed by SMACOF MDS. The bonds are closer to their original length, but still show distortion. However, an advantage of SMACOF over classical MDS is that weighting can be applied to the stress function, making some distances more important than others. In Figure 4.1(d), inter-atomic distances greater

than 4Å are zero-weighted during optimisation, so the distances between pairs of atoms that are close enough to be bonded are considered, whereas more distant atoms are ignored. In this version the bonds are visibly improved, with no apparent distortion. Section 4.4.2 will elaborate on how the 4Å was chosen.

The third MDS method examined here is multi-grid MDS[21], which uses the relationship between atoms on a coarse level of points and atoms on levels with a finer resolution to achieve faster minimisation of a stress-like function. The principles of the updates applied at each step are based on those of the SMACOF method, but the hierarchical method provides better performance.

Our multigrid acceleration was based on the implementation provided by Bronstein and Bronstein[21], with minor changes to the code in order to implement weighting and faster running by moving the matrix multiplication to the GPU.

We construct our 3-level hierarchy from the hierarchy inherent in the biomolecule. When the structure to be morphed is a protein, the full protein with all atoms is our first level, with the highest resolution. The second level is coarser, and includes only the backbone atoms from each residue. Finally we use only the Cα atoms from each residue in the coarsest level. The restriction matrix P_1^2 defines the restriction from the first (all-atom) level, l_1, to the second (backbone) level, l_2. If the first level contains n atoms, and the second m, P_1^2 is a sparse matrix of size $m \times n$, where element (i, j) is 1 if the i^{th} atom in level 2 is the closest backbone atom (belonging to the same residue) as the j^{th} atom in level l. The interpolation matrix P_2^1 describes the inverse of this relationship, and is set as $(P_1^2)^{\text{T}}$. The restriction matrix P_2^3 sets element (i, j) to 1 if the i^{th} element in level 3 is the Cα atom belonging to the same residue as the j^{th} element in level 2. Again, the interpolation matrix P_3^2 is set to the transpose of the restriction matrix.

Weighting is applied to the multi-grid MDS method by weighting inter-atomic distances in the stress function as in the standard SMACOF technique; these weights are assigned

a value of 0 if the atoms are more than a set distance apart. We use three different "cut-off" distance thresholds as the three different levels will have different atomic density. We restricted the thresholds such that they would increase linearly in order to reduce the space to search for optimal values, though other weighting patterns were considered in Section 4.4.4.

When DNA or RNA is to be morphed, a similar approach is taken. All atoms make up the finest layer, which is restricted to the sugar-phosphate backbone's atoms in the middle layer, and then a single carbon atom (the $5'$ carbon) from each nucleotide in the coarsest layer.

All versions of MDS considered for this process require some post-processing to form a coherent animation, as MDS is concerned with the relative position of objects and not their location in a set of coordinates. For example, if an MDS method was passed a dissimilarity matrix derived from the distance between cities, and output a set of 2-dimensional coordinates to act as a map, there would be no means of identifying their true orientation with regard to the cardinal directions, and the set of cities may have been flipped about the North-South or East-West axis. It is important for our method to maintain the same orientation between frames in order for the viewer to follow the motions of the morph, and because changing the chirality of the structure would have physical implications. We perform a Procrustes or least-squares best fit alignment to bring each frame into alignment with the initial solved structure, allowing rotation and inversion of the atoms' coordinates but maintaining the scale.

Tests for the following results were conducted on several groups of test data. Preliminary work was conducted using T4-lysozyme and calmodulin, representing an easy and a difficult case respectively. To explore the relationship between the angle of rotation undergone during a morph and its strain, a test set of 19 pairs of structures were taken from the DynDom database which represent a range of angles. For a larger bulk set of results, the chains entered for 100 DynDom runs were selected at random. The full lists of the latter two datasets are included in Appendix A.

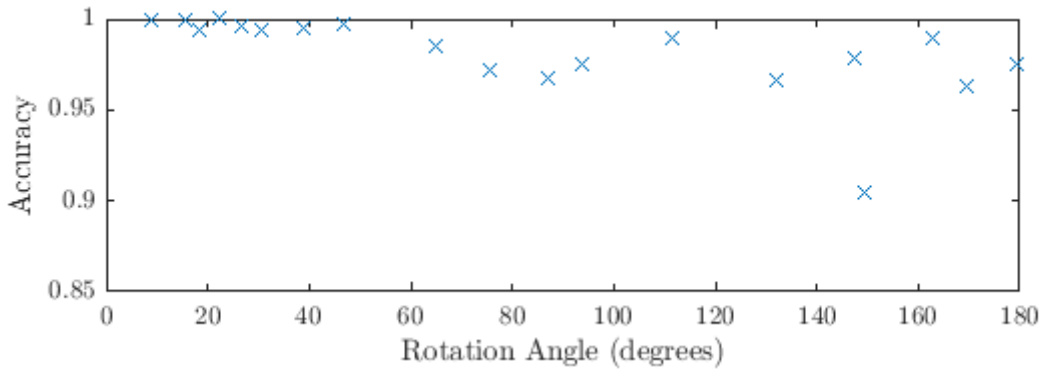


Fig. 4.2 Angle of rotation versus accuracy (strain) of 20 classical MDS morphs, using pairs of structures chosen for their range of rotation angles.

4.2 Accuracy of morph versus angle of domain rotation

During development of the method, preliminary tests were performed on T4-lysozyme and calmodulin, on the basis that the method should be able to morph small, direct changes like the T4-lysozyme as well as large rotations and translations like the 149.6° rotation displayed by calmodulin. This assumption was investigated using classical morphs of the 19 structures in our dataset containing a range of rotations. Figure 4.2 shows the accuracy achieved by each of these morphs produced using classical MDS as a function of their rotation angle, which in general supports this principle. Figure 4.3 shows the same plot using the 100 pair dataset, which contains a spread of rotation angles more representative of the DynDom data as a whole. While both plots feature some low-accuracy outliers among their small rotation angles, most morphs of small rotations have an accuracy close to 1, while accuracy of morphs of larger rotations is more spread out.

4.3 Calmodulin example

Among the pairs of structures used for preliminary testing, a pair of calmodulin structures (PDB codes 1CLL and 1CM1 as shown in Figure 4.4) were used as an example of a

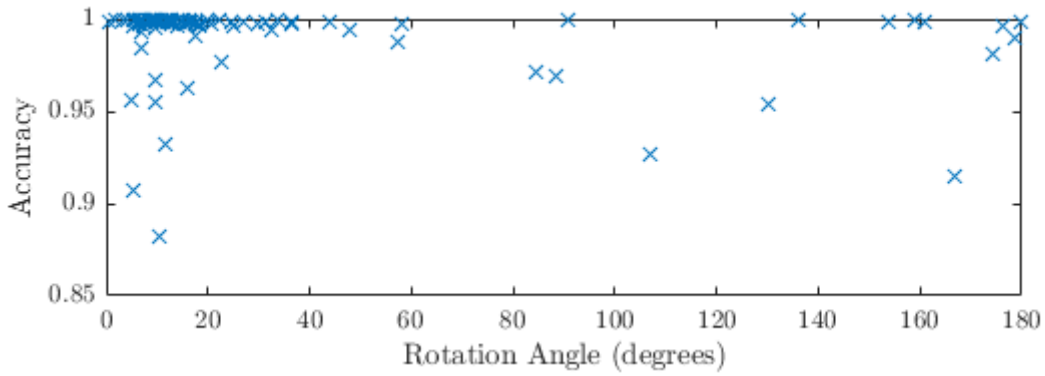
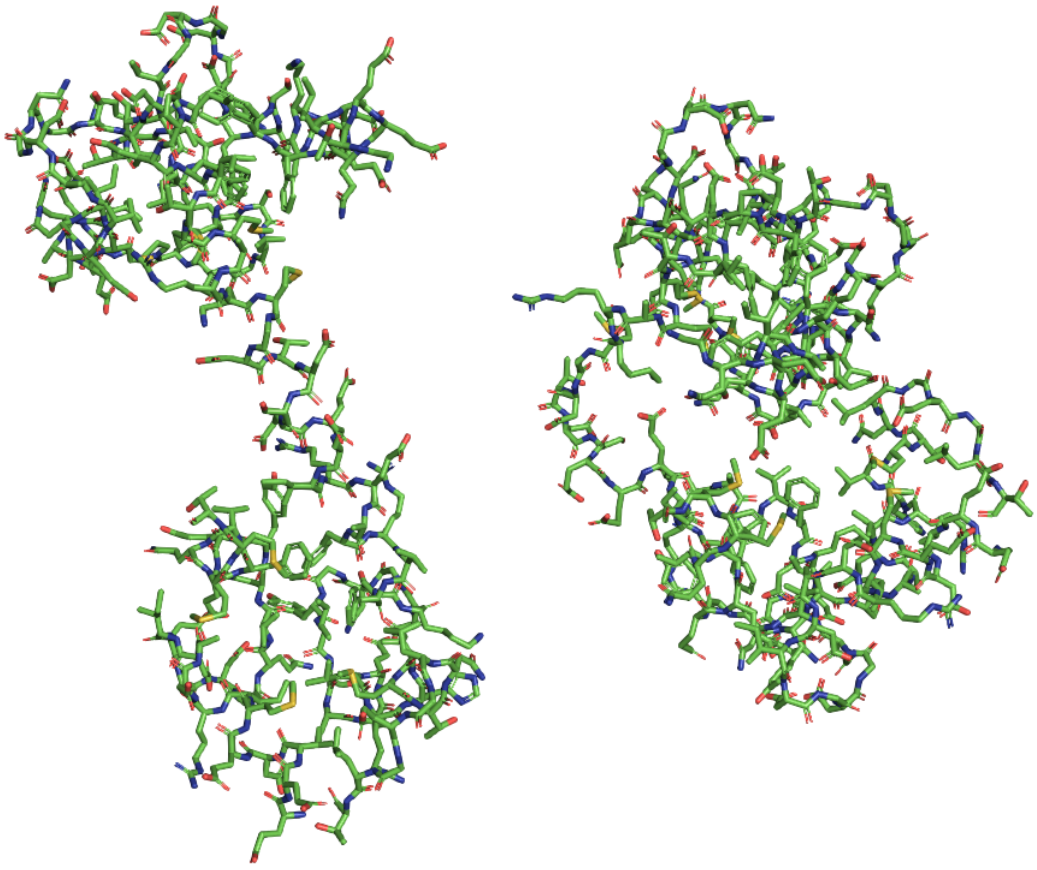


Fig. 4.3 Angle of rotation versus accuracy (strain) of 100 classical MDS morphs.

pronounced domain motion, where the large rotation of 149.6° presented a difficult challenge for the morphing methods. The following subsection details the results of the preliminary testing on these calmodulin structures. In particular, this will focus on the components of the structure that are included in the MolProbity score, the validation method which we will use as a measure of successful morphing in the next section.

We examined the central frame of the morph, as it represents the furthest point from either solved structure. Figure 4.5(a) shows this frame taken from a classical MDS morph between the two solved structures, which exhibits visible distortion in its bond lengths. The domains appear flatter and more skewed when compared to their shapes in the known conformations. Figure 4.5(b) shows the same frame taken from a morph created using the multi-grid SMACOF method.

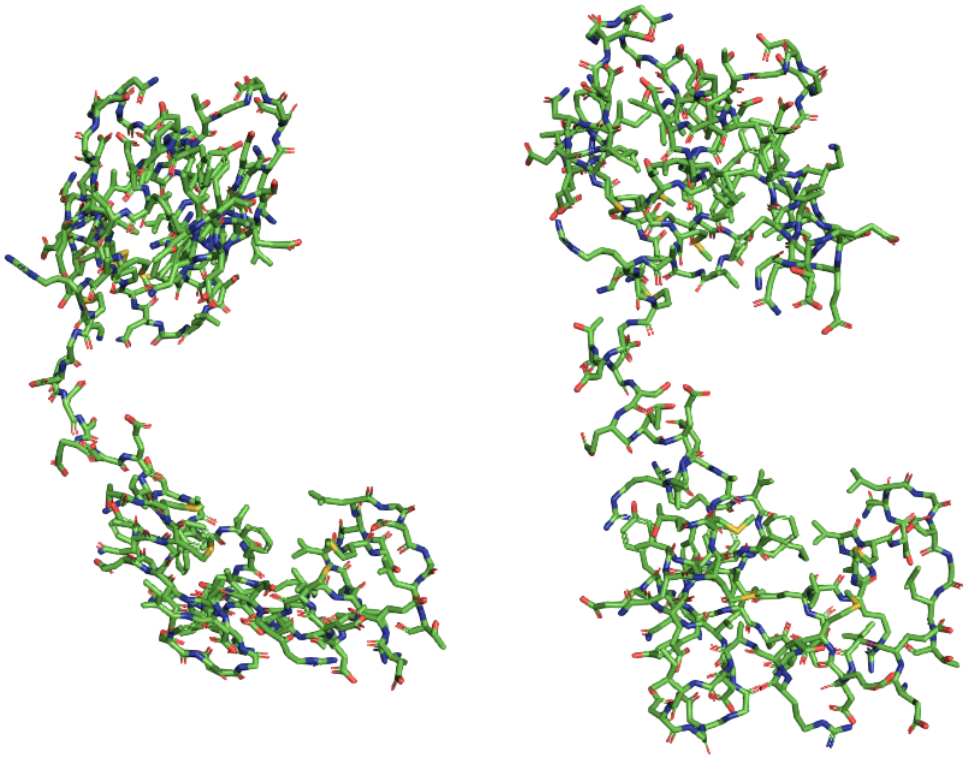
The torsion angles were calculated for the middle frame of the animation, and have been plotted in Figure 4.6. As discussed in section 2.1.2.1, these Ramachandran plots show the ϕ and ψ angles calculated along the backbone. The marked regions show areas where the angles are permitted for valid protein structures, where the red sections are the most favoured regions. The classical result in Figure 4.6(a) and the weighted multi-grid SMACOF method in 4.6(b) both feature disfavoured ϕ-ψ pairs, but the classical plot's points are more scattered and less tightly clustered in the favoured regions.



(a) Calmodulin structure with PCB code 1CLL

(b) Calmodulin structure with PCB code 1CM1

Fig. 4.4 Solved calmodulin structures in open and closed conformations.



(a) Central frame of a classical MDS morph between solved calmodulin structures.

(b) Central frame of a multigrid SMA-COF MDS morph between solved calmodulin structures.

Fig. 4.5 The central frame of a morph between solved calmodulin conformations.

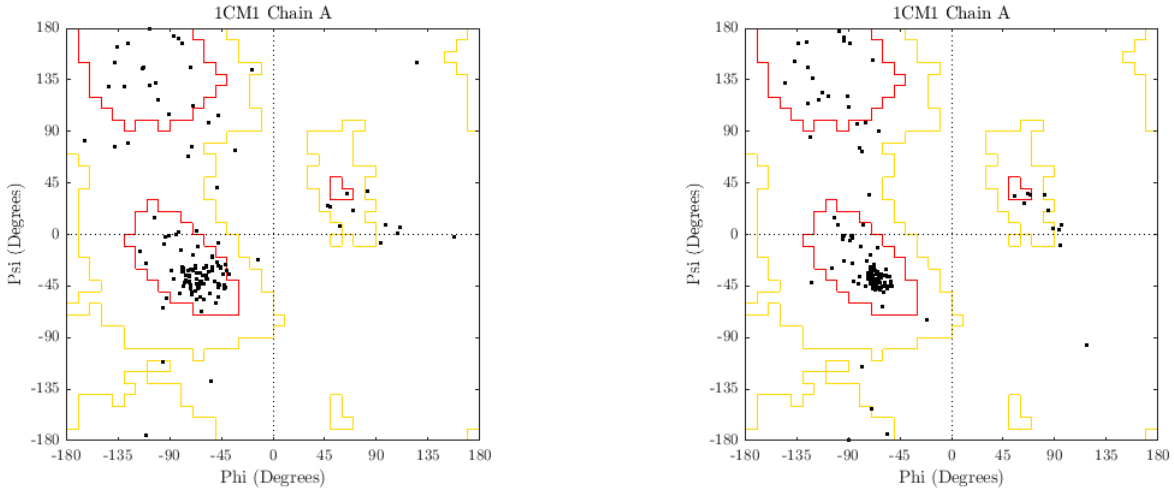


Fig. 4.6 Ramachandran plot for the central frame of a calmodulin morph (PDB codes: 1CLL to 1CM1) using the classical MDS (top) and multigrid MDS (bottom) methods.

It is a particular problem for both the clarity and feasibility of a protein structure when atoms clash. The result will be both incorrect and difficult to follow due to overlapping objects, and the stretching or compressing of bonds would violate physical laws. Therefore intermediate structures should maintain numbers of atomic pairs with short inter-atomic distances. Figure 4.8 shows a histogram containing counts of shorter inter-atomic distances. The initial structure, PDB code 1CM1, contained 13,092 inter-atomic distances below 4Å and the end structure contained 12,898. The central frame of the classical morph contained 22,304, whereas the MG MDS morph contained 12,976, which is very close to the mean of the solved structures, 12,995.

The principle of protein morphing methods that use inter-atomic distance interpolation is that the interpolated target distance, if faithfully reconstructed, would yield a valid structure on the path between the two solved structures. We have plotted how well our (good) multi-grid result, and our (poor) classical result adhere to their ideal distances in Figure 4.9. Figure 4.10 shows these distortions limited to the atom pairs that may be bonded in the pair of solved structures; their inter-atomic distance is less than or equal to 4Å in both. For the multi-grid method, the mean difference is 0.0388Å when restricted to bonded atoms and 2.4586Å for

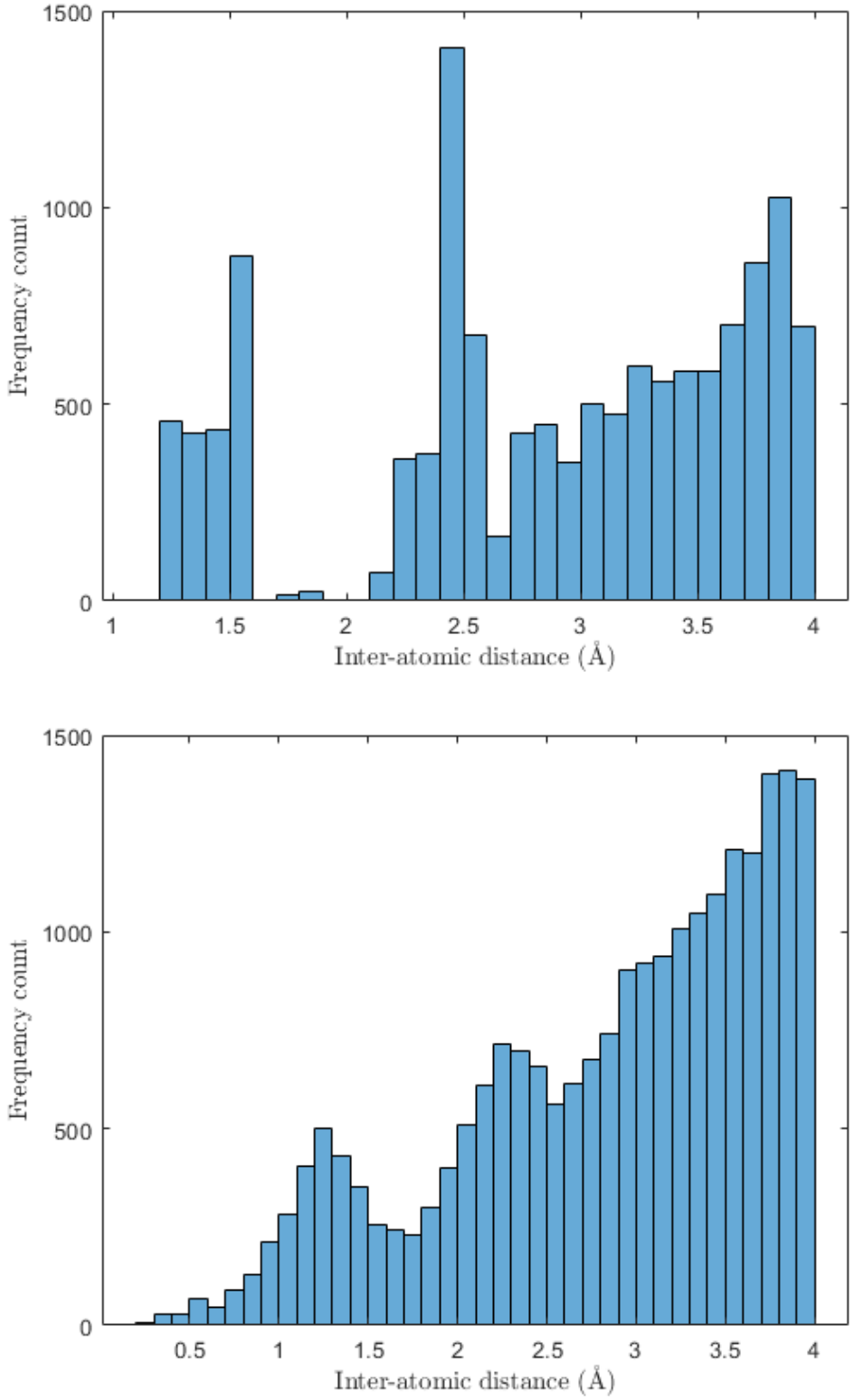


Fig. 4.7 Histogram showing counts of inter-atomic distances of 4Å or less in calmodulin morphs (top: original 1CLL structure; middle: classical MDS).

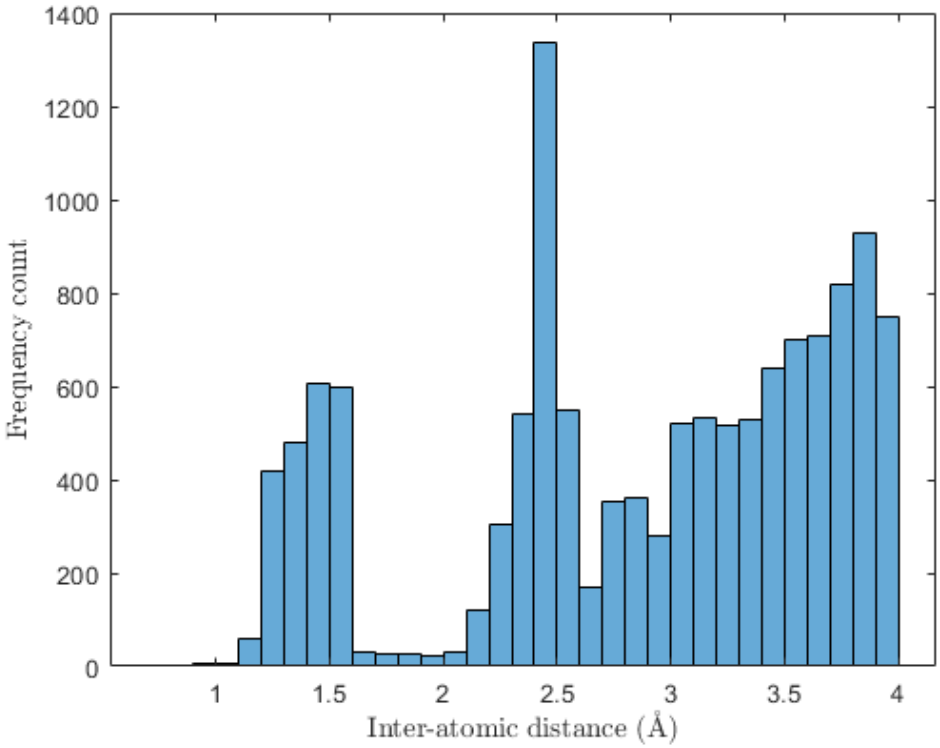


Fig. 4.8 Histogram showing counts of inter-atomic distances of 4Å or less in calmodulin morphs, using multi-grid MDS morphing with a cut-off distance of 4Å.

all atom pairs. The classical method had a smaller overall mean difference, 1.5864Å, but a longer mean difference of 0.4503Å. Due to the weighting applied to the multi-grid MDS method, there are no constraints looking at the longer inter-atomic distances, so while the multi-grid MDS method reproduces the short distances very accurately it has much larger discrepancies beyond the longest (12Å) cut-off, as seen in Figure 4.11. This figure contains a point for each atom pair, marking the difference between the ideal and observed distance between them as a function of their initial inter-atomic distance. The high performance of multi-grid MDS compared to the classical method, despite the fact that it is on average worse at reproducing all ideal distances, indicates that the most important atom pairs to consider are those with small inter-atomic distances in the solved structure.

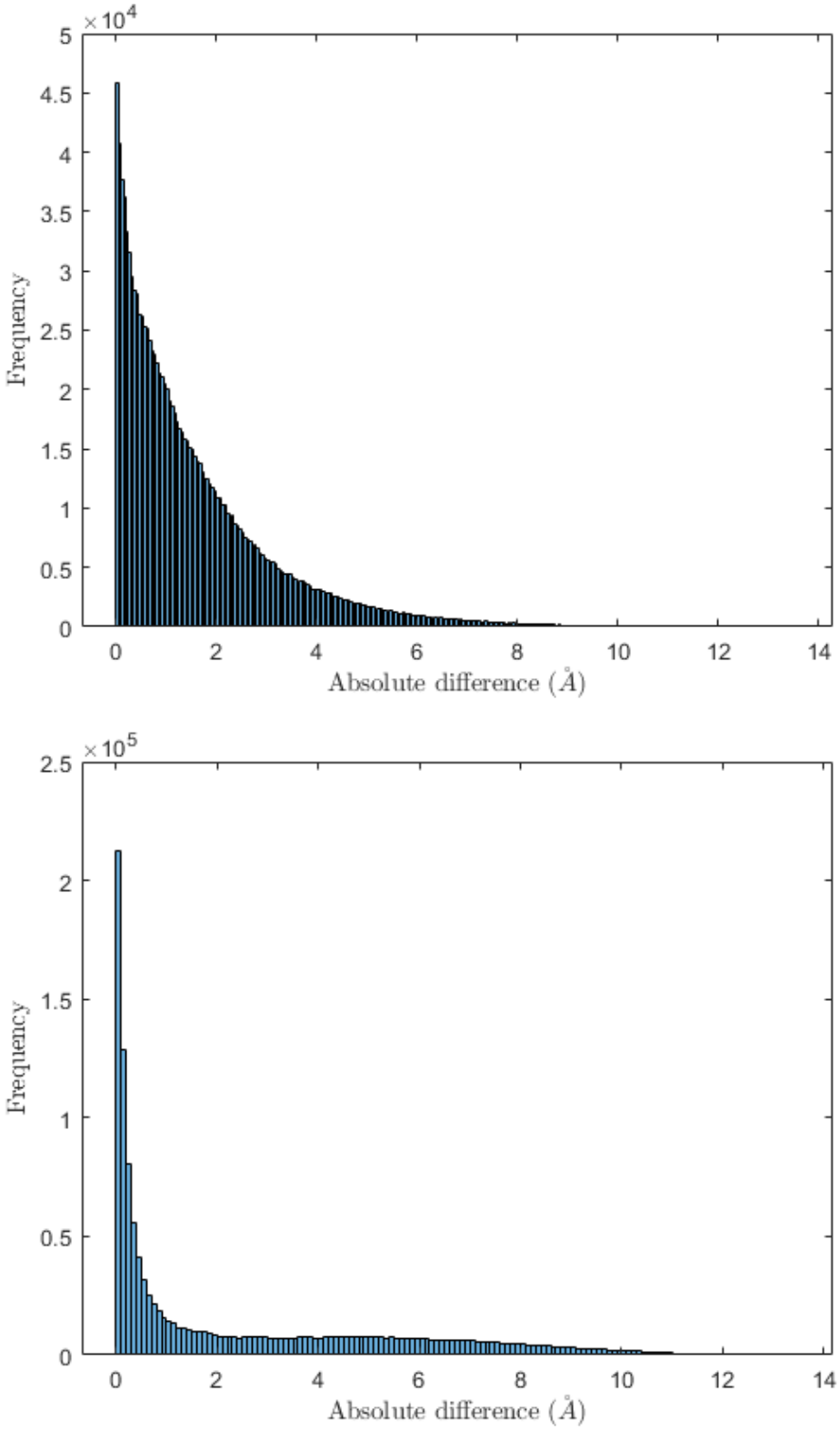


Fig. 4.9 Histogram showing absolute difference in Å between ideal and constructed interatomic distances in the central frame of a calmodulin classical (top) and MG MDS 4 (bottom) morph.

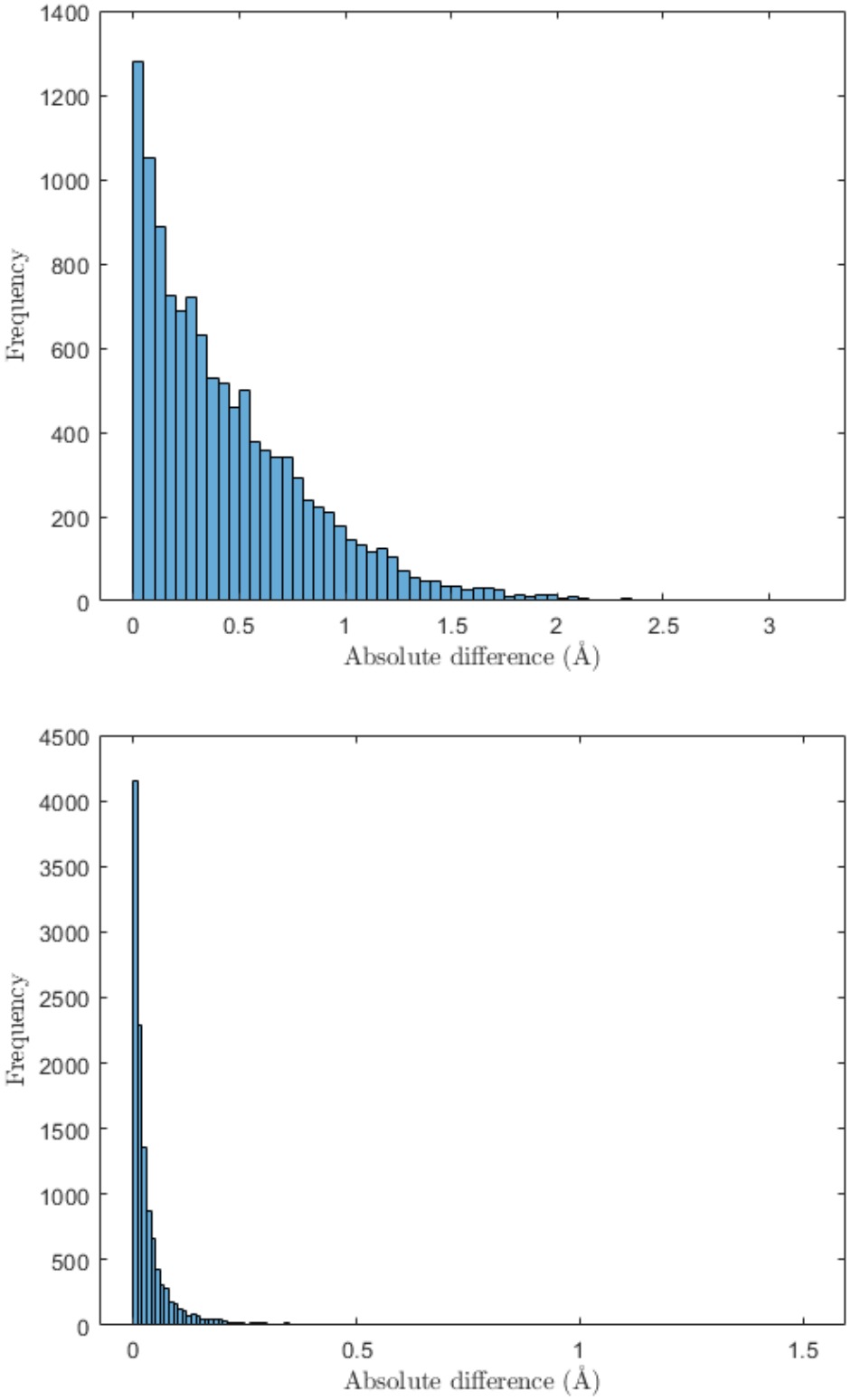


Fig. 4.10 Histogram showing absolute difference in Å between ideal and constructed interatomic distances in the central frame of a calmodulin classical (top) and MG MDS 4 (bottom) morph, looking only at potentially bonded atoms (distances less than or equal to 4Å.

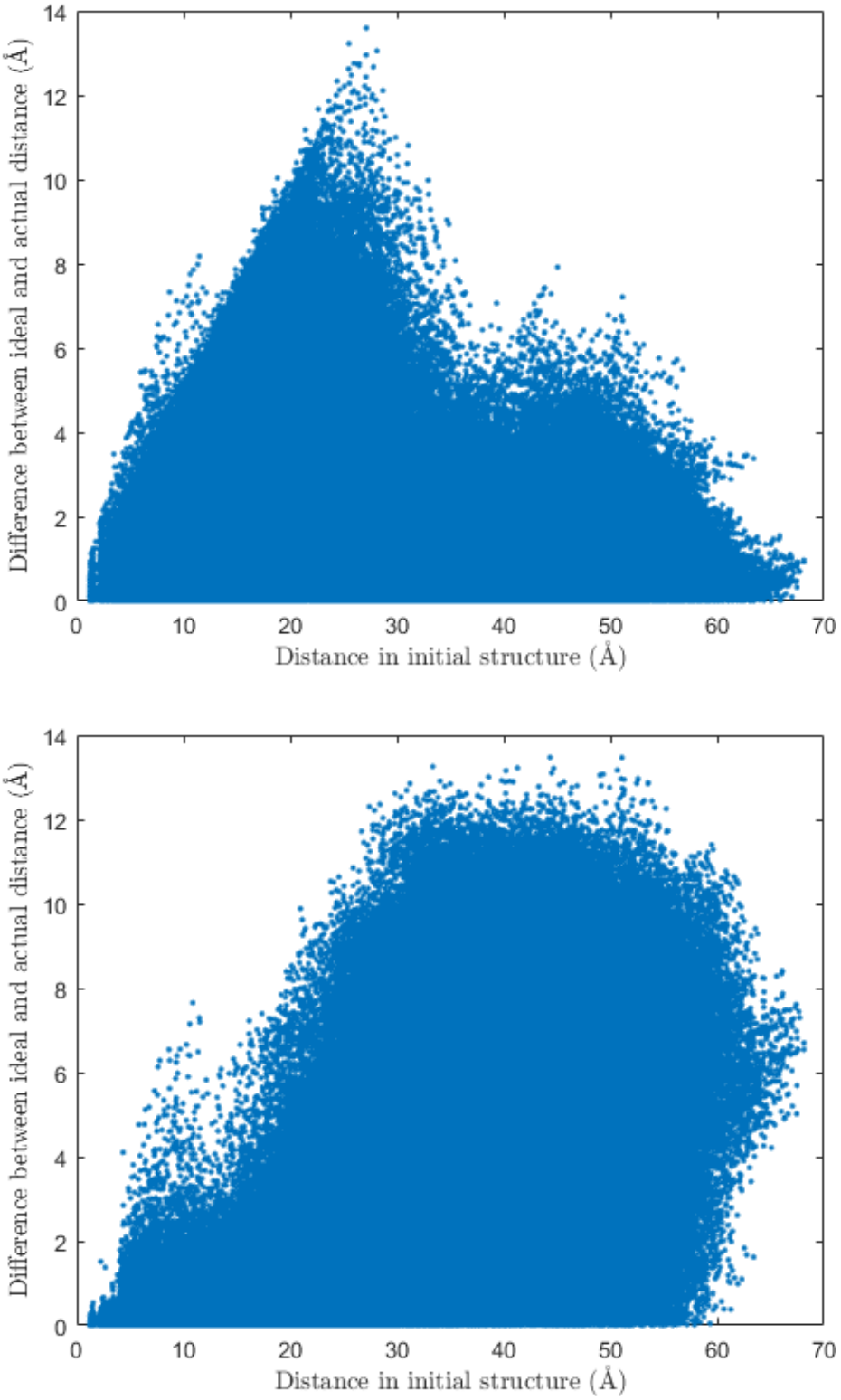


Fig. 4.11 Plot of inter-atomic distance in initial structure against difference between interpolated and actual distance for calmodulin morph (top: classical MDS; bottom: MG MDS cut-off 4Å.

4.4 MolProbity Score

The following sections will consider our set of 100 protein morphs, to identify whether the findings of the calmodulin study apply in these other cases and to gauge the the overall performance of our methods. With a larger dataset, we require a quantitative measure of structural quality. There are many properties of biomolecular structures that should fall within known parameters. As discussed in relation to our preliminary tests on calmodulin, the lengths of bonds and angles between bonded atoms must adhere to known ranges. For our bulk study, the quality of constructed structures as proteins was ascertained using the MolProbity validation tool[29]. This identifies various statistics such as bond angles, clashes, and bond lengths, and also returns a single score which can be used as an overall measure of quality.

$$\begin{aligned} S_{val} = {} & 0.42574 \times \ln(1 + \texttt{clashscore}) \\ & + 0.32996 \times \ln(1 + \texttt{max}(0, \texttt{rotamers} - 1)) \\ & + 0.24979 \times \ln(1 + \texttt{max}(0, 100 - \texttt{ramachandran} - 2)) \\ & + 0.08755 \end{aligned}$$

Where `clashscore` is the number of atoms that overlap by at least 0.4Å per 1000 atoms, `rotamers` is the percentage of side-chain rotamers which have been classed as outliers, and `ramachandran` is the percentage of angles which do not fall within favoured Ramachandran regions. This code snippet is taken from the MolProbity source code[1] with variable names changed for clarity. Higher scores represent worse structures, and the scale of the measure is logarithmic.

[1] https://github.com/rlabduke/MolProbity

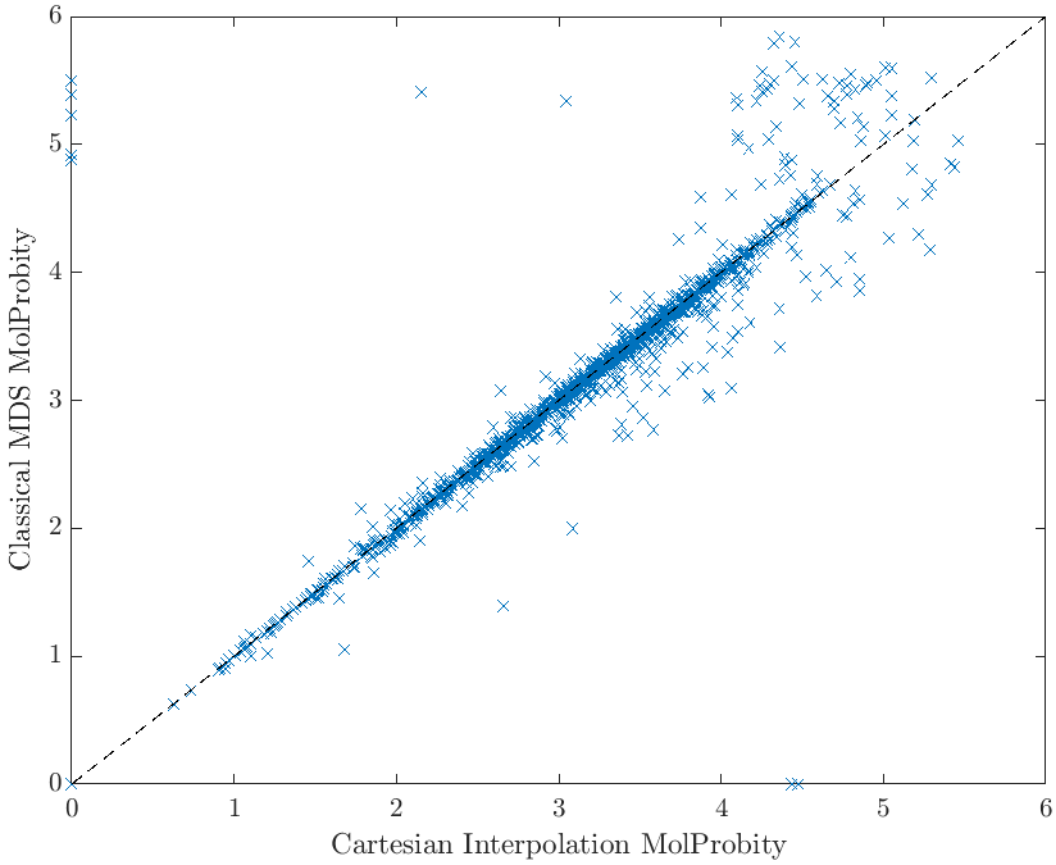


Fig. 4.12 Each morph created using Cartesian interpolation's MolProbity result plotted against the corresponding classical MDS MolProbity score from every frame of the 100 13-frame test morphs.

4.4.1 Classical MDS and Cartesian interpolation

To compare our method with a naive approach, we performed a basic Cartesian interpolation on each of our 100 DynDom examples, as well as creating classical MDS morphs for the same set of inputs.

Figure 4.12 shows a plot of the MolProbity of each Cartesian interpolation against the result of the frame of the corresponding classical MDS morph for every frame of every morph, including the initial and final structures. A dashed black line marks the $x = y$ line; points above this line represent morphs that were poorer using the classical MDS method than

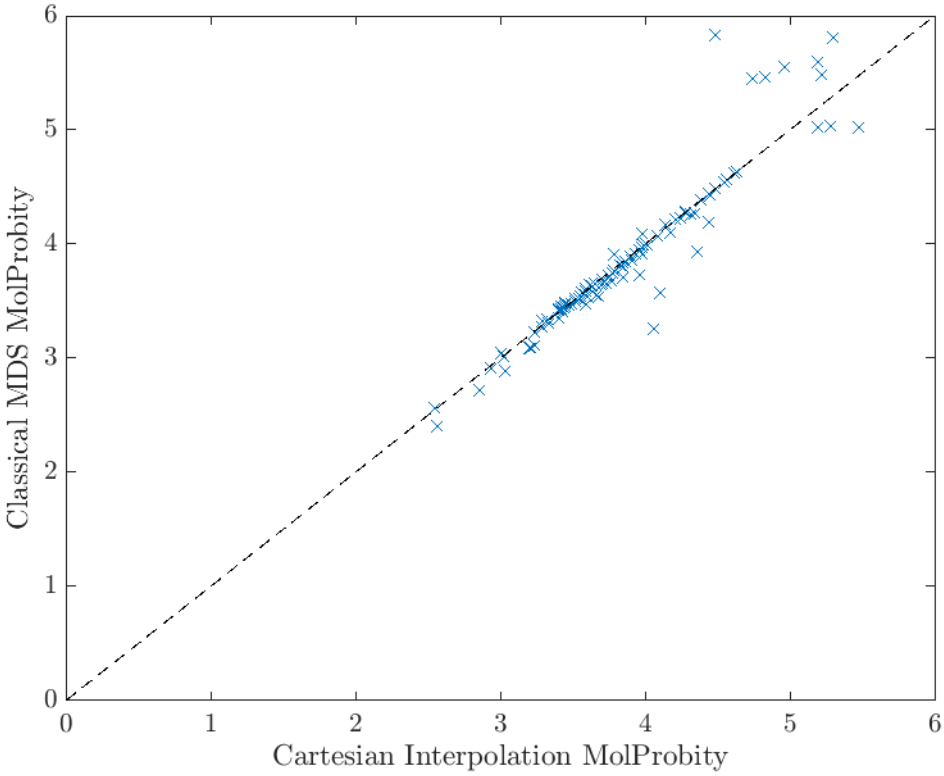


Fig. 4.13 The worst MolProbity score of each Cartesian interpolation morph plotted against the worst MolProbity score of the corresponding classical MDS morph for each of the 100 13-frame test morphs.

the Cartesian interpolation, and points below this line are morphs where the classical MDS morph produced a better structure than the Cartesian interpolation. Most low MolProbity scores were near this line. Towards the poorer scores, there were more differences between methods but not consistently in either method's favour. Paired t-testing shows no significant difference between the two sets of results - the null hypothesis could not be rejected (p-value = 0.6031). Comparing the scores of each intermediate frame, the classical MDS result was on average very slightly poorer, with a mean increase in MolProbity score over the Cartesian result of 0.0171 (with a standard deviation of 0.5058).

Figure 4.13 shows a similar plot, where for each morph only the worst-scoring frame was included. Again, the methods performed similarly, with no strong preference either way.

Paired t-tests failed to reject the null hypothesis with a p-value of 0.9841. The classical MDS results for the worst frames in each morph were on average very slightly better than their corresponding Cartesian peaks, with a mean improvements of 0.0018 (std=0.2319).

4.4.2 Selecting the cut-off radius

Modern MDS methods allow a weighting to be applied to particular target distances when calculating the loss resulting from a constructed set of coordinates. We apply a weighting to atom pairs based on their distance in the solved conformations; if the atoms are above a distance cut-off, then their weight is set to 0, otherwise the pair's weight is set to 1. We sought a constant value for the distance cut-off that would most often provide the best maximum MolProbity score. The cut-off was chosen to restrict the interatomic distances considered in MDS to each atom's closest pairs, which we found important for maintaining appropriate bonds lengths and angles, while also ensuring that all atoms have enough considered distance pairs to recreate their position within the protein. To select a value for this cut-off distance we morphed our sample of 100 proteins taken from the database of protein domain movements on the DynDom server. A series of cut-off distances were tried, between 2Å and 10Å at an interval of 0.5Å. A short thirteen-frame morph was produced for each protein at each cut-off distance, and for every frame a MolProbity score was calculated to determine the plausibility of the structure as a protein.

The MolProbity scores obtained from the MDS method have been compared against those achieved by MolMovDB, such as their result for calmodulin shown in Figure 4.16. The effects of varying the chosen cut-off radius on the resulting MolProbity score for our example calmodulin are shown in Figure 4.14. In order to find the cut-off distance which gives the most consistently good performance, we took the highest (worst) MolProbity score from each morph as its representative. The worst MolProbity scores achieved at each cut-off distance are plotted in Figure 4.15, which shows a minimum at 4Å.

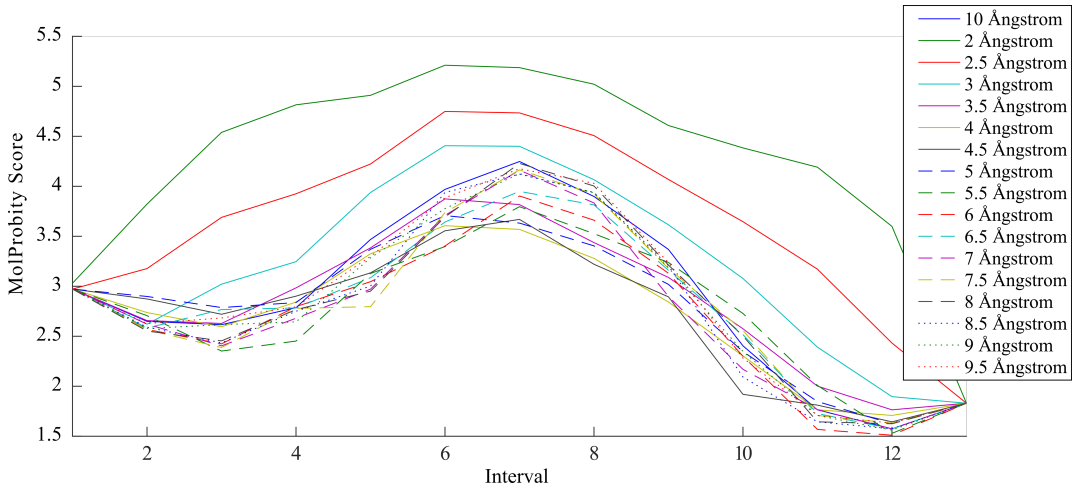


Fig. 4.14 The effect of cut-off radius on MolProbity score running on Calmodulin (Protein Data Bank IDs 1CM1 and 1CLL).

These peak MolProbity scores from each morph were compared against the peak scores achieved by other morphs created from the same initial and final structures, identifying a cut-off distance at which the worst MolProbity score was lowest (and therefore gave the least implausible structure). The proteins' optimal cut-off distances were calculated and then tallied. Plotting the frequencies of proteins which have the best peak score at each cut-off identified that, as in the calmodulin example, a cut-off of 4Å most commonly gave the best results as shown in Figure 4.17.

We re-ran this experiment with the same 100 proteins, focusing only on the 3.5Å to 4.5Å cut-off distances with a smaller interval of 0.1Å, which again identified 4.0Å as the most commonly optimal cut-off distance.

4.4.3 Influence of weighted MDS

In the calmodulin example, weighting the inter-atomic distances such that only small distances affect the MDS result improved the resulting intermediate structures. These improved structures were both easier to follow due to a lack of atomic clashes and much more plausible according the MolProbity validation tool. We performed both classical MDS and weighted

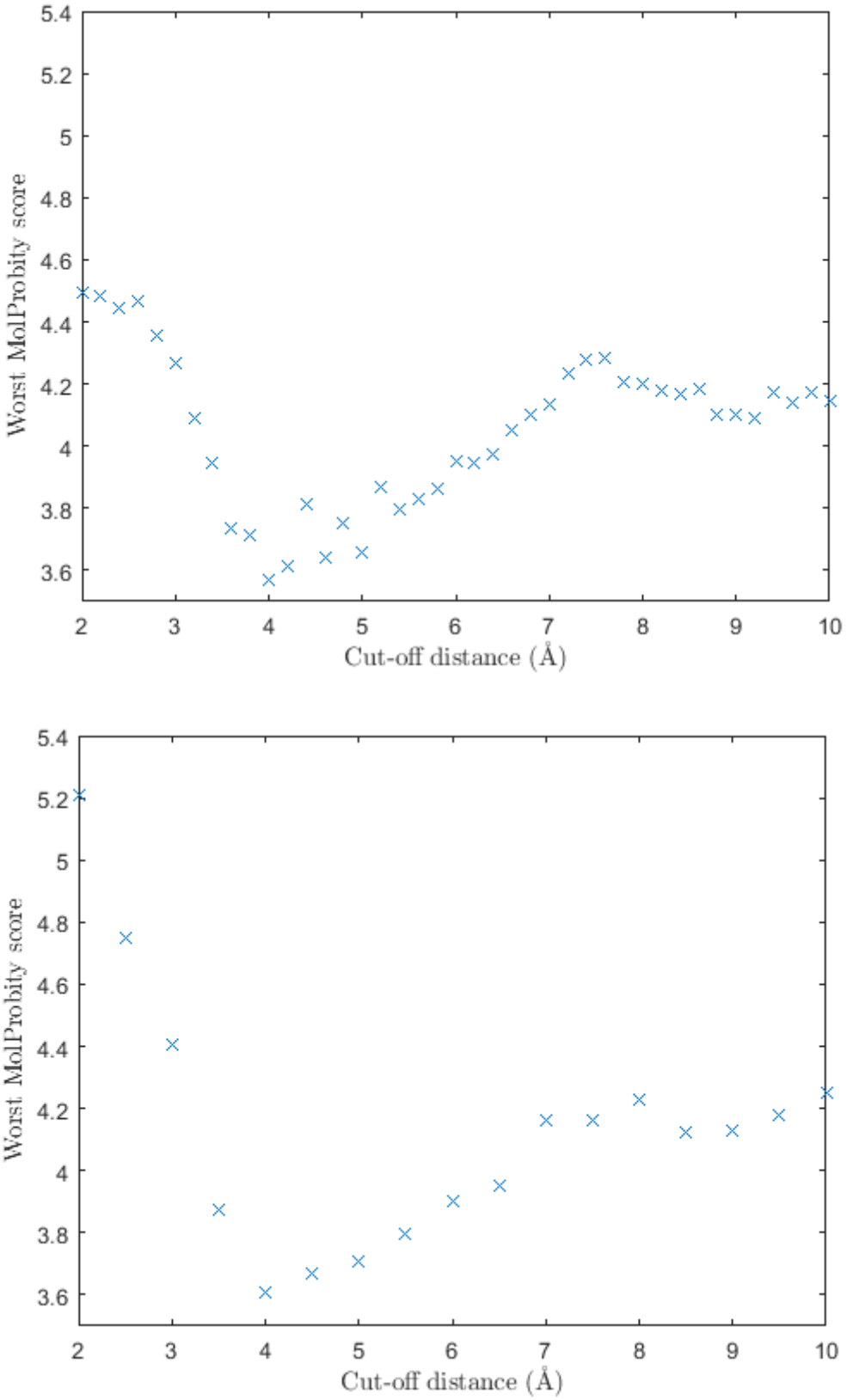


Fig. 4.15 The worst MolProbity score achieved by metric MDS morphs of calmodulin across a range of cut-offs (top: metric MDS; bottom: multi-grid SMACOF).

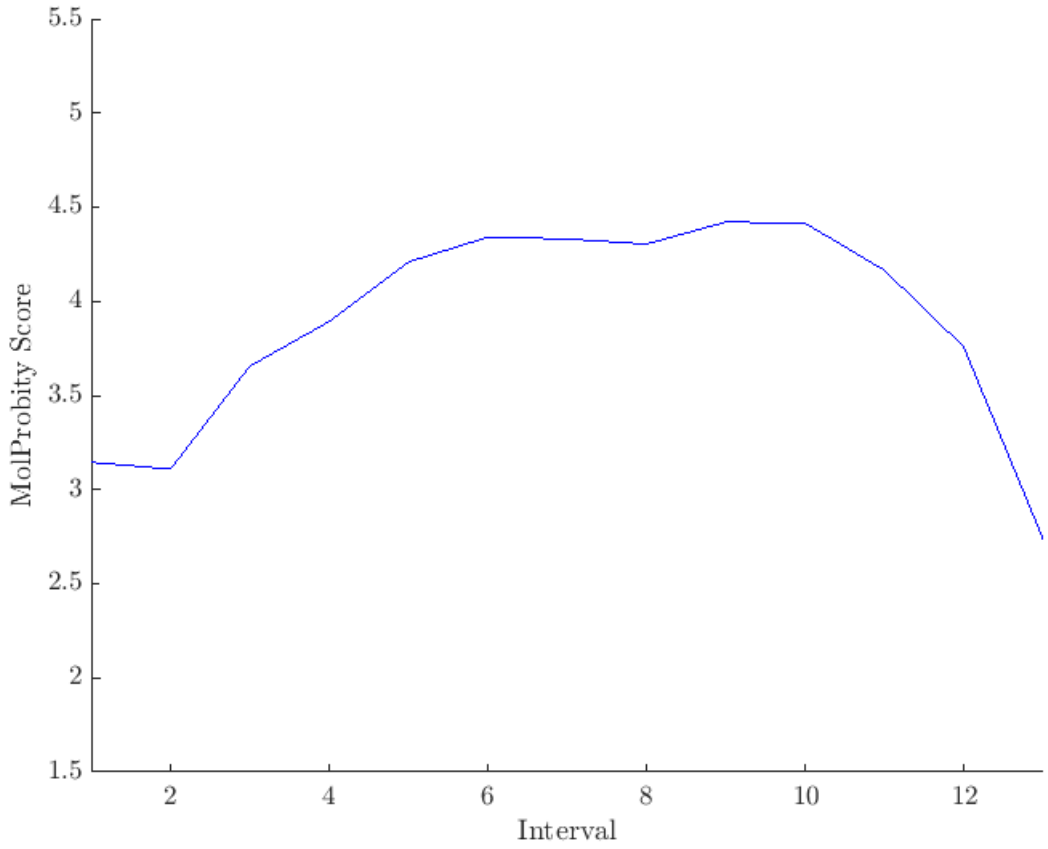


Fig. 4.16 The MolProbity result from the MolMovDB web server running on Calmodulin (Protein Data Bank IDs 1CM1 and 1CLL).

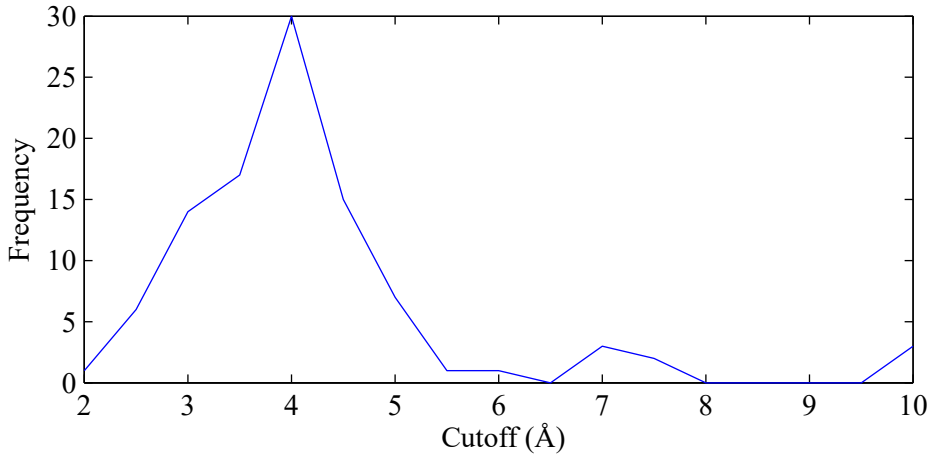


Fig. 4.17 From 100 protein morphs tested, the amount of times each potential cut-off radius between 2 and 10 yielded the best MolProbity peak.

SMACOF MDS on each of our 100 DynDom test cases using the GPU-accelerated multi-grid method for computational performance. The coarser layers could not use the 4Å cut-off as this would leave atoms without enough weighted distance pairs to anchor their position in the protein so we applied a linear increase in cut-off distance between layers. This had little impact on the MolProbity results when compared to the original SMACOF results provided the cut-off was not very small (3Å or smaller). Figure 4.15 shows the MolProbity values for calmodulin using the multi-grid and original SMACOF methods. The results of each of the multi-grid morphs, using the selected optimal weighting values of 4Å, 8Å, and 12Å for each layer, were compared to initial results using only classical MDS. One morph (PDB code 1DF1, chains A, to chain B of the same PDB file) was edited for both sets of results to remove a misaligned atom that caused the MolProbity attempts to fail even for the solved structures; other MolProbity runs that failed due to problems created by the morphs are represented with a score of 0.

Figure 4.18 shows the MolProbity of all frames, with the x-axis showing the classical result and the y-axis showing the multi-grid result. A dotted line represents the $x = y$ - the case where both methods achieved the same result. The majority of cases achieved a lower (and therefore better) result for the weighted multi-grid version of the method. A paired t-test of the MolProbity scores rejected the null hypothesis with a p-value of 9.26×10^{-20}, showing a significant separation between the sets of results. Removing the scores from the solved structures, which must be the same for each method, the multi-grid method achieved a mean improvement over the classical method of 0.3676, with a standard deviation of 0.6541.

Figure 4.19 shows the same plot, but each point represents the worst frame of the morph. Again, most cases are on or below the dotted $x = y$ line, showing an improvement of the multi-grid method over the classical method. Paired t-testing rejected the null hypothesis with a p-value of 5.23×10^{-4}. The mean improvement achieved by the multi-grid method was 0.3509, with a standard deviation of 0.5389.

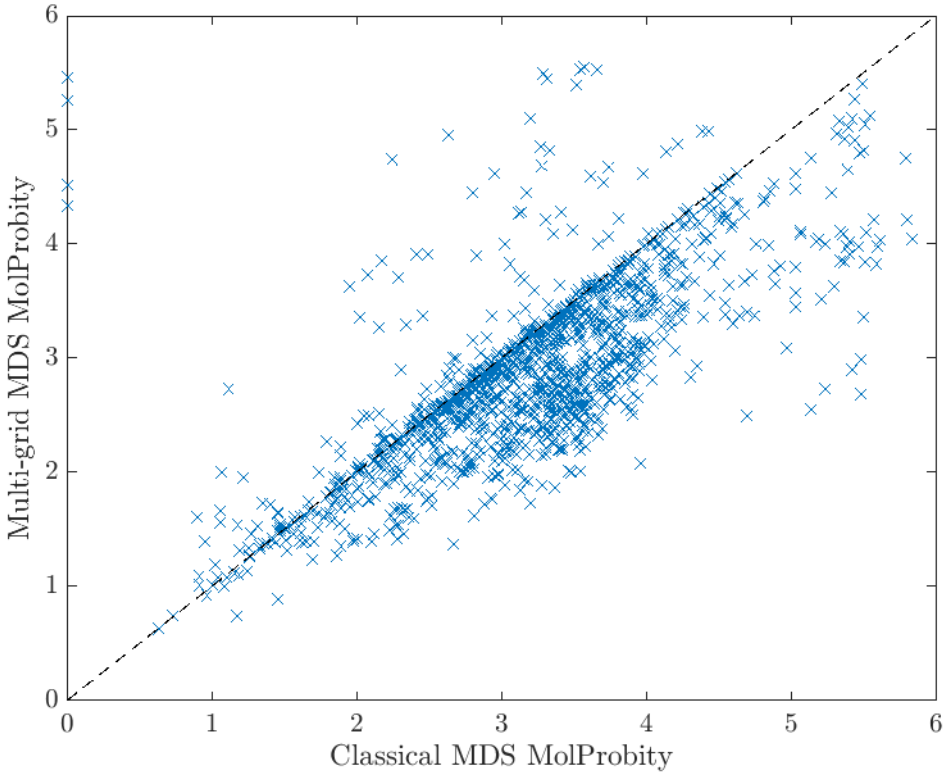


Fig. 4.18 Each classical MDS MolProbity result plotted against the corresponding 4Å multi-grid SMACOF MDS morph MolProbity score from every frame of the 100 13-frame test morphs.

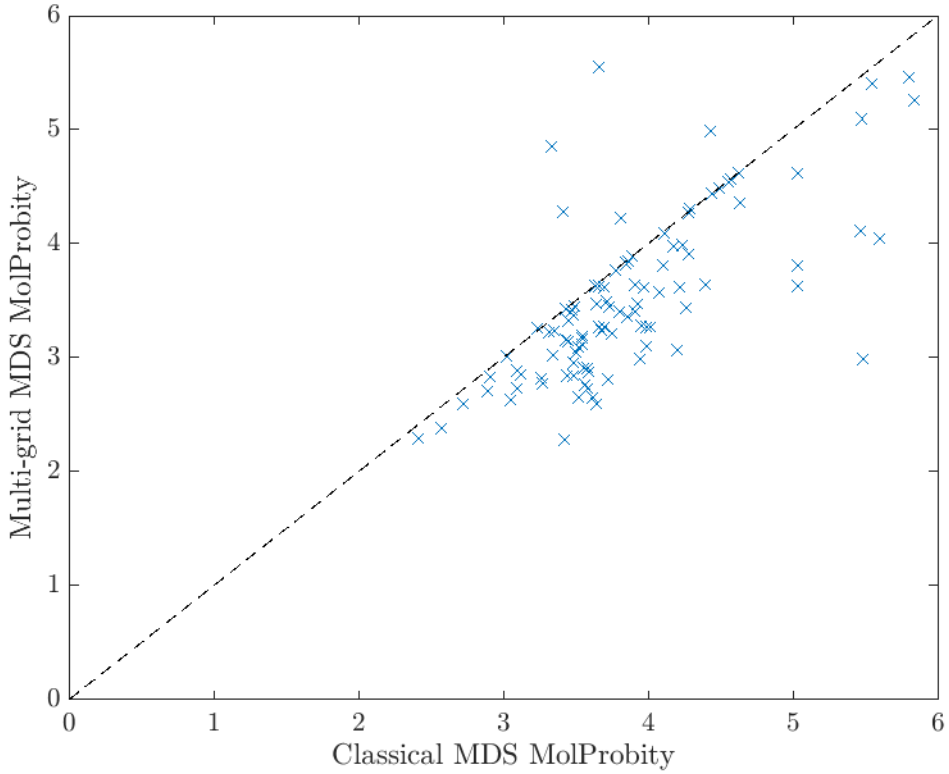


Fig. 4.19 The worst MolProbity score of each classical MDS morph plotted against the worst MolProbity score of the corresponding 4Å multi-grid SMACOF MDS for each of the 100 13-frame test morphs.

4.4.4 Alternative weighting patterns

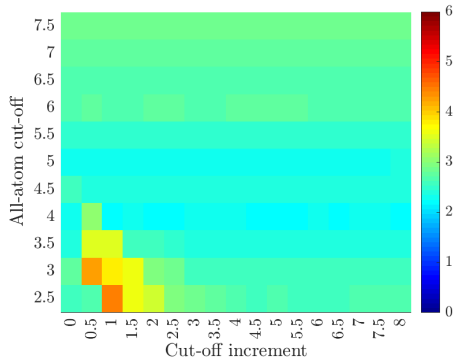
The layers of the multi-grid method have a linearly increasing cut-off; the all-atom layer has a cut-off of 4Å, which increased to 8Å for the backbone layer and 12Å for the Cα layer. Alternative increments for the cut-off were explored both for our difficult test case calmodulin, and for each of the 100 test morphs.

With the 100 morphs in our DynDom dataset, we again focussed on the central frame. Figure 4.20 shows eight example heat maps displaying the MolProbity scores for the central frame of each morph. Where Figure 4.20(c) has a MolProbity score of 0, the structure was too poor to return a score at all. Most of the morphs showed a similar pattern: very short initial cut-off lengths with small increments obtained the poorest results, but for initial cut-off lengths of roughly 4.5Å or more, there is little change in MolProbity score when the gradient is varied. The overall minimum was commonly seen around the 4Å initial cut-off found by the linear optimisation. Where the results dramatically differed from this pattern, as in Figure 4.20(d), all MolProbity scores were poor (above 5).

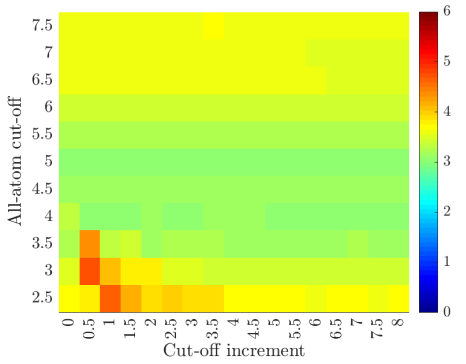
4.4.5 Smoothness of Motion

The smoothness of motion from one conformation to the other was a key priority for this implementation. The MDS method does this as when taking an interval $\lambda = 0.5$ halfway along the timeline, the RMSD from the start model to the model at λ and the RMSD from the final model to the model at λ are both likely to be close to 0.5. This shows a smooth transition from one state to the other with its midpoint roughly halfway through the morph. Figure 4.21 shows the progression of RMSDs from one structure to the other for all 100 morphs.

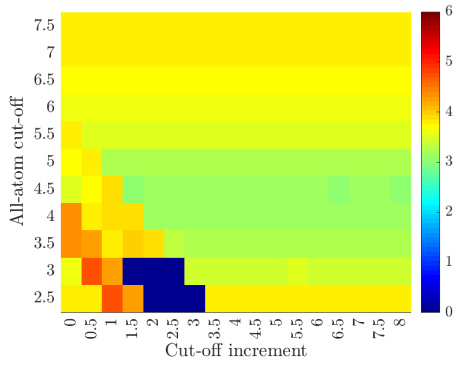
Comparing the plot of RMSD against time from the MDS method against other morphing methods shows that this is a particular feature of our method. Figure 4.22 shows a comparison



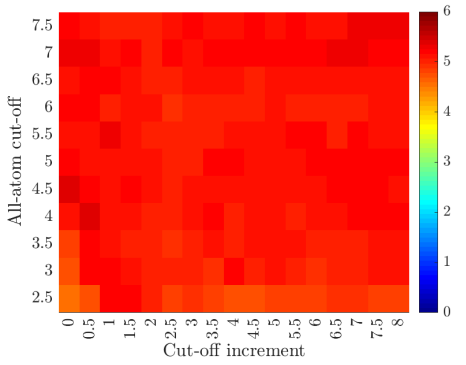
(a) Aac(6')-II



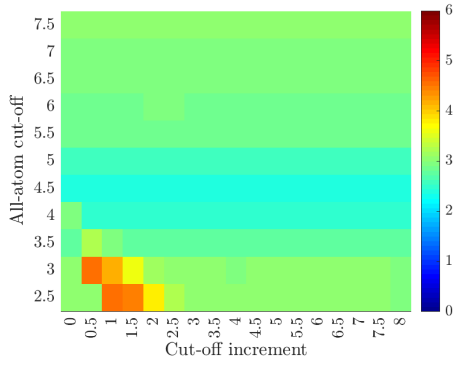
(b) Adenylate cyclase



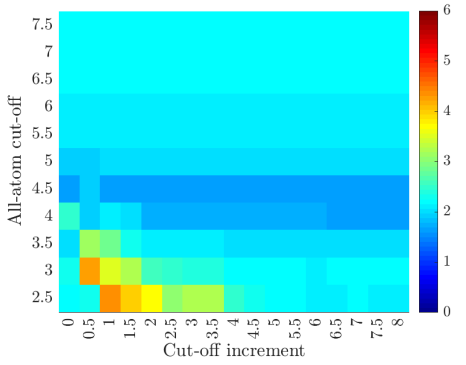
(c) Exocyst complex component Exo70



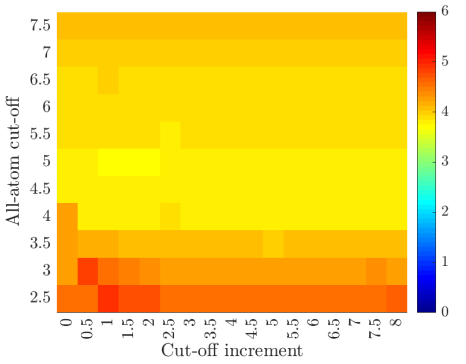
(d) Igg2a fab fragment



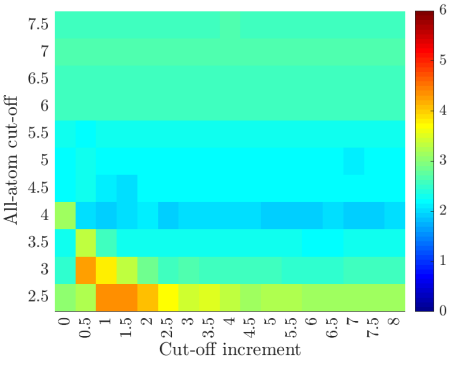
(e) D-glyceraldehyde-3-phosphate dehydrogenase



(f) GluXyn-1



(g) Helicase pcrA



(h) Electron transfer flavoprotein alpha-subunit

Fig. 4.20 MolProbity scores for the central frame of examples from the 100 DynDom morph examples.

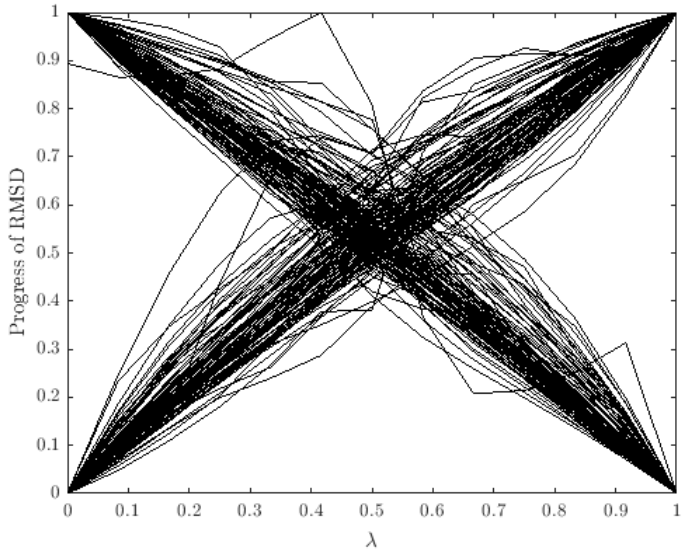


Fig. 4.21 Percentage of maximum RMSD from start and end structures against progress for each of the 100 proteins in the sample.

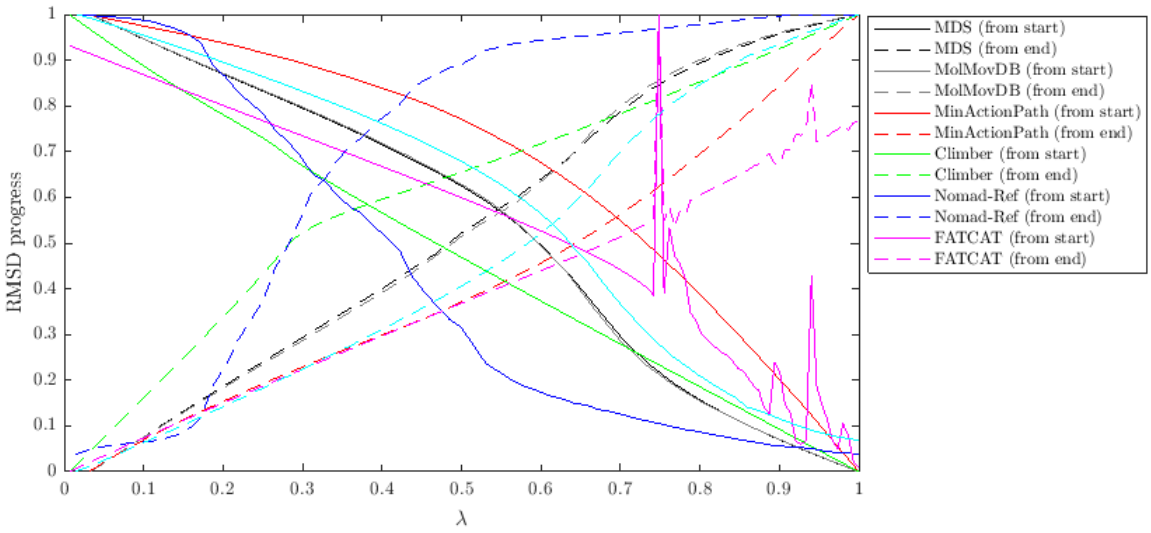


Fig. 4.22 RMSD from start (solid) and end (dashed) models in each domain for a morph of calmodulin.

between the changes in RMSD for a protein morph of Calmodulin (PDB files 1CLL to 1CM1). Our results show that as the RMSD from the start conformation increases, the RMSD from the end structure decreases, resulting in a smooth crossover from one structure to the other. The MorphServer, Climber, FATCAT, and MinActionPath methods cross further from the

central $x = 0.5,\ y = 0.5$ position. The Nomad-Ref result is showing jagged peaks where it was unable to smoothly morph part of the trajectory.

4.4.6 Comparison with Known Intermediate

Weiss and Levitt [168] propose a measure of quality for protein morphs based on the similarity between frames along the produced path and a solved intermediate structure (B) that lies between the two end conformations (A and C). RMSD values are calculated between each model in the morph and the intermediate structure, represented as rmsd(iB). An improvement score is calculated from:

$$\frac{\text{min[rmsd(AB),rmsd(CB)]} - \text{min[rmsd(iB)]}}{\text{min[rmsd(AB),rmsd(CB)]}} \times 100 \tag{4.3}$$

Thus the improvement score is a percentage between 0% and 100%, where a higher scoring morph will pass much closer to the intermediate frame than to either of the initial solved structures that were used to create it.

In addition to proposing this metric, Weiss and Levitt [168] identify five cases where as well as solved structures at the start and end of the conformational change, a third structure has been solved at an intermediate stage.

Figure 4.23 shows the improvement score achieved by each method on each of the 5 test cases. Both the all-atom multigrid MDS with a 4Å cut-off (MDS_4) and a Cα-only SMACOF MDS with a 12Å cut-off (MDS_CA12) were included. As there is no single method that outperforms all other methods in every example, we looked to ranking systems like the Condorcet system of voting [20]. In Condorcet voting, a comparison is performed between each pair of candidates' votes. A candidate is elected if they receive more votes than each of their competitors in the pairwise comparisons. We created Table 4.1, in which each cell contains the number of times the method in the row outperformed the method in the

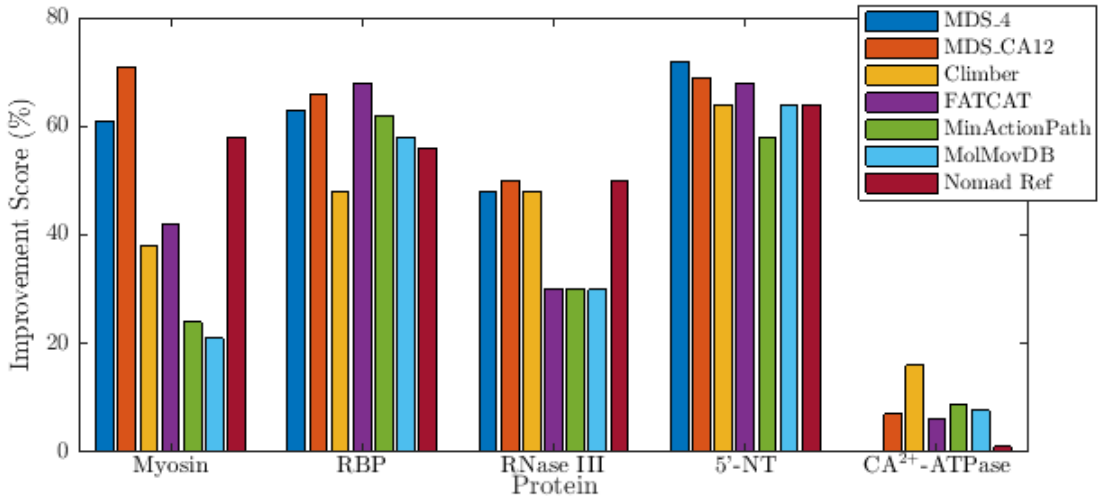


Fig. 4.23 A comparison of the performance of the MDS morphing technique compared with competing methods using the measure of improvement proposed by Weiss and Levitt[168].

column on the five test cases. We found that both versions of the MDS method outperformed the other methods on more test cases than they were outperformed on.

4.4.7 Performance

The suitability of protein morphing techniques relies on striking a balance between accuracy and speed, providing results to the user ideally within minutes. Table 4.2 lists the average number of seconds taken to produce each frame for five 24-frame morphs where the start and end structures were selected for their range of sizes. The SMACOF morphs were produced using a fixed 20 iterations. The multigrid method always used 16 iterations per cycle, but the number of cycles varied; the method was run for each until either three cycles were complete or the stress reached the stress calculated at the final step of the SMACOF run.

All calculations were performed on the same machine. CPU operations were performed on an Intel Core i7 870 @ 2.93 GHz processor, with 16 GB RAM. GPU versions of the SMACOF and multigrid methods were performed on an NVIDIA Titan X card.

	MDS_4	MDS_CA12	Climber	FATCAT	MinActionPath	MolMovDB	Nomad-Ref
MDS_4	-	1	3	3	4	4	3
MDS_CA12	4	-	4	4	4	4	4
Climber	1	1	-	2	4	3	1
FATCAT	2	1	3	-	3	3	3
MinActionPath	1	1	1	1	-	3	2
MolMovDB	1	1	1	1	1	-	2
Nomad-Ref	2	0	3	2	3	2	-

Table 4.1 A representation of pairwise comparisons of protein morphing techniques' improvement score.

The following PDB structures and chains were selected: the pyruvate kinase structures used were PDB codes 1E0T (chain A) and 1E0U (chain A); glycogen phosphorylase b 1GPB (A) and 1GPA (C); hirustasin 1HIA (I) and 1BX7(A); the heavy chain of Fab frag 7G12 1N7M (H) and 1NGY (A); and phosphoenolpyruvate carboxykinase 2RKA (C) and 2QF2 (A). The morphs cover a range of small to medium rotations, from 11.1° to 60.6° . All results include an initial classical MDS stage for each frame. All results include the initial classical MDS step to get the starting coordinates for each frame.

The SMACOF operation contains many repetitions of matrix operations which are particularly suited to efficient calculation on the GPU [47], which is reflected in the results listed in Table 4.2. The multigrid method in particular benefited from this GPU acceleration. The performance of the CPU-based multigrid method appears to be poorer than the CPU-based SMACOF method. This is potentially in part due to additional calculations performed when restricting and interpolating that are not required by the basic SMACOF algorithm. It may also be a result of the chosen number of cycles and iterations, as identified when the experiment was repeated on the Fab frag 7G12 alone. With the selected 20 iterations of the

Protein	No. atoms	Average MDS runtime per frame (seconds)			
		SMACOF	SMACOF (GPU)	Multigrid	Multigrid (GPU)
Hirustasin	350	0.09	0.06	0.17	0.61
Fab frag 7G12	1,638	2.92	0.67	5.32	0.71
Pyruvate kinase	3,300	13.89	4.73	23.40	1.93
Phosphoenolpyruvate carboxykinase	4,844	33.60	14.16	64.14	4.23
Glycogen phosphorylase B	6,656	74.65	35.95	112.17	8.83

Table 4.2 A comparison of the runtimes of implementations of SMACOF and multigrid MDS on the CPU and GPU.

basic SMACOF method, as with 30 and 40 iterations in the repeated experiment, the final stress reached was achieved within the first cycle of the multigrid method. This resulted in an unchanged performance for the multigrid method compared to a poorer performance for the SMACOF algorithm.

4.4.8 Lambda Optimisation

As described earlier in the chapter, during the interpolation of inter-atomic distances (Equation 4.2) we assume that our interpolation function λ operates linearly with the time step, t, for every atom pair. This is not likely to be the case during an actual morph, as shown by the stress, which would be zero at every frame if the interpolated distance matrix were perfectly possible. Some investigations were made into whether it would be possible to optimise this function by minimising a cost function for a single morph: either the MolProbity score or the stress resulting from the morph. However, it was found that the computational expense of optimising a λ value for every weighted atom pair in every frame quickly became untenable even for small substructures of proteins.

4.4.9 Energy Minimisation

We considered whether we could further improve the results of the MDS morphs by performing on the intermediate frames Energy Minimisation (EM) using GROMACS. This would resolve any clashes or unrealistic bond lengths and angles that were causing implausibly high energy. The EM process is affected by missing or incomplete residues, so a pair of zinc peptidase structures, chains A and C in PDB 1AMJ, were chosen as a member of the available DynDom motion pairs that did not require any reconstruction. Figure 4.24 shows the MolProbity scores resulting from the original multi-grid MDS morph, and from MDS with EM applied at various different stages. Methods which did not end on a round of EM performed poorer, and the most consistently low MolProbity score was achieved when the MDS was both preceded and followed by EM. This result is not surprising as properties that will raise a protein's MolProbity score would be targeted by the EM. However, the additional processing time and the requirement that structures be complete detracted from the advantages of the MDS method, which on its own is fast and can accept any coordinate set.

4.5 Docking Visualisation

In this chapter, conformational changes have been described as the motions of a single molecule operating in isolation, but these changes are often the result of the interaction between multiple biomolecules. These conformational changes may accompany the interaction between an enzyme and a substrate, or the binding of a protein ligand to a protein receptor.

Section 3.3 discusses the lack of comparable available servers; visualisations offered typically either use a linear Cartesian interpolation or protein morphing within each component without considering the location of the other component's atoms. We looked to the related problem of docking prediction for insight in the requirements of docking visualisation. From two unbound structures, docking prediction methods seek to predict the formed complex.

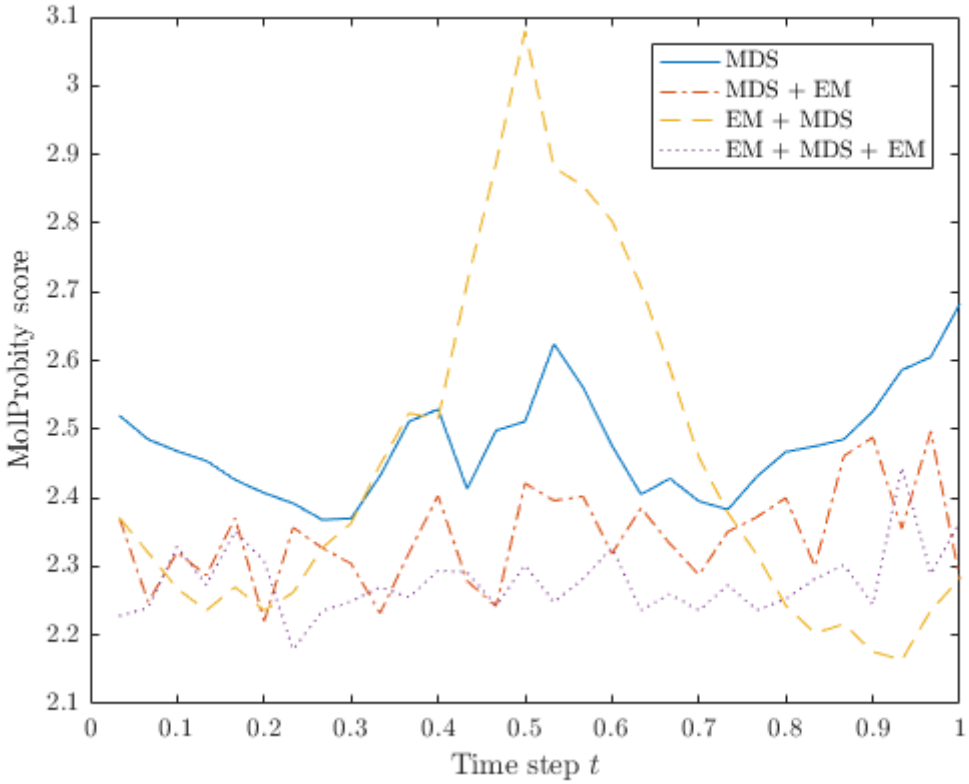


Fig. 4.24 Impact of energy minimisation on the MDS morph process when applied at different stages: the MDS results without EM, MDS with EM applied after each step, EM performed on the start and end structures before MDS, and EM performed on both the starting structures and the resulting frames.

The key finding of this literature review is that recent docking prediction methods are attempting to achieve higher levels of flexibility in proteins to improve their accuracy due to conformational changes on binding. It is hoped that our tailored docking morph method has the potential to provide demonstrative visualisations that may help docking teams investigate these interactions.

Morphit_Pro's docking server produces animations showing each constituent moving into its docked configuration as well as intramolecular conformational changes. As with protein morphing for the visualisation of conformational changes, docking visualisation is useful in that it allows the user to track movement they might otherwise have missed by showing each atom move from start to end, and shows areas where clashes occur or where parts of the proteins would have to move out of the way.

The docking morph takes as its input three PDB files containing the coordinates of protein or DNA / RNA molecules. The user then indicates which chains of the bound structure should be considered the receptor and which chains are part of the ligand, and highlights these chains in the unbound ligand and receptor structures. In the output trajectory, the receptor will be held stationary while the ligand is translated towards it.

In order for the morph to focus on the docking motion, the ligand and receptor are placed in an initial structure such that only a small translation is required to bring them together. This initial positioning, the "near-approach pose", is created by superposing each component in its unbound conformation onto its corresponding molecule in the bound structure, and then translating the ligand until no atom of the ligand is within 4Å of any atom of the receptor.

However, these near-approach poses do not represent a known real structure. When building the weighting matrices for our multi-grid MDS we take only the inter-atomic distances from solved structures; intra-molecular distances in the near-approach pose, and both intra- and inter-molecular distances from the experimentally derived native complex structure.

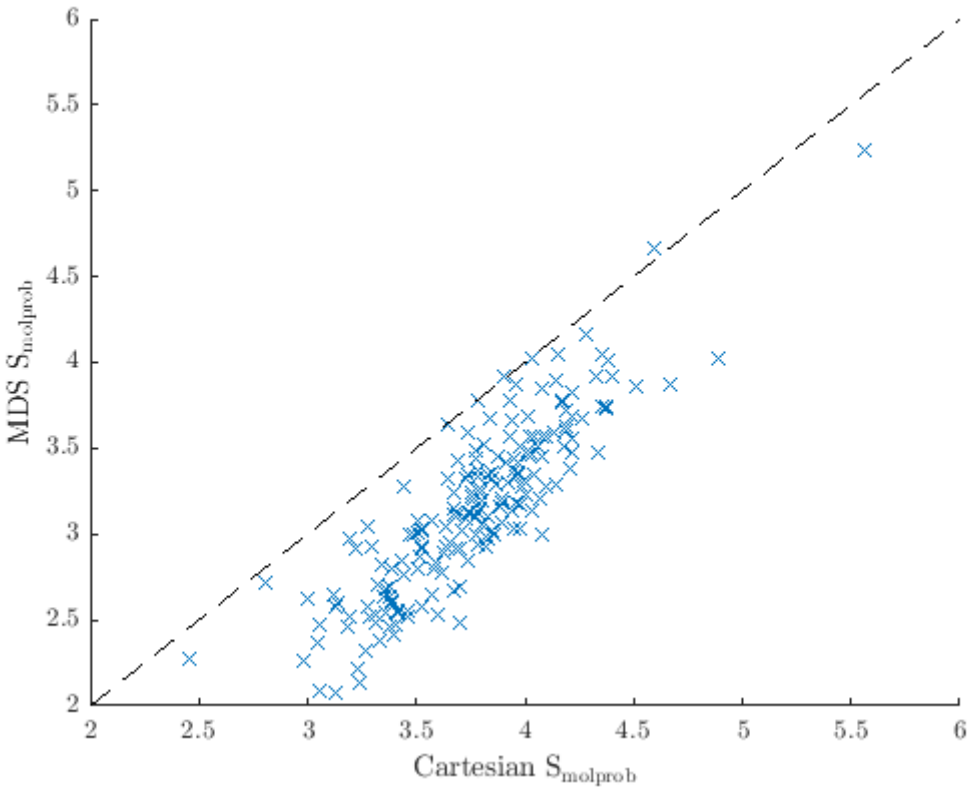


Fig. 4.25 MolProbity results for Cartesian and MDS morphing on 189 docking cases taken from the Protein-Protein Docking Benchmark 5.0. From each morph, the MolProbity score is the worst score yielded by any frame's structure.

We created docking visualisations for 189 cases from the Protein-Protein Docking Benchmark 5.0 [164]. Some examples had to be removed from the dataset due to memory limitations; the input to the multi-grid MDS method is restricted as the GPU's memory is lower than typical allowances for the CPU.

Due to the lack of comparable methods available online, we implemented a Cartesian interpolation of the same data for comparison. The maximum MolProbity score for each Cartesian result against the MolProbity score for the corresponding maximum peak of the MDS morph is shown in Figure 4.25. Almost all of the MDS morphs achieved a lower MolProbity result than their Cartesian counterparts. There were just two examples where the

Cartesian interpolation performed slightly better (by 0.03 and 0.06) than the MDS method. A paired t-test indicated that this separation is significant, with a p-value of 1.48×10^{-32}.

The difficulty level set by the Protein-Protein Docking Benchmark 5.0 appears to be correlated with the MolProbity result. The benchmark divides its test cases into three categories. In the lowest difficulty category, "rigid body", the Cartesian method achieved a mean peak MolProbity score of 3.67, with a standard deviation of 0.365, while the MDS method achieved a mean peak of 3.04 with a standard deviation of 0.473. On the middle category, "medium difficulty", the Cartesian interpolation had a mean score of 3.90 (standard deviation 0.294), and the MDS method's mean score was 3.34 (standard deviation 0.300). Finally, for cases in the "difficult" category the Cartesian and MDS methods achieved mean MolProbity scores of 4.07 (0.741) and 3.55 (0.580) respectively.

4.6 Web Server

The method is available on a web server at http://morphit-pro.cmp.uea.ac.uk/MorphItPro/. The single structure morphs and the docking morphs are both available. Figure 4.26 shows the input form for the single structure morphs. Start and end structures are chosen as PDB files, and can either be uploaded by the user or selected from a mirror of the PDB database using the four-letter PDB code. Morphs that are created using only files from the mirrored PDB repository, or where the "include in database" option is selected, are stored in a database that is also accessible on the web server. After the user has selected PDB files, they are parsed by the server and displayed to the user in the JSmol [66] viewer, so that the user can select the chains on which the morph can be performed. Morphs can be between 3 and 30 frames, and either all-atom or Cα only. Completed morphs are presented to the user through the JSmol viewer, with options on the results page for downloading a PDB file containing a model for the structure in each frame.

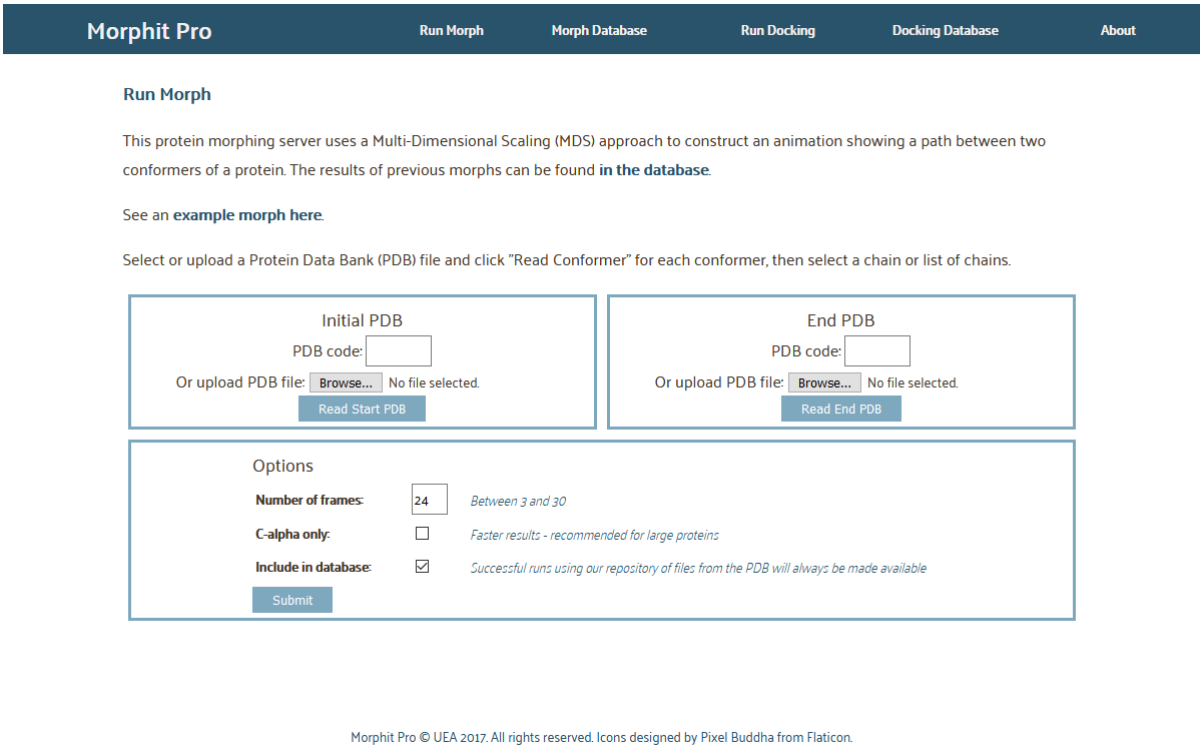


Fig. 4.26 A screenshot of the input form for the single structure morphs in the Morphit Pro web server.

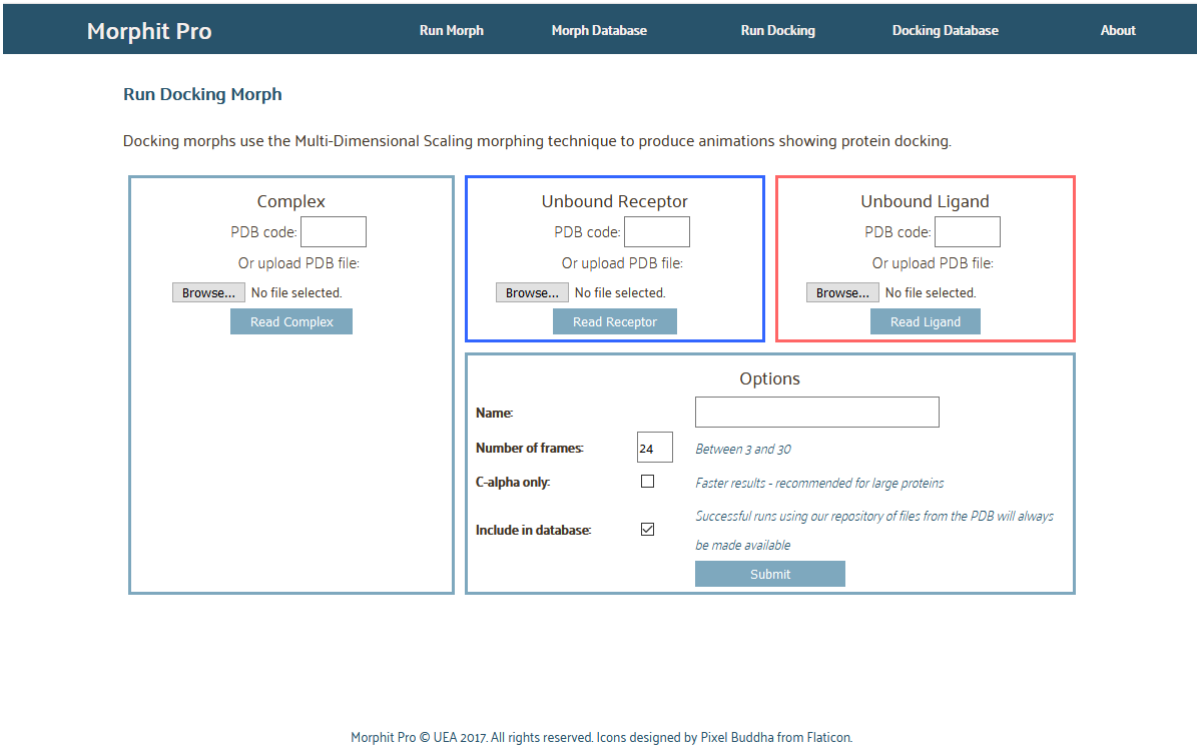


Fig. 4.27 A screenshot of the input form for the docking visualisation morphs in the Morphit Pro web server.

Figure 4.27 shows the input form for the docking visualisation version of the morph. Three PDB files are entered: the bound complex containing both the receptor and the ligand, the unbound receptor and the unbound ligand. The results are shown on a similar JSMol viewing page with additional options for showing the ligand and receptor structures separately.

The web server was implemented in JavaScript and Java Server Faces 2.0 using Facelets. The morphs are performed using a single Nvidia Titan X GPU, so requested morph jobs are placed into a queue and performed one at a time. The 30-frame limit is intended to keep jobs moving quickly through the queue, and a limit of 10,000 atoms in a structure is imposed due to memory limitations of the GPU.

4.7 Molecular Dynamics and MDS Morphing

4.7.1 Molecular Dynamics

We performed classical MD simulations of the PD-1 structure (with PDB code 2M2D) and its complex with nivolumab (PDB code 5GGR, shown in Figure 4.28). We constructed the near-approach pose as is done in the MDS morph method, by superimposing unbound versions of each structure over their positions in the complex and then moving the nivolumab fragment away along the line connecting the two molecules' centres of mass until there was a minimum distance between the two of 4Å.

The nivolumab molecule contains two chains, between which the PD-1 is bound. To reduce the number of atoms in the structure, each chain was cut in half and the section of each chain furthest from the binding sites were removed. The ends of the chains were capped with acetyl and n-methyl residues to prevent unnatural interactions involving the newly-created terminals. The structure contains some missing atoms in the flexible loop; this missing section was also capped.

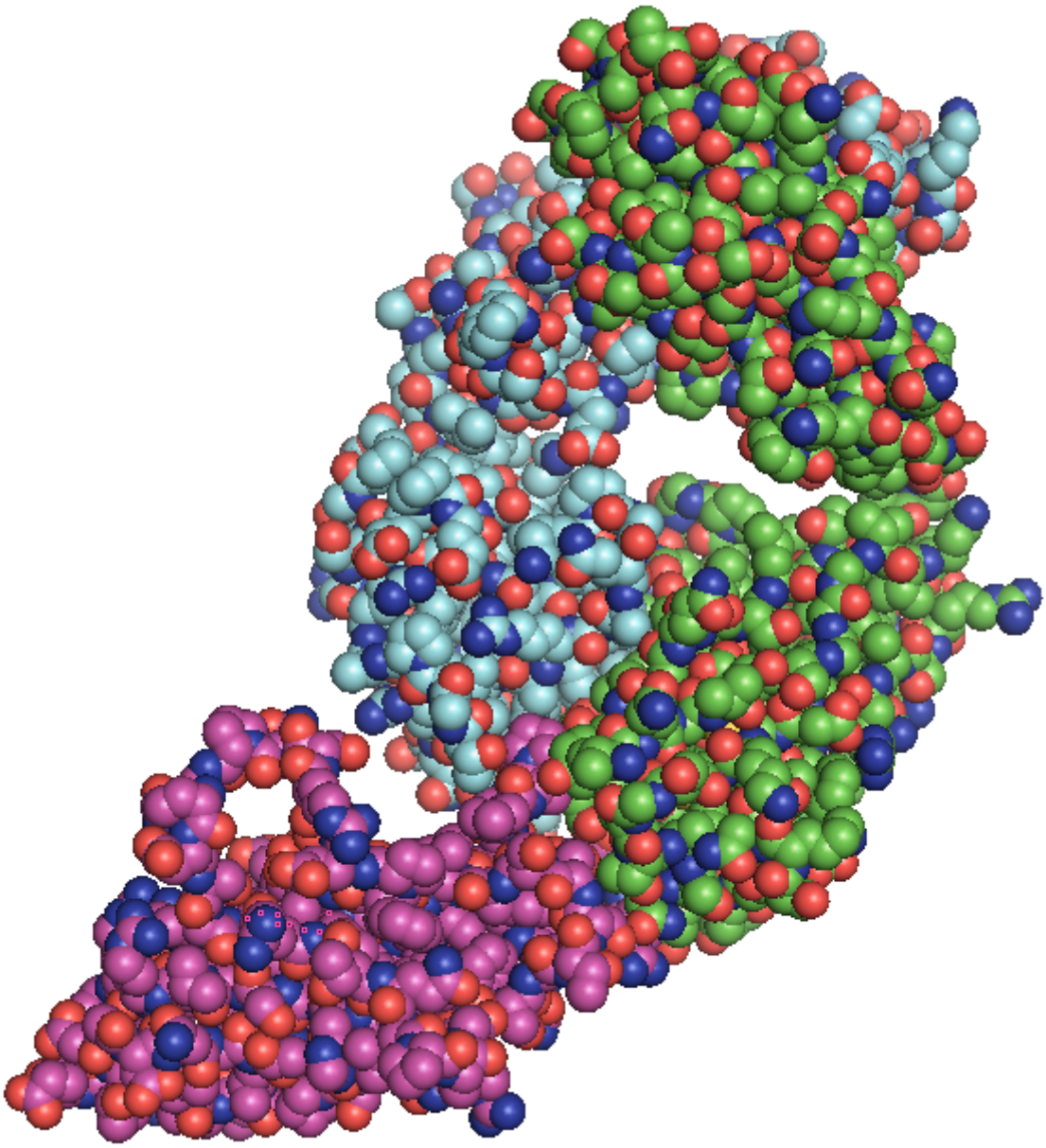


Fig. 4.28 The complex formed by PD-1 (the pink chain) and nivolumab (the cyan and green chains) with PDB code 5GGR.

The AMBER 99SB-ILDN forcefield was selected for all simulations.

Our initial MD simulations ran from the constructed near approach pose with no attempts to induce docking, to investigate whether simulating from the near-approach pose alone was enough to simulate a native or near-native complex. We found that this was not the case, as the nivolumab would immediately bind to the nearest part of the PD-1 in a non-native position and become "stuck". This may indicate that the enforced 4Å minimum distance in the near-approach pose should be longer if the method is to replicate the native binding.

4.7.2 Steered Molecular Dynamics

We used Steered Molecular Dynamics to apply force to the PD-1 ligand in order to separate it from its complex with nivolumab (PDB code 5GGR). This applies force to the ligand, PD-1, pushing it away along the line connecting the centres of mass while holding the nivolumab in place. Figure 4.29 shows the progression of the PD-1 across the frames of the steered MD simulation; the distance between the two components' centres of mass are plotted in red using the right y-axis and the force applied is plotted in blue using the left y-axis. A frame is added to the output trajectory every picosecond.

4.7.3 PaCS-MD

We used PaCS-MD[67] to generate as realistic a docking visualisation as possible from a position like our near-approach pose to the bound structure. We started with a short simulation of the starting structure and used the minimum distance between atoms of the nivolumab and atoms of the PD-1 protein to control the MD. First, snapshots were chosen to increased this minimum distance until it reached 4Å, and then the snapshots that decreased this minimum distance were chosen, to guide the components to form the complex and then guide them apart. The PaCS-MD was performed with 10 parallel iterations over 50 cycles.

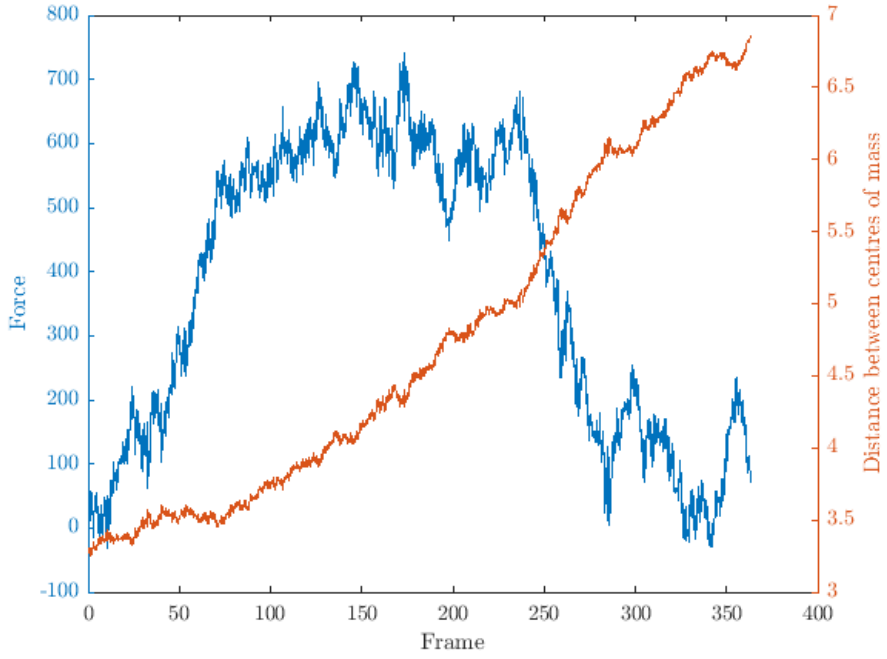


Fig. 4.29 Frames of a steered MD simulation, plotted against distance between centres of mass of nivolumab and PD-1 (right, red) and force applied (left, blue).

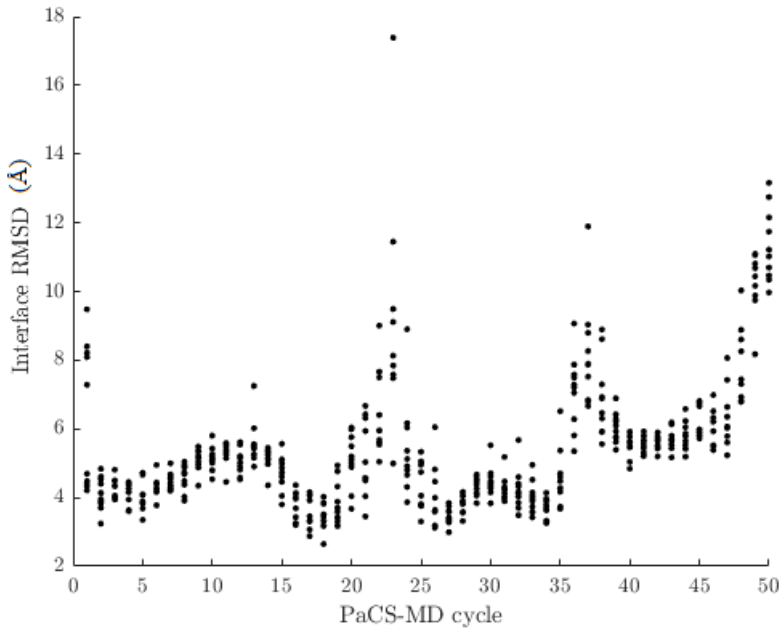


Fig. 4.30 Interface RMSD over time for all snapshots from each cycle of PaCS-MD.

The nivolumab was fully separated from the PD-1 during the first, 23rd, 37th and 50th cycles, so the full PaCS process simulated three full bindings and separations. The interface RMSD (iRMSD) [96] was calculated for each snapshot from each cycle by searching the solved bound structure for atoms within 10Å of atoms in another chain. Cα atoms from this interface section were then aligned to the corresponding atoms in the PaCS-MD snapshots, and the iRMSD for each structure was the RMSD between these Cα atoms. Figure 4.30 shows a scatter plot of these iRMSDs.

We also investigated the contacts between atoms in our simulated structures to identify whether the same contacts were made in our predicted frames as in the native structure, using the fractions of native and non-native contacts [96]. Figure 4.31(a) shows the fraction of native contacts achieved by each snapshot taken from each cycle. The fraction of native contacts is found by counting the number of residue pairs that are in contact in both the PaCS snapshot structure and the native structure (PDB code 5GGR), divided by the number of native contacts. Figure 4.31(b) shows the fraction of non-native contacts, which is found by counting contacts in the snapshot structure that do not exist in the native structure, divided by the amount of contacts in the snapshot. Contacts in both calculations were defined as residues within 5Å of one another.

The closest that our results came to the native structure occurred during the first iteration of the 18th cycle, though the PaCS-MD attempted to further minimise the inter-atomic distances until it reached its minimum during the 22nd cycle. This minimum point has greater than 0.5 fraction of native contacts, an iRMSD greater than one, and a ligand RMSD of 2.790, which puts it in the "medium" category of CAPRI's docking judging criteria [96]. We produced docking morphs from the 1st to the 18th cycle using our MDS-based method (without constructing the near-approach pose) and a simple linear interpolation of Cartesian points.

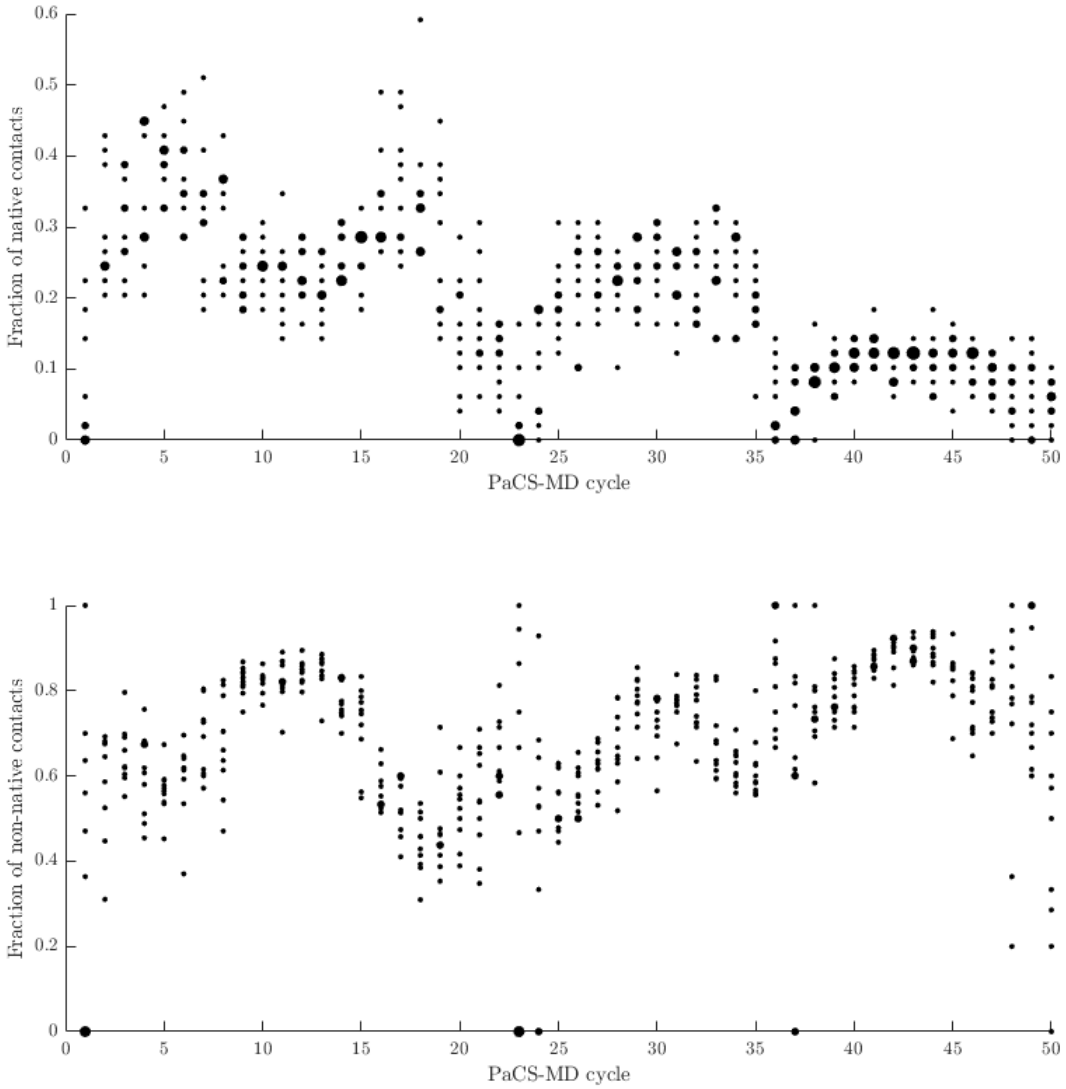


Fig. 4.31 Fraction of native contacts (top) and fraction of non-native contacts (bottom) for snapshots from each cycle and iteration of PaCS-MD. Size of point indicates number of iterations in the cycle with the same fraction.

Figure 4.32 shows the iRMSD produced by the morphing methods. Figure 4.32(a) shows the morphs created from the first snapshot to the start of the 18th cycle, and 4.32(b) shows the morphs that run to the start of the 22nd cycle which is the first point at which the PaCS-MD changed direction.

We calculated the RMSD of the loop sections that are involved in the binding of PD-1 and PD-L1, using the native bound structure 5ggr and the separated and relaxed unbound structure after MD. The FG loop, which spans residues 127 to 134, seems to have a more pronounced correlation with the positions in the MD as plotted in Figure 4.35. The BC loop (residues 57-63) is plotted in Figure 4.34.

4.8 Conclusion and Discussion

We have developed a new protein morphing technique using classical and weighted multi-grid SMACOF MDS to create structures that optimally satisfy a matrix of interpolated inter-atomic distances. The method has been applied to visualisations of the conformational changes of single proteins and other biomolecules. In addition, we have tailored our method to create a novel tool that creates morphs showing biomolecules taking part in docking interactions.

4.8.1 Validity of constructed intermediate structures

Our aims for this method was that the animations resulting from our output frames should be smooth and easy to follow, and that each frame should depict a physically plausible structure. The MolProbity validation tool was used to determine the quality and validity of our structures, providing a metric for clashing atoms and unrealistic bond lengths and angles.We found that using classical MDS on our interpolated distance matrix yielded intermediate structures that were not on average better than a simple linear interpolation of

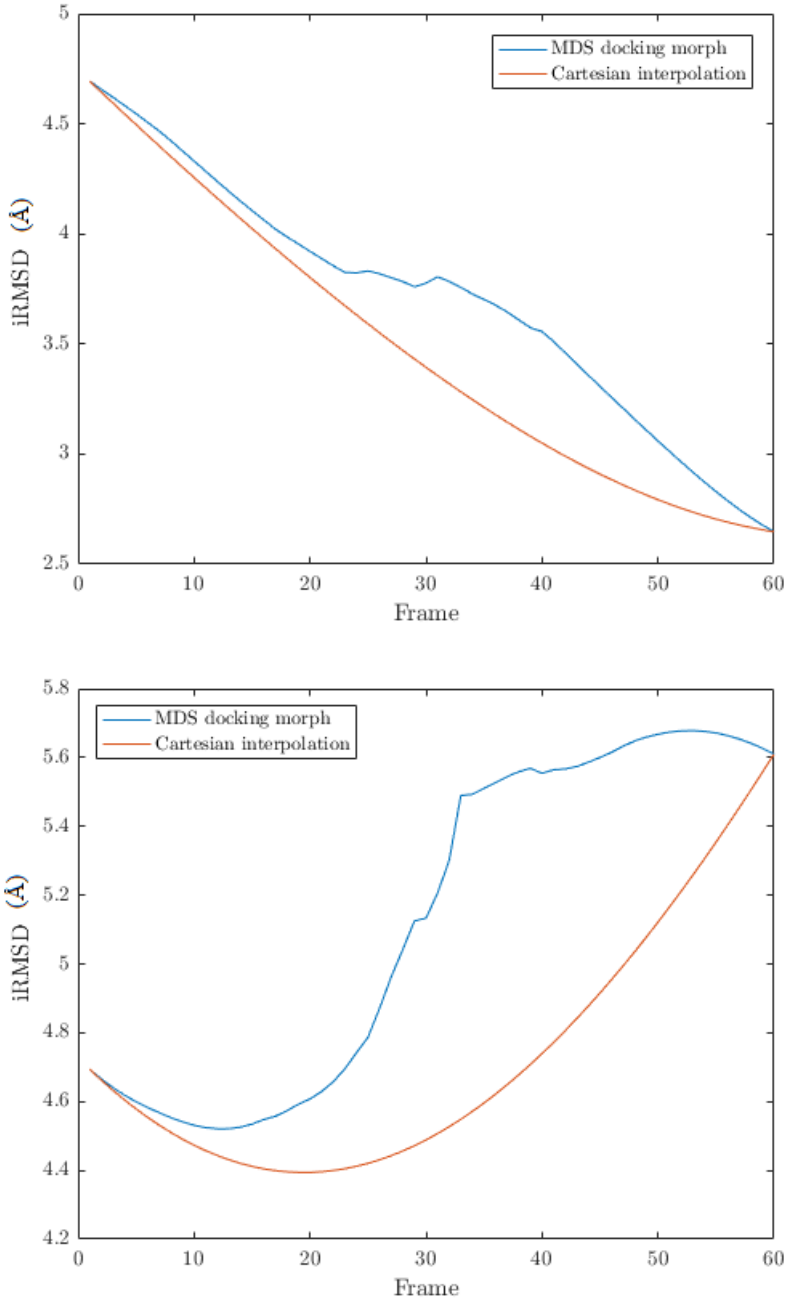


Fig. 4.32 Interface RMSD over time for each frame of the MDS multi-grid docking morph and the linear Cartesian interpolation between snapshots from the first and 18th cycles (top) and the first and 22nd cycles (bottom).

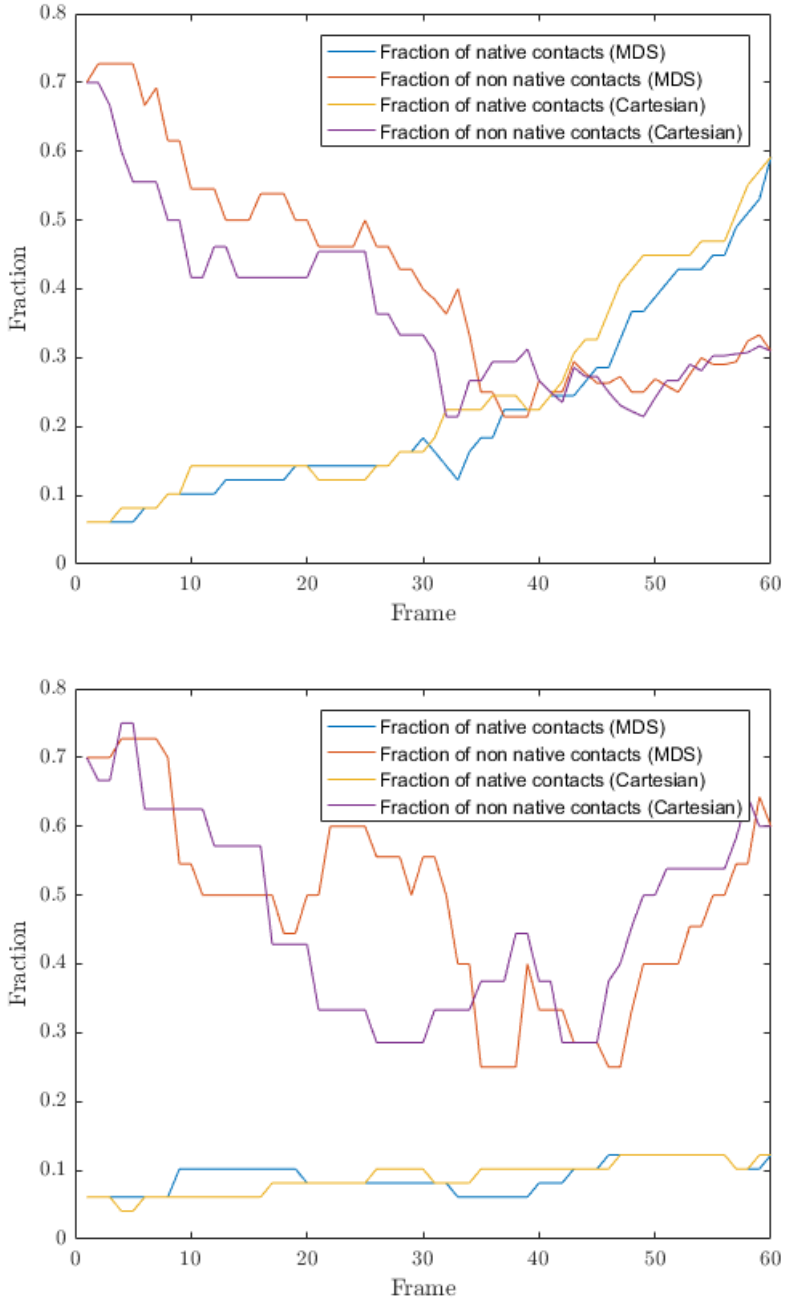
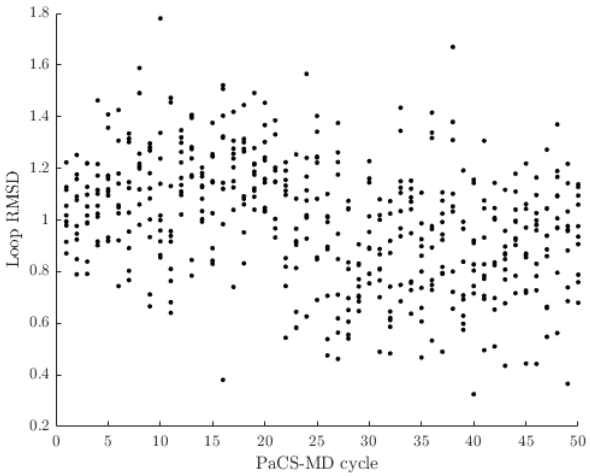
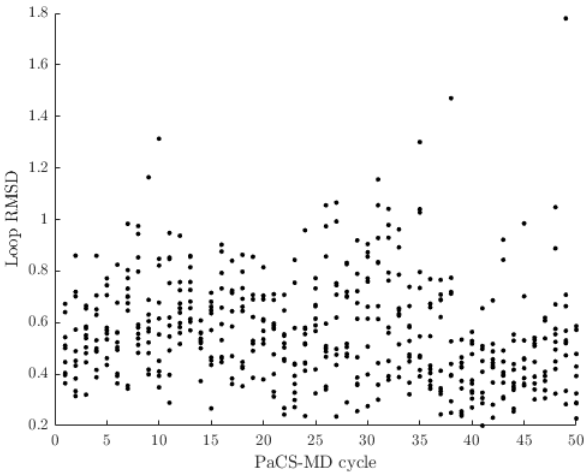


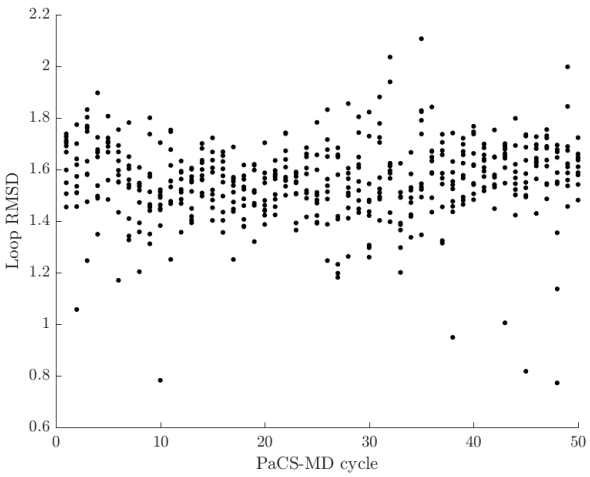
Fig. 4.33 Fraction of native and non-native contacts over time for each frame of the MDS multi-grid docking morph and the linear Cartesian interpolation between snapshots from the first and 18th cycles (top) and the first and 22nd cycles (bottom).



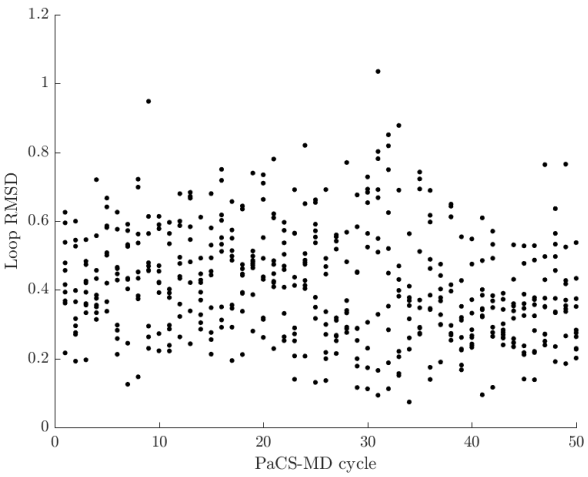
(a) PDB code 5GGR



(b) PDB code 5GGR after separation and relaxation

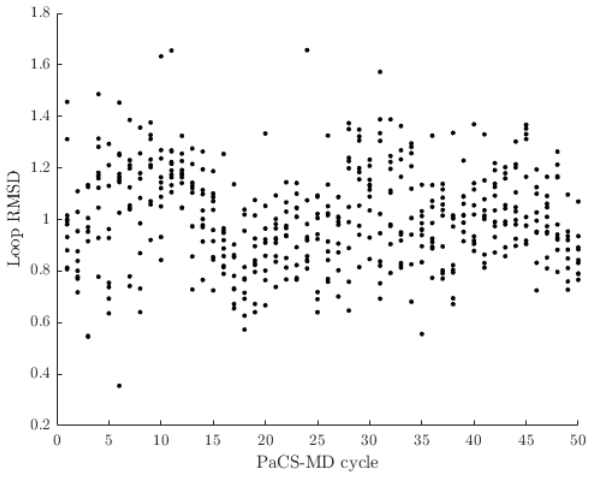


(c) PDB code 2M2D

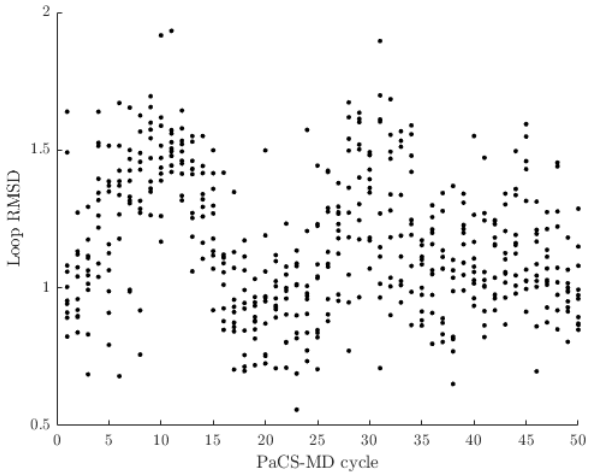


(d) PDB code 5RRQ

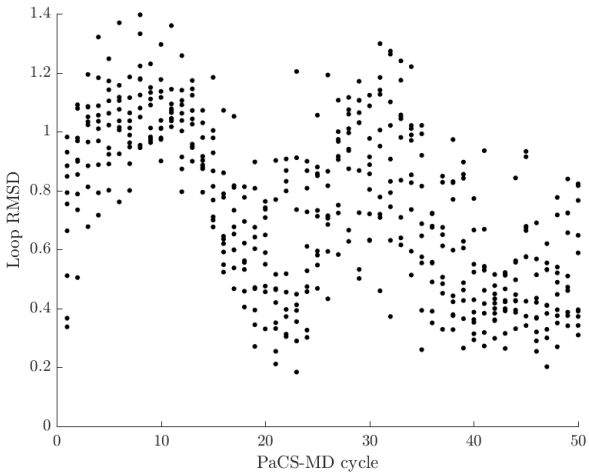
Fig. 4.34 RMSD of the PD-1 BC loop (57-63) in the PaCS-MD simulation, compared to the solved complex 5GGR, the unbound relaxed 5GGR produced by our earlier MD simulations, and the solved structures 2M2D and 3RRQ.



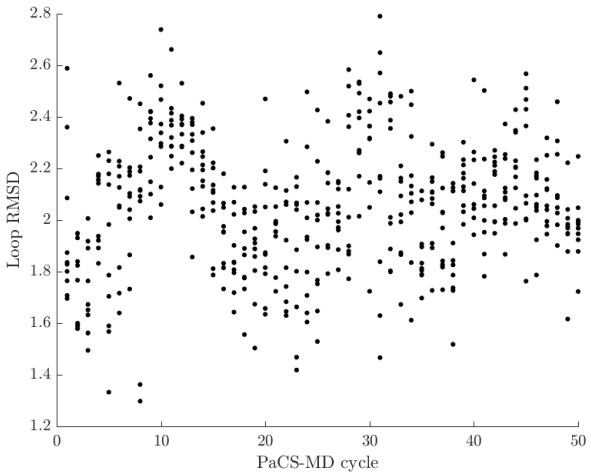
(a) PDB code 5GGR



(b) PDB code 5GGR after separation and relaxation



(c) PDB code 2M2D



(d) PDB code 3RRQ

Fig. 4.35 RMSD of the PD1 FG loop (127-134) in the PaCS-MD simulation, compared to the solved complex complex 5GGR, unbound relaxed 5GGR, 2M2D and 3RRQ.

Cartesian coordinates. However, by using modern MDS we were able to apply weighting to the inter-atomic distances, selecting only atom pairs that were close enough to be bonded. This had a significant positive impact on the appearance of the morph animations and on the validity according to the MolProbity score.

4.8.2 RMSD from Intermediate Structures

We applied our method to the performance metric and test cases proposed by Weiss and Levitt[168], in which a morph is performed from a start structure to an end structure, and then compared at each resulting frame to a solved intermediate that is known to lie along the trajectory of the conformational change.

The improvement scores shown in Figure 4.23 do not show any one morphing technique to consistently outperform the others, though our MDS method performs well on all but one of the morphs. The ranking table in Table 4.1 shows that when compared to each other method, the MDS_CA12 version of our technique outperforms the other method on four of the five cases. For each method, MDS_4 outperforms on more cases than it is outperformed, apart from against MDS_CA12. The high performance of MDA_CA12 compared to the all-atom version is initially surprising as it could be assumed that using all atoms in a morph would result in a more accurate morph.

A large (96° rotation) conformational change in the 5'-nucleotidase protein was morphed. Our method achieved the same improvement score as the FATCAT([173]) morph method, which was the highest scoring technique measured for this morph by [167].

The poorest result for both of our methods was obtained from the skeletal muscle Ca^{2+}-ATPase protein, which produced a 0% improvement score as our morph was as closest to the intermediate structure at the end as in the middle of the morph. While none of the tested methods achieved an improvement score higher than 11.6%, MDS was particularly unsuccessful. The protein used was undergoing a large conformational change in which

three domains are translated and rotated[155]. Our method would assume that these motions happen synchronously, but as the domains are well-separated it is possible that they do not which would explain why the intermediate structure was not reached.

RNase III also provided a difficult test case for the morphing methods, as none achieved an improvement score above 50%, which was achieved by Nomad-Ref and MDS_CA12. The Myosin morph result demonstrates our strongest result wherein the MDS method achieved an improvement score of 64%, higher than the NOMAD-Ref([99]) method's previous highest score of 58%. The ribose-binding protein morphed produced similar results across the morphing techniques, with improvement scores ranging from 47% to 68%. Our MDS improvement score was 63%, which was lower than that of FATCAT but higher than all others.

4.8.3 Docking morphs

Our docking morphs are a feature not included in other protein morphing web servers, so our only comparison is to a naive linear interpolation of Cartesian coordinates. However, the performance improvement that the multi-grid MDS method showed over the Cartesian interpolation is large and significant. We hope that this tool will be useful in the study of observed and predicted interactions between biomolecules, as well as providing illustrative animations for use in presentations and demonstrations.

4.8.4 Molecular Dynamics

We used Molecular Dynamics to investigate the interaction between the drug nivolumab and its target, PD-1.

MD from the near-approach pose has the molecules come together too quickly to observe much, which could indicate that the 4Å near-approach pose is too close for the docking morph.

When the components first come together the native contacts and non-native contacts appear roughly negatively correlated, as the PD-1 moves first towards its native position between the heavy and light chains of the nivolumab, and then away from it and towards a single chain. Later retreats and re-approaches are roughly correlated.

The PD-1 molecule's motion into its near-native position was simple, which makes it difficult to compare to our morphing method. While the MDS morph emphasises the changes in the two PD-1 loops, the Cartesian linear interpolation which suggests that the docking path simulated by the MD was too subtle to provide a good test case for a comparison of docking visualisations. However, studying the produced MD trajectory along with the Cartesian and MDS morphs underlined the usefulness of docking morphs as a tool for creating a short, clear indication of movements during an interaction between two proteins.

4.8.5 Limitations and future work

While the MDS approach achieves good results in structural validation and in approaching solved intermediate structures, as a mostly geometric approach the results are not guaranteed to show the exact trajectory between the initial and end structures. We make an assumption that all changes within the biomolecule are happening steadily and simultaneously, which results in smooth and direct motions but is not necessarily realistic. We made a preliminary investigation into optimising the interpolation function for atom pairs but this was very computationally expensive. A potential future direction for this work would be to cluster the atoms into groups that move together between the conformations and optimise the interpolation functions for distances between and inside these groups, which would require far fewer iterations.

While the most commonly optimal cut-off distance was 4Å, this was not unanimous. It could be interesting to study the relationship between properties of a structure and the cut-off distance that provides its optimal result. Additionally, some initial explorations of

alternatives to the linearly incremented weights of the multi-grid layers were performed, but a full investigation and optimisation of these values would have taken an impractically long time.

Chapter 5

Improvements to the Identification of Dynamic Domains using 6-Dimensional k-means Clustering

5.1 Introduction

Chapter 4 described the development of a method of visualising a conformational change which treats the biomolecules involved as a geometric structure of points, with a motion that is calculated residue-by-residue (or nucleotide-by-nucleotide) and atom-by-atom. This method was compared to MD simulation where the indivisible units are atoms. We will now look at biomolecules using the concept of domains. Section 2.3.2 discussed a number of definitions of protein "domains", including the dynamic domain: a rigid or semi-rigid globular region connected to a separate dynamic domain by flexible hinge-bending regions. It is this definition that will be the focus of the following two chapters.

Identification of these domains, and of their motions relative to one another, yields a quantifiable and analysable description of the motion undergone during the conformational change. A single chain can contain multiple domains, as in the calmodulin structure that was

examined in detail in Chapter 4; the large globular areas at each end of the calmodulin structure remain semi-rigid as they move relative to one another, whereas the thin middle segment is flexible and so can enable the large, twisting closure motion. In large macromolecules, a domain can consist of many chains, which can combine to form complex motions acting as molecular machines. ATP is synthesised from ADP in a process that includes the F_0F_1-ATP synthase protein [19]. The process features a mechanism that involves domain movements in the β subunits in F_1 being driven through rotation of the central γ-subunit which acts as a rotor shaft connecting the F_0 motor to F_1. The ribosome, recently examined in high resolution using new cryo-electron microscopy techniques [102], exhibits a ratchet-like motion [54] as strands of messenger RNA and transfer RNA are translocated between its two subunits.

A number of methods exist to determine the domains within a protein. Several examples of these are discussed in Section 2.3.2; here we will examine the DynDom family of programs. The DynDom program slides a window over the backbone atoms of a given protein structure, then calculates the rotation vectors required to bring each overlapping window into alignment with the corresponding atoms in a second structure. A stand-alone program, DynDom3D, developed this method further, selecting atoms by sliding a 3-dimensional block over a grid rather than focussing solely on the main chain.

This chapter will focus on the implementation of improvements that take the DynDom method to its conclusion,identifying dynamic domains using all 6 dimensions required to define a rigid body motion, and to make the program usable on the very large macromolecules solved using cryo-electron microscopy. The new software tool, DynDom6D, produces domain classifications for structures containing protein, DNA and RNA chains. Where results can be compared with the original DynDom program or with DynDom3D, they are in reasonably good agreement. The implementation was successful without the costly connectivity step that caused size limitations in DynDom3D, meaning that the new software can be used on very large structures that would have previously been impossible to analyse.

The software was tested on macromolecules with known interesting machine-like motions such as the motor-like F_0F_1-ATP synthase and the ribosome, and the domains performing these motions were identified.

Results from this work were published in *Biophysics and Physicobiology* [161].

5.2 Previous DynDom Methods

5.2.1 DynDom

The DynDom [95][70][73][128] program accepts as its input a pair of protein structure files which contain the same chain in two different conformations. It assigns residues in the chain to dynamic domains by finding sections that move semi-rigidly. The program moves a sliding window over the backbone of a protein chain, building a list of overlapping segments. The whole chains from both conformations are superposed, and then each segment is examined to calculate the rotation vector required to bring it into alignment with the corresponding segment of the other conformation. The rotation vectors are treated as points in three-dimensional space and a *k*-means clustering algorithm is called on these points, under the assumption that as domains are semi-rigid, segments of a protein are likely to undergo similar rotations to other parts of the same domain, and less likely to undergo similar rotations to segments from another domain.

Because the rotation vectors are calculated from movements of the main chain, it takes as input a pair of structures for a single chain; when a macromolecule is entered to the program, the user must select one of the chains and all others are ignored. Additionally, any potentially informative side-chain motions are ignored as only the backbone atoms' positions are used in the fitting procedure. These limitations were addressed by the next iteration of the DynDom method, DynDom3D.

5.2.2 DynDom3D

DynDom3D [126] extends this concept to macromolecules by taking an all-atom, spatial approach. As in the original DynDom program, rotation vectors are calculated that describe a movement between the start and end positions for subsets of atoms, and these vectors are clustered to give groups of similar rotation vectors that are considered potential domains. Atoms are assigned to the cluster to which the majority of subsets it belongs to were assigned. However, the sliding window from the original DynDom algorithm is replaced by a block; the overlapping subsets of atoms are created by taking the atoms that fall within the block at each of its positions. The following section will describe this algorithm in more detail.

The two structures are input and aligned into one-to-one correspondence, and the coordinate system is changed to that of the principal axes of the first structure. A 3-dimensional cubic grid is constructed around the protein, with points separated in each dimension by a grid length parameter g. A block is slid through the grid such that at each point it contains everything within a scalar parameter b^3 grid points, or b grid points in each dimension, where b is an integer referred to as the "block factor".

For each block the corresponding atoms in the second structure are identified and a least-squares best fit method calculates the rotation vector required to bring them into alignment.

DynDom3D then performs k-means clustering on the blocks based on the rotation vectors, resulting in groups with similar rotations. Provided these groups meet a ratio criteria wherein their relative movement with respect to another domain is greater than their internal change, they are treated as valid domain pairs. Results for k are only stored if every pair of potential domains is a valid domain pair.

In k-means clustering, a valid value of k must be chosen. DynDom3D begins with a single cluster and repeats for increasing values of k, until a value of k is found which gives results where a minimum domain size threshold has not been met.

The atoms in a group may be spatially separated, so a computationally expensive connectivity algorithm selects only atoms which form a connected domain.

5.3 DynDom6D

DynDom6D is a new generation of the DynDom method in which the *k*-means clustering is performed on all 6 dimensions describing a rigid body movement as a rotation about a screw axis according to Chasles' theorem. DynDom3D performed clustering on the rotation vector alone, so regions of the protein that had a similar rotation about distinct axes would be clustered together. A costly post-processing step was implemented to fix this by testing for connectivity between the atoms assigned to a domain using a memory intensive and time consuming subroutine. DynDom6D includes additional information about each blocks' movements to prevent the initial incorrect clustering so that the connectivity testing is no longer required. This dramatically reduces the computational complexity of the procedure, allowing the technique to be performed on larger macromolecules which would previously have either taken a very long time or require more memory than was available. Additionally, because the atoms no longer need to be fully connected, the input structures may now include missing or incomplete sections. The new implementation also allows for DNA or RNA structures as input.

The method contains several subroutines and processes. Figure 5.1 shows an overview of the key steps of the DynDom6D process, which will be explained in the following section.

5.3.1 Chasles' theorem

The principle behind the 6 dimensions of DynDom6D is Chasles' theorem [28], which states that any rigid body motion between two positions and orientations can be described in terms

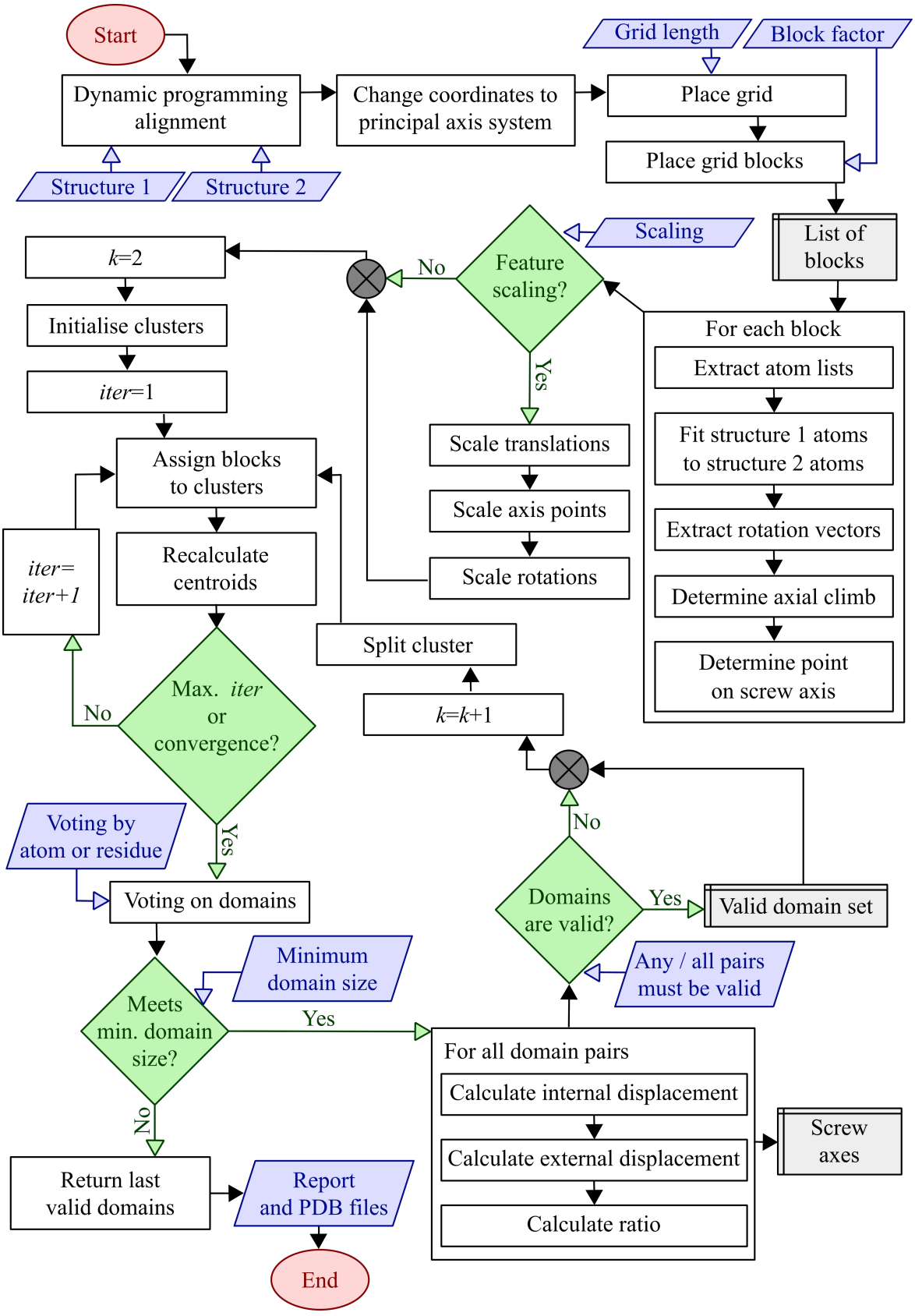


Fig. 5.1 Flowchart of the key stages of the DynDom6D process.

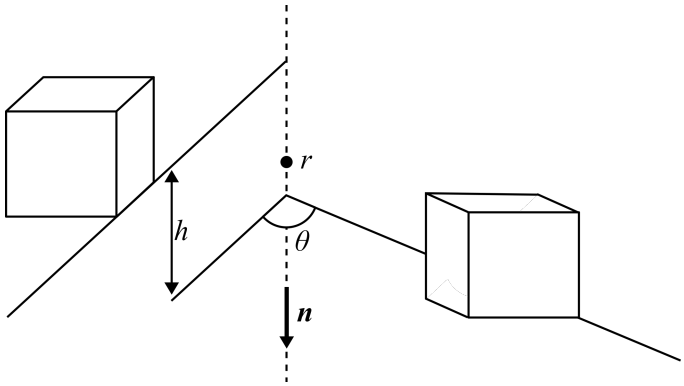


Fig. 5.2 Diagram of a rigid body motion about a screw axis (represented by a dashed line), where r is a point on the screw axis, $\boldsymbol{n}$ is the axis' unit vector, h is the translation along the axis and θ is the rotation about it.

of the location and direction of a unique screw axis, and the degree by which the body is rotated about this axis, and the translational movement that the body undergoes along it.

Figure 5.2 shows a diagram of the movement of a rigid body, in this example a cube. Here, $\boldsymbol{n}$ is the unit vector in the direction of the axis, $\boldsymbol{r}$ is a point on the axis given as a position vector from the origin, θ is the rotation about the axis, and h is the translation along the axis.

5.3.2 Parameters

There are user-defined parameters which will be referred to in the explanation of the algorithm, which control the specificity and tolerance of the program. The function and default value of each parameter have been listed for reference in Table 5.1.

5.3.3 Pre-processing

The program takes as its input a pair of structural files containing chains of protein, DNA and/or RNA structures, representing points at either end of a biomolecule's conformational change. These can be files in either PDB or mmCIF format. These structures are brought into alignment using a dynamic programming-based sequence alignment, which was originally

Parameter	**Default Value**	**Description**
Grid Size (Å)	4.0	Distance between grid points in each dimensions
Block Factor	2	Number of grid points per block in each dimension
Occupancy	0.4	The maximum number of atoms in a single block is found and multiplied by this factor. Blocks with occupancy below this product are discarded
Minimum Domain Size	200	Minimum number of atoms in a valid domain
Hinge-Bending Threshold	0.51	Threshold for confidence in domain assignment, below which an atom or residue is assigned as a hinge-bending region
Assign domains by residue	True	Perform domain assignment for each residue (if true) or each atom (if false)
Perform feature scaling	True	Perform feature scaling before clustering (if true)
Save clustering results if ALL domain pairs meet ratio	True	Results will only be returned if every pair of domains meet a given ratio of external to internal movement (if true), or results are returned if at least one domain pair meets this ratio (if false)

Table 5.1 The purpose and default value of each DynDom6D parameter requested from the user.

implemented by Girdlestone and Hayward[60] for the DynDom3D software, adjusted to allow the alignment of sequences of nucleotides as well as amino acids.

The alignment operates first on the chain level, removing any chains that do not have a corresponding chain in the other model. Each of these chain pairs are then aligned using dynamic programming (see Section 3.1), performing a global alignment with the BLOSUM62 substitution matrix for chains of amino acids and the NUC44 substitutional matrix for chains of nucleotides. Finally, the atoms in each amino acid or nucleotide are brought into alignment using dynamic programming on the atoms' names, using the identity matrix for the substitution matrix.

After alignment, the coordinate system is changed to the principal axes system. A least-squares best-fit routine superposes the second model onto the first to bring the structures into an initial all-atom alignment.

5.3.4 Grid points and blocks

A 3-dimensional grid is constructed over the co-ordinates of the first structure by creating a 3-dimensional array of "cell" objects. Each cell represents a $g \times g \times g$ cube where g is a cell length parameter (see Table 5.1), and contains a list of the atoms that it encloses. The "sliding block" is then simulated by creating a "block" object for each grid point, where b cells in each dimension are considered to be inside the block. This integer block factor, b, is another user-defined parameter with a default value of 2; for any b above 1, atoms will each belong to multiple blocks, which has a smoothing effect as each atom belongs to different subsets of atoms.

Figure 5.3 shows a diagram visualising this process, in which grid points are constructed that separate the space into cells which contain atoms, and blocks overlap to create subsets of cells. The block highlighted in red will contain all atoms in the blue cell, as well as the

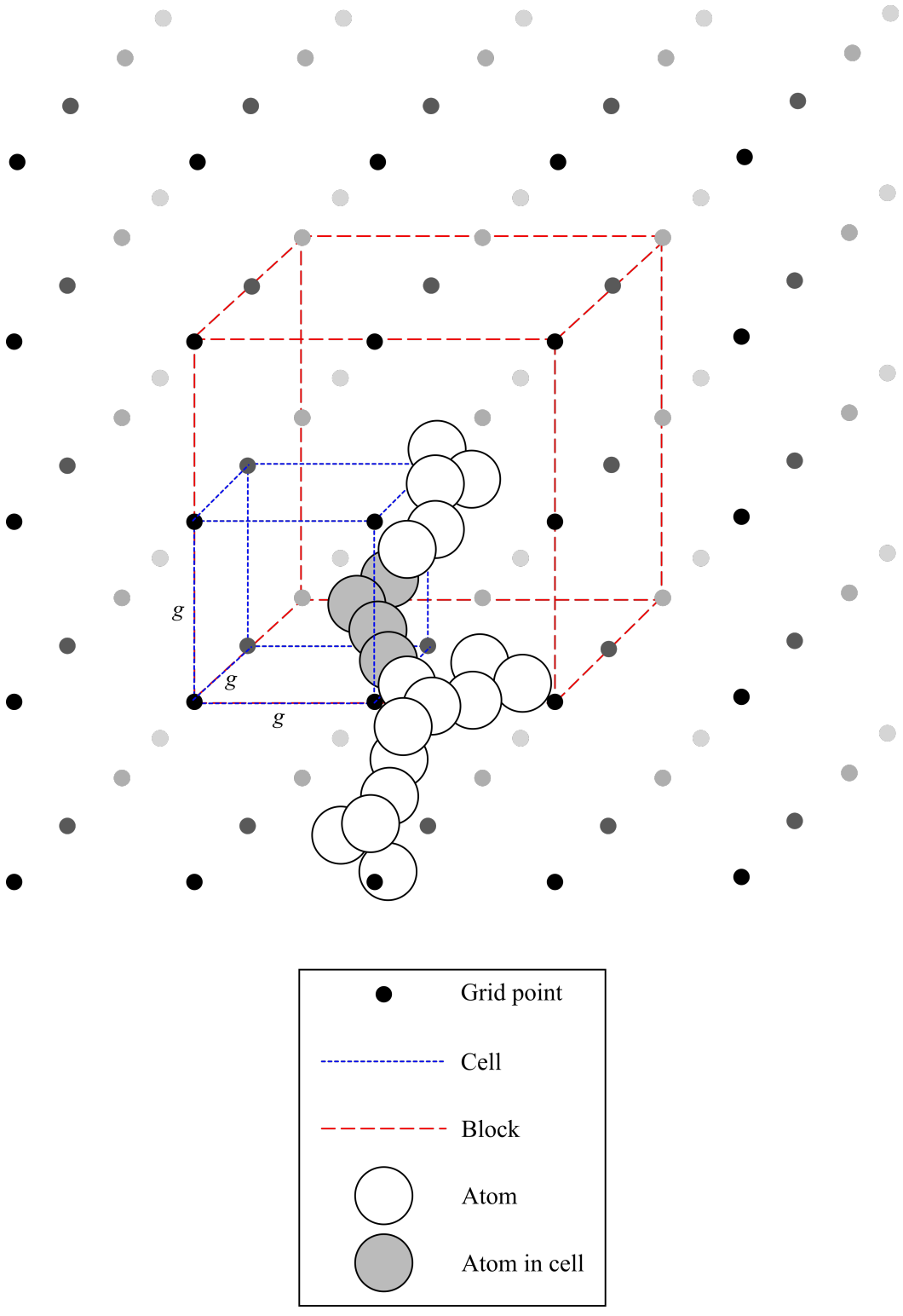


Fig. 5.3 The assignment of atoms to grid cells and blocks. Each grid point is separated in each direction by the grid length parameter g. The block factor is two; each edge of the block contains two cell edges.

other 7 cells. Blocks are constructed at each grid point, so the blue cell and its atoms will be included in several blocks.

Before clustering, we discard blocks containing very few atoms, as they are likely to contain noisy surface motions rather than acting as part of a dynamic domain. The number of atoms N in each block is counted, and all blocks where the number of contained atoms is less than the product of N_{max} and the occupancy parameter (see Table 5.1) are not passed to the clustering subroutine.

5.3.5 Features for clustering

A quaternion-based fitting method begins the process by superposing the block's first and second model atoms; as it is a quaternion method, the rotation unit vector $\boldsymbol{n}$ and angle of rotation θ can be extracted from the subroutine's results directly.

The following process now distinguishes it from DynDom3D, which clusters only on $\theta\boldsymbol{n}$. The axial climb, h, is determined by the dot product of $\boldsymbol{n}$ and the vector of external displacement calculated during fitting. The calculated values of h, $\boldsymbol{n}$ and θ are then used to identify the location of a point on the screw axis. The rotational components are calculated by subtracting $h\boldsymbol{n}$ from the vector of external displacement, obtaining the vector of the rotational component as the square root of the sum of squares of the rotational components. Dividing each rotation component by this magnitude gives the unit vector in the direction of the rotational components, from which the perpendicular vector can be calculated to yield a vector in the direction of the axis from the atoms in the block. By adding this vector to an original position vector, we then have a point on the screw axis, $\boldsymbol{r}$.

The program then has all of the information it needs to pass features fully describing the motion to the k-means clustering subroutine. The first 3 dimensions of clustering are provided by the three dimensions of the rotation vector $\theta\boldsymbol{n}$. The fourth dimension is the axial climb, h. The remaining dimensions are provided by the screw axis location and represented

in the clustering features by $\boldsymbol{r}+\boldsymbol{n}\tau$, which is the equation of the straight line that coincides with the screw axis. The point on the axis $\boldsymbol{r}$ adds a further three dimensions bringing the total to seven, which one more than strictly needed to describe a rigid-body movement. However, we deal with this issue in a variant of the k-means clustering as described below. The screw axis could be fixed in place by giving the coordinates at which it crosses a 2-dimensional plane, however this may result in points that are more spread out than they would be if clustered in 3-dimensional space.

5.3.6 Feature Scaling

The blocks are clustered on an orthogonal combination of the $\boldsymbol{r}+\boldsymbol{n}\tau$, $\theta\boldsymbol{n}$ and h features, but the scale of the blocks' features often differ. This could affect the clustering, as differences between features with a greater scale would then contribute more to the distance between a point and a cluster's mean than a feature where blocks' values fall within a smaller range. An optional scaling of the features was implemented in order to prevent the larger features from outweighing the smaller features and, when requested by the user, is performed for each feature space as follows.

Each translation along the axis is scaled by the mean of h values from all blocks:

$$h_i' = \frac{h_i}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} h_i^2}} \tag{5.1}$$

Points on the screw axis are scaled by the magnitude of the points' position vectors:

$$\boldsymbol{r}_i' = \frac{\boldsymbol{r}_i}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} |\boldsymbol{r}_i|^2}} \tag{5.2}$$

The rotational parts are scaled by the mean angle of rotation, θ:

$$\theta_i' \boldsymbol{n}_i = \frac{\theta_i \boldsymbol{n}_i}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} \theta_i^2}} \tag{5.3}$$

This ensures that features within the same space, such as the x, y and z co-ordinates of the arbitrary point on the screw axis, are scaled by the same amount.

5.3.7 6-Dimensional Clustering

In Section 5.3.5 the features for clustering were listed: $\theta\boldsymbol{n}$, h, and $\boldsymbol{r}+\boldsymbol{n}\tau$. Each block is now represented as a point in our 6-dimensional feature space, and to find domains containing atoms that make very similar movements, we will look for clusters where points are close together in the feature space. Each cluster will identify a set of atoms that are moving in a very similar manner, which will indicate that they belong to a semi-rigid domain.

The k-means clustering algorithm, as described in Section 3.7.1, requires a predetermined value of k, but the number of domains is not known until after clustering is completed. We take an iterative approach, starting at $k = 1$ and incrementing k after each cycle of clustering until either a stopping criterion is reached or k reaches the maximum number of clusters, which we set high at 100. After each cycle, if the clustering is to be continued then the cluster with the highest variance is split at its mean, and the next cycle begins with $k+1$ clusters.

5.3.7.1 Clustering of Lines

To assign a point in our feature space to a cluster, we find the distance between the point and the mean of each cluster, and add the point to the cluster with the closest mean. The mean of h and each dimension of the $\theta\boldsymbol{n}$ values for a given cluster can be calculated using the normal method for calculating the mean of a set of numbers. However, the mean of the screw axes $\boldsymbol{r}+\boldsymbol{n}\tau$ requires a new process.

The mean of a set of values is the point at which the distance to each value is minimised. If cluster l contains blocks S_l, this mean point would then be the point with position vector $\boldsymbol{R}_l$ that minimises the distances to the screw axes of every block j in S_l:

$$D_l = \sum_{j \in S_l} |(\boldsymbol{r}_j - \boldsymbol{R}_l) \times \boldsymbol{n}_j|^2. \tag{5.4}$$

Where "$\times$" is the cross product. For each line j we construct from the unit vector a matrix:

$$\boldsymbol{\eta_j} = \begin{bmatrix} n_{yj}^2 + n_{zj}^2 & -n_{xj}n_{yj} & -n_{xj}n_{zj} \\ -n_{xj}n_{yj} & n_{xj}^2 + n_{zj}^2 & -n_{yj}n_{zj} \\ -n_{xj}n_{zj} & -n_{yj}n_{zj} & n_{xj}^2 + n_{yj}^2 \end{bmatrix}. \tag{5.5}$$

These matrices are summed together to form the 3×3 matrix $\boldsymbol{N}$. We define $\boldsymbol{P}_l = \sum_{j \in S_l} \boldsymbol{\eta}_j \boldsymbol{r}_j$, and $Q_l = \sum_{j \in S_l} \boldsymbol{r}_j^t \boldsymbol{\eta}_j \boldsymbol{r}_j$ for each block.

Equation 5.4 can be rewritten as:

$$D_l = \boldsymbol{R}_l^t \boldsymbol{N}_l \boldsymbol{R}_l - 2\boldsymbol{R}_l^t \boldsymbol{P}_l + Q_l, \tag{5.6}$$

so the position vector to the minimum point R_l will be located where:

$$\bigtriangledown D_l = 2\boldsymbol{N}_l \boldsymbol{R}_l - 2\boldsymbol{P}_l = 0. \tag{5.7}$$

Therefore, for each cluster we can use the calculated $\boldsymbol{N}_l$ matrix and $\boldsymbol{P}_l$ to find our minimum point $\boldsymbol{R}$ by solving:

$$\boldsymbol{R}_l = \boldsymbol{N}_l^{-1} \boldsymbol{P}_l. \tag{5.8}$$

Distances to the mean are then calculated from the sum of Equation 5.4 and the sum of squared differences for the h, $\theta \boldsymbol{n}_x$, $\theta \boldsymbol{n}_y$, and $\theta \boldsymbol{n}_z$ features. All blocks are assigned to the

cluster with the minimum distance to the mean, and the mean for each cluster is recalculated based on its new population. This step is repeated until either the means stop changing or the maximum number of iterations (50) is reached.

5.3.8 Voting

In order to produce domain assignments from the clustered blocks, a "voting" procedure is performed. By default this is performed for each residue, and the rest of this section will proceed under the assumption that this has not changed, but this may be performed for each atom instead by changing the appropriate parameter (see Table 5.1).

For each cluster, every block that was assigned to the cluster is examined in turn. Every residue that contains at least one atom in the block is given one "vote" for the cluster. When every block in every cluster has voted, for each residue the votes are counted, such that v_l is the number of votes that the residue belongs to cluster l. Residues are assigned to whichever cluster gave it the most votes, $v_{l_{max}}$.

5.3.8.1 Hinge Assignment

While the original DynDom program returned the location of hinge-bending regions, DynDom3D only returned information about regions that belonged to domain clusters and did not provide any assignment of potential hinges. The DynDom6D program uses the outcome of the voting to identify potential hinge-bending regions.

We find the ratio of votes each residue receives towards the domain to which it is eventually assigned,

$$\frac{v_{l_{max}}}{\sum_l v_l}, \tag{5.9}$$

and the result is compared to *bend*, the bending threshold parameter in Table 5.1. Where the ratio is less than the threshold, the residue is added to a hinge-bending region.

This identifies regions which do not move entirely rigidly with the domain to which they have been assigned, on the grounds that the hinge bending region is more flexible. However, this also selects residues that happen to make small movements that are not part of the main domain movement, which results in the selection of areas on the outside of protein surfaces that do not actually belong to the hinge-bending region.

This does not identify all hinge-bending regions, so an additional procedure runs over the atoms or residues in sequence order, and wherever it encounters two adjacent residues belonging to different domains, the two residues at the boundary are changed to hinge-bending regions.

DynDom6D is sensitive to potentially distant noise as small disconnected areas that happen to move similarly to a domain are not be removed by a connected set algorithm. If these areas are surrounded by residues belonging to another domain then they are likely to fall below the hinge threshold due to votes for the other cluster from the overlapping blocks. Therefore, in contrast to DynDom, DynDom6D removes hinge-bending regions from the set of residues or atoms within a domain to reduce the effect of noise on the following stages.

5.3.8.2 Stopping criterion

After the clustering is complete and atoms have either been assigned to a domain or labelled as a hinge-bending region, the domains are examined in turn to count the number of atoms each contains. If any domain contains fewer atoms than the limit imposed by the "minimum domain size" parameter (see Table 5.1), the clustering ends. The domains are discarded, and the final set of domains are returned to the main method.

5.3.8.3 Ratio for accepting clusters

If the set of domains passes the minimum domain size criteria, the proposed domains are examined to determine whether they would describe valid domain movements. Every pair of

domains is considered as a "domain-movement pair". A set of co-ordinates for each model is constructed containing only the atoms belonging to the two domains. One domain is selected to be held static and the relative movement of the other is calculated, producing vectors of displacement for each atom i, $\Delta\mathbf{d}_i$. The models are aligned using only the atoms of the static domain, and the motion required for the moving domain to (move) between its two positions are calculated. We compare the displacement within each domain to the relative displacement, with a weighting applied based on the number of residues in the domains, N_A and N_B, using the ratio described in Hayward and Berendsen[70]:

$$\sqrt{\frac{\left(\sum_i^{N_\mathrm{A}}\left|\Delta\mathbf{d}_i^{\mathrm{ext}}\right|^2/N_\mathrm{A}\right)+\left(\sum_i^{N_\mathrm{B}}\left|\Delta\mathbf{d}_i^{\mathrm{ext}}\right|^2/N_\mathrm{B}\right)}{\left(\sum_i^{N_\mathrm{A}}\left|\Delta\mathbf{d}_i^{\mathrm{int}}\right|^2/N_\mathrm{A}\right)+\left(\sum_i^{N_\mathrm{B}}\left|\Delta\mathbf{d}_i^{\mathrm{int}}\right|^2/N_\mathrm{B}\right)}} \tag{5.10}$$

Where the ratio is greater than 1 there is more external displacement than internal displacement, which we consider necessary for a valid domain-movement pair.

By default, DynDom6D requires every pair of domains found by the k-means clustering to be a valid domain pair. If the clustering for a given value of k results in a pair of clusters that fail to meet this ratio then all domains from this k-means step are discarded, and the subroutine continues, setting $k = k+1$. The user may request a more lenient criterion instead, where the results are saved as long as at least one domain pair meets the required ratio.

5.3.9 Post-processing

Output files showing the assigned domains are created and presented to the user, along with a report file listing the properties of each domain pair. The aligned models are (built) in the PDB or mmCIF format, and a PyMol script file is created which loads each structure and colours atoms according to their domain. During the calculation of the ratio of internal to external displacement, the rotation vector and screw axes defining the relative movements of

the each domain pair are obtained. These axes are drawn into the output and coloured using the colouring of the static (shaft) and moving (head) domains.

5.4 Results

5.4.1 Citrate Synthase

The program was run using open and closed citrate synthase structures with PDB codes 1CTS and 2CTS. All parameters were left on their default settings.

As a protein consisting of a single chain that displays a clear domain movement, [130], citrate synthase presents a good test case for comparison with the original DynDom method. The application of DynDom to citrate synthase was the focus of a previous study [70] which identified one large and one small domain, and highlighted the screw axis passing between the N and C terminals of a β-hairpin. This β-hairpin loops from the larger, blue domain but moves with the smaller red domain.

While there are some small differences between the two sets of results, the same features can be seen in the DynDom6D output. The screw axis again passes through the β-hairpin, which emerges from the blue domain, into a hinge-bending region, into the red domain, and then loops back into the larger domain.

5.4.2 Aspartate transcarbamoylase

Aspartate transcarbamoylase (ATC-ase) provides a more challenging test case as a larger enzyme, which consists of 12 chains arranged symmetrically. A pair of larger trimers make up catalytic units, and three smaller dimers each contain two regulatory chains [64]. We performed a DynDom6D run using ATC-ase structures with PDB codes 1AT1 and 1RAA, where the grid length parameter was set to 6Å and the block factor was set to 5.

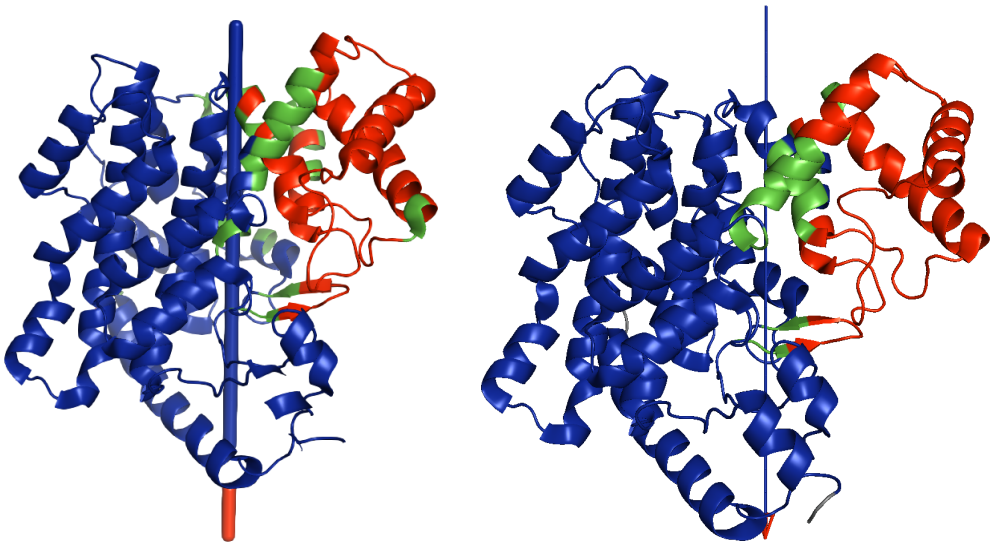


Fig. 5.4 Citrate synthase domains assigned by DynDom6D (left) and DynDom (right).

This symmetry can also be seen in the domain assignments shown in Figure 5.5. Five domains were identified, which each encompass one of the two dimers or three trimers. Each pair of domains met the ratio of external to internal motion required to be classed as a valid domain pair; the two catalytic trimers rotate and translate relative to one another about the axis of symmetry.

5.4.3 Bovine heart mitochondria ATP Synthase

The F_0F_1-ATP synthase protein is an important molecular machine that resembles a motor [114] [136] [19], consisting of an F_0 subunit that acts as a rotor for the F_1 subunit.

We ran DynDom6D on bovine heart mitochondria ATP synthase using structures with PDB codes 5ARA and 5ARH, using the default parameters. The F_0 subunit and rotor stalk comprise the red domain, which rotates about a central axis while the F_1 subunit and stator make up the blue domain which is held in place.

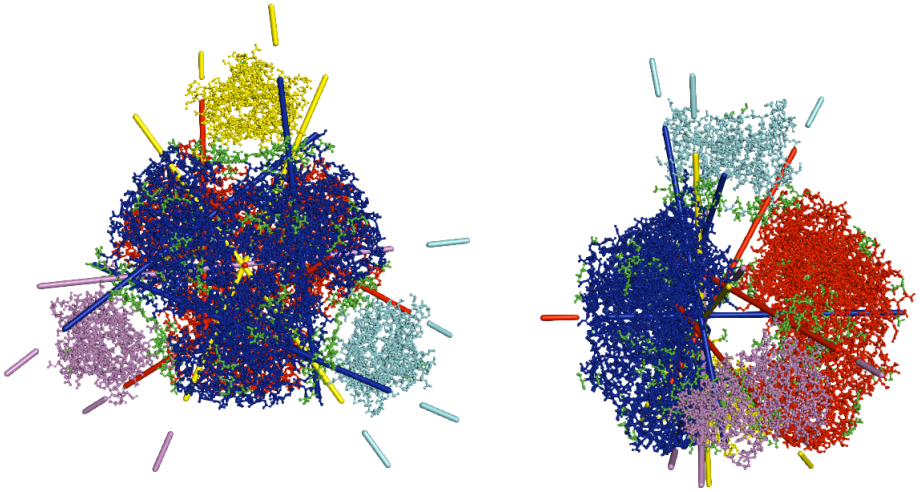


Fig. 5.5 DynDom6D results on aspartate transcarbamoylase.

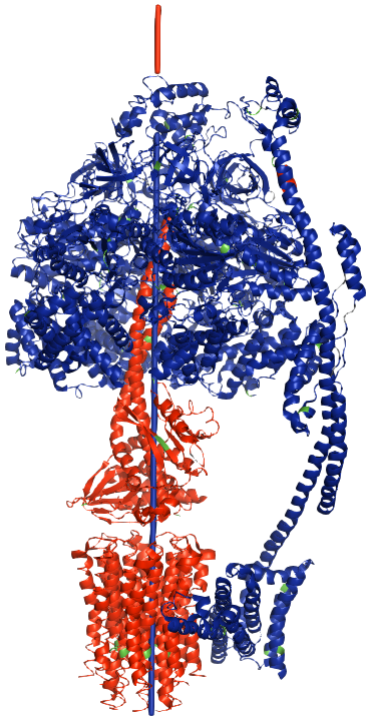


Fig. 5.6 DynDom6D results on bovine heart mitochondria ATP synthase.

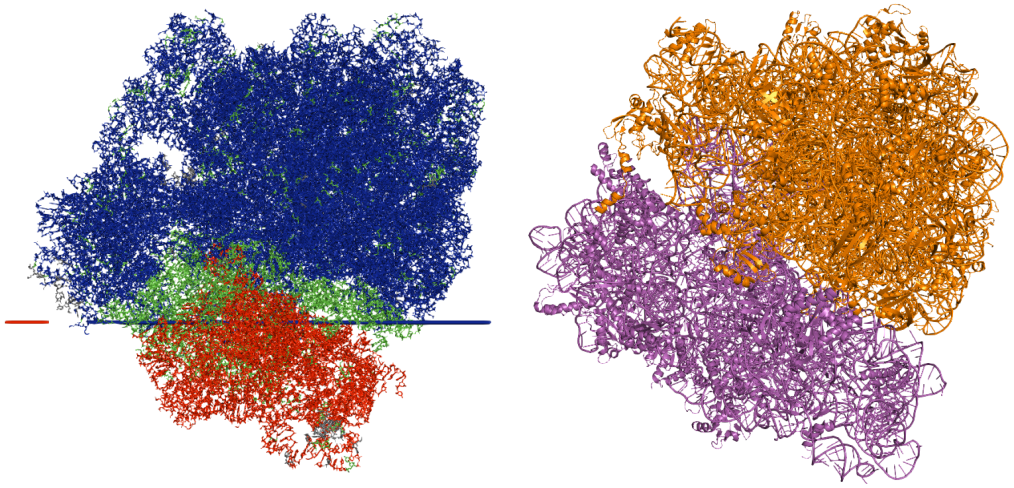


Fig. 5.7 DynDom6D results on the 70S ribosome (left), and a diagram showing the 50s and 30S subunits highlighted in orange and magenta respectively (right).

5.4.4 Ribosome

The *E. Coli* 70S ribosome is a large macromolecule, consisting of a 50S subunit and a smaller 30S subunit, both made up of chains of rRNA and protein[102]. As it carries out its function, the ribosome exhibits a "ratchet-like" motion [54] whereby the 30S subunit rotates relative to the 50S subunit. Due to the scale of the ribosome, and its inclusion of RNA, both DynDom and DynDom3D would be unable to take it as input in their current implementation and server architecture.

Figure 5.7 shows the results of the DynDom6D program run on the *E. Coli* 70S ribosome, using PDB entries with codes 5UYL and 5UYM. Default parameters were used, apart from a grid size of 8Å and a block factor of 4. Two domains were identified: the arrow shows the smaller red domain, which contains residues from the 30S subunit, moving about the larger blue domain, which contains all of the 50S subunit and part of the 30S subunit.

5.5 Discussion

This chapter presented the methods and results of the development of DynDom6D, a new generation of the DynDom domain movement analysis programs. On single-chain proteins, the program produces similar domains to the existing DynDom web server. In addition, the program produces results for input structures that would have been previously unobtainable due to this scale, presence of RNA or DNA chains, or missing or incomplete sections.

The program has produced interesting results on the selected test cases, particularly on aspartate transcarbamoylase which displays strong symmetry of domain boundaries that mirrors the structural symmetry.

However, there are some limitations of the process as it is currently working. Feature scaling is performed to prevent larger Cartesian co-ordinate values from outweighing the much smaller rotation features, resulting in a similar range of values in all three feature spaces. This assumes that each has equal discriminatory value, but if that assumption does not hold then there is a potential for improvement in exploring whether separate weighting of each of the three outputs of feature scaling may achieve better results.

There is further room for improvement through additional optimisation of the parameters. As can be seen in the results on our selected test cases, it is sometimes necessary to alter the parameters in order to achieve a result. Through the process of finding parameters that worked in each case it was observed that rough patterns were emerging: larger proteins performed better with larger grid sizes, and proteins that contain both dense areas and thin isolated chains like F_0F_1-ATP synthase obtain better results with a lower occupancy threshold. However, these are general patterns observed on a small sample of inputs. The lack of labelled data makes the programmatic optimisation of parameters more challenging. The DynDom web server only stores the results of the DynDom program on single protein chains, so optimal parameters found using this data may favour small proteins.

Chapter 6

Investigations into the Location and Composition of Hinge-Bending Regions using Kernel Logistic Regression

6.1 Introduction

Chapter 4 described the development and application of a method of visualising conformational changes undergone by biomolecules, using a largely geometric series of calculations working from a known set of start and end co-ordinates. Chapter 5 detailed a method for analysing conformational changes to identify parts of the structure that move as a semi-rigid body. This method, DynDom6D, operated on pairs of co-ordinates for each atom, taken from structures solved before and after the motion. The starting biomolecular structure is divided into overlapping blocks of atoms, which are clustered on information about the rigid body movement that would superpose them into the positions that they take in the ending structure. Atoms belonging to blocks that shared similar movements are assigned to domains. Atoms that do not strongly belong to a domain are identified as potential hinge-bending regions, which are flexible regions between domains that drive conformational change.

The location of hinge-bending regions are important for understanding the motion undergone by a protein. Chapter 2 (Section 2.3.2) discusses the properties of the hinge bending region in more detail. Chapter 3 (3.8) outlines existing methods that seek to identify the location of hinge-bending regions. Some methods, like FlexProt[145] and the programs in the DynDom family, begin with a pair of solved structures. Others attempt to find the hinge-bending residues using only a single structure, through graph theoretical approaches[82], energetics[50], or Normal Mode Analysis[24]. Some [52][12] have attempted to build methods that find the hinge-bending region from sequence data alone, with mixed success.

The following chapter details the training and testing of a series of predictors that operate on protein sequence data taken from the DynDom database, with hinges labelled using DynDom's hinge assignments. We use Kernel Logistic Regression, as described in Section 3.7.2, which models the likelihood that an input vector belongs to a target class. The models take as input a segment of a sequence of amino acids, and output the likelihood that the central residue of the segment will belong to the hinge-bending region. The weights applied to sequence features by the models will be examined for potential biological insight into the composition of hinge bending regions.

A predictor using the trained models, is available at http://hingeseek.cmp.uea.ac.uk. A paper describing the method and its results was published in BMC Bioinformatics[159].

6.2 Methods

KLR maps an input vector of feature onto a feature space, where it can find a linear separation of non-linear data. By using a kernel function which defines the inner product of feature vectors, this mapping to feature space is implicitly defined. We use three kernels for the following methods as defined in Section 3.7.2 the linear kernel, the polynomial kernel (quadratic and cubic), and the radial basis function (RBF) kernel.

6.2.1 Constructing the Datasets

Data was taken from DynDom's user constructed database [95] and non-redundant databases [128]. These databases contain examples of different classes of motion; some are clearly defined hinge motions, some shear, and some mixed [154]. In order to use examples of clear, defined hinge motions, some filtering was performed as in Table 6.1. Two datasets were created, Group 1 which was smaller and more stringently filtered and the more lenient Group 2. Both datasets permitted only two-domain proteins, and where each domain contained at least 80 residues. The relative angle of rotation between the domains has a minimum value for a pronounced motion about the hinge. A limit on the length of the bending region is imposed to prevent apparent "bending" regions that are actually long areas of disorder. Each group has a restriction on number of bending regions as low quality results commonly contained many scattered bending regions.

To build the training and test data sets from the sequence and bending region data, a sliding window was placed over each sequence, resulting in a subsequence centred on each amino acid. A series of window lengths were used. Section 3.9 discussed the previous application of machine learning to protein sequences and found a range of optimal window lengths between 3 and 20, so an initial window length range of 1-31 was used. As the results were returned, performance improved with window length, so this range was increased to 1-101 until the results stopped improving.

To get from our windowed sequence to a suitable input vector we employ "one-of-n-encoding". For each residue in the sequence we add 24 columns to the input vector corresponding to the 24 letters in our alphabet: 20 amino acids represented by their standard single character code; "B", "X" and "Z", standing in for ambiguous amino acids; "-" as a dummy character if there are fewer residues to one side of the central character than required by the sequence length. The value of each of these 24 columns is set to zero, apart from a one in the column corresponding to the character in the sequence.

Criterion	Group 1	Group 2
Number of domains	2	2
Minimum number of residues in domain	80	80
Minimum angle of rotation	20°	15°
Maximum intradomain backbone RMSD	2.5 Å	3.0 Å
Maximum number of bending regions	3	5
Maximum number of residues in a bending region	10	15
Number of domain movements before CD Hit filtering (90%)	910	1389
Number of domain movements after CD Hit filtering (90%)	241	372
Number of domain movements after CD Hit filtering (40%)	171	268

Table 6.1 Dataset criteria for training and test data.

10% of each dataset, selected at random, was retained as test data and the remaining 90% was used as the training data. This proportion allowed a larger sample of the dataset to be learned from, but at the expense of a smaller test set. The time taken to train the model meant that 10-fold cross-validation would have been prohibitively computationally expensive.

6.2.2 Redundancy

The CD-hit [98] web server was used to filter the data such that no two proteins had greater than a 90% sequence identity, following Flores et. al[52]. This dataset, referred to as CDHit-90, still allowed for a high likelihood of homologous pairs in the test and training sets, so an additional dataset, CDHit-40, was created using a sequence identity cut-off of 40%. This was selected due to the work of Sander and Schneider [138], as discussed in Section 3.1.1. This resulted in a large reduction in the number of sequences contained in the dataset, so it was deemed appropriate to perform 10-fold cross validation rather than rely on the results of a single partition of test and training data.

Further work was performed using a 20% sequence identity, which is the strictest boundary. This was based on the work of Rost[133], who identified the "twilight zone" of homology to be between 20% and 35% sequence identity. A CD-hit filtered dataset, CDHit-20, was

produced, which contained 136 sequences. Due to concerns that this was too small a dataset to achieve significant results, a series of tests were performed as a form of leave-one-out cross validation of the CDHit-90 and CDHit-40 datasets, where each sequence was selected in turn, and then a training set was produced by removing all of the remaining sequences that had a sequence identity with the selected test sequence greater than or equal to 20%. There were subsequent concerns that this was an unfair test as it was selecting a maximally different training set for each test sequence, so a 10-fold cross validation experiment was also performed on the CDHit-20 data.

6.2.3 Balance

The full group 1 dataset contained 7,395 residues that were considered part of a hinge and 296,818 that were not; positively labelled data was greatly outnumbered by negatively labelled data. The training process takes considerably longer for larger datasets, and because of the large window lengths, the training process was already lengthy. Therefore some negatively labelled data was discarded from the training set in order to bring the total number of training examples down while retaining the same number of positively labelled examples. This was performed using a number of different ratios, but a $1 : 9$ ratio of positive : negative data was selected due to a trade-off between running time and effect on accuracy.

6.3 Results

6.3.1 Hinge Index

In order to get an insight into the propensities of occurrence of amino acids, we calculated the "Hinge Index" (HI) values for each amino acid using the CDHit-90 versions of our group 1 and group 2 datasets, following the process proposed by Flores et al.[52]. This yielded for each amino acid the HI, which measures how over- or under-represented it was in the

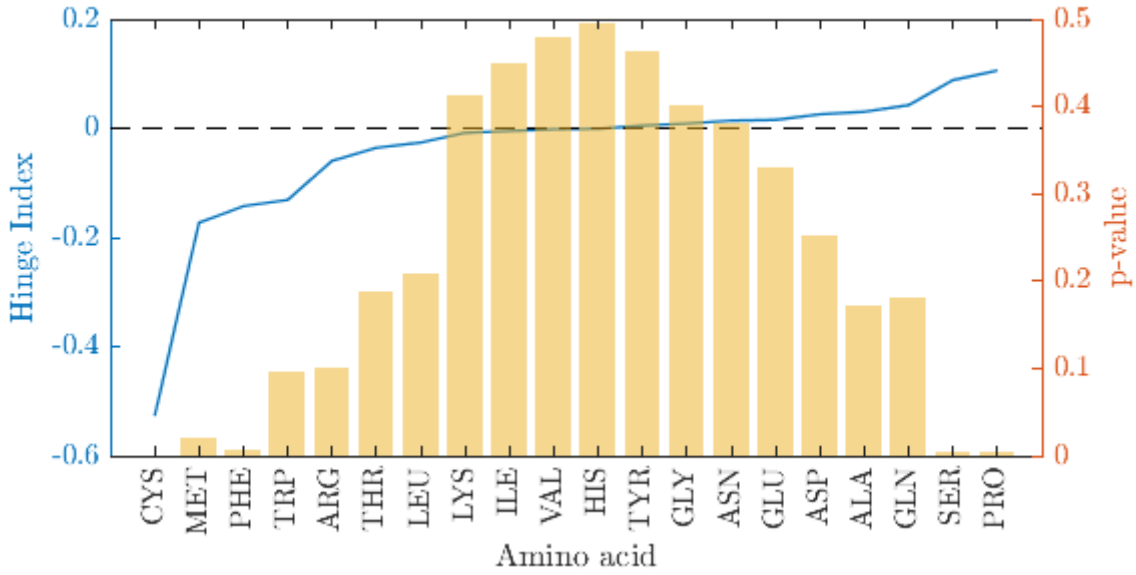


Fig. 6.1 Hinge Index results from Group 1 of CDHit-90.

hinge-bending regions as compared with the rest of the dataset, and its p-value, signifying the statistical significance of the HI.

The HI is calculated for a given amino acid a from the prior probability of an amino acid belonging to class c, $p(a_c)$, and the probability that it occurs in a hinge region (h), $p(a_c|h)$:

$$HI(a_c) = \log_{10} \frac{p(a_c|h)}{p(a_c)} \tag{6.1}$$

These prior probabilities are estimated from frequencies observed in the dataset. A positive HI value indicates the amino acid is over-represented in the hinge-bending regions, whereas a negative value indicates under-representation.

Figure 6.1 shows both of these results for group 1, with the HI value shown by the line and the left y axis, and the p-value shown as a bar and the right y axis. Figure 6.2 shows the same values for group 2. Many of the HI results did not pass the test of statistical significance, in both our calculations and in the original Flores paper, but those that did were generally in agreement. Our most consistent HI results with high significance were those of proline

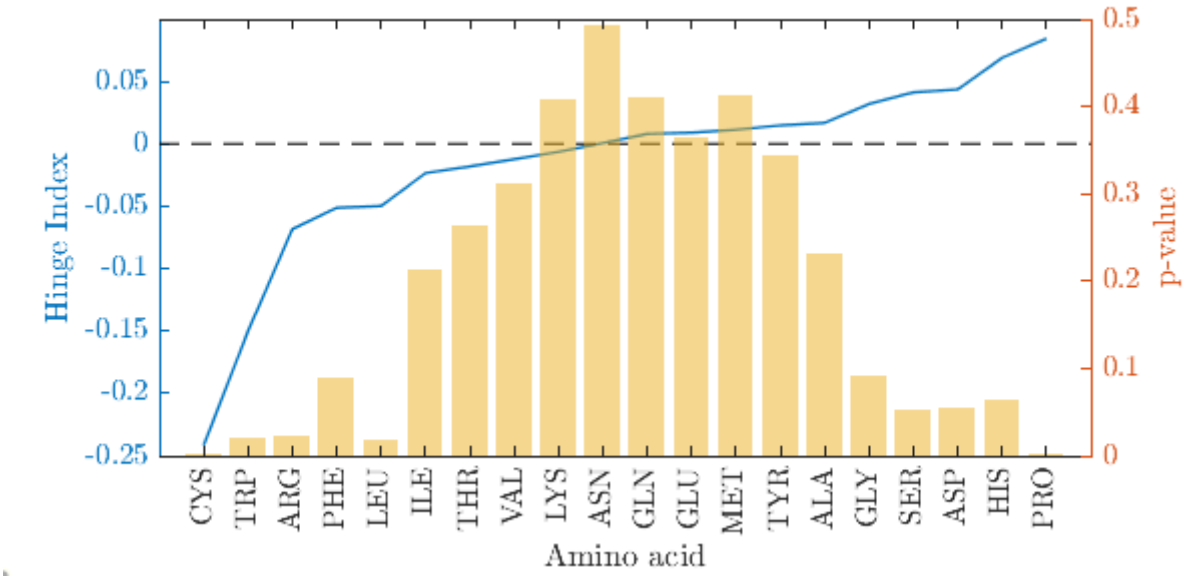


Fig. 6.2 Hinge Index results from Group 2 of CDHit-90.

and cysteine; proline yielded the highest HI in both group 1 and group 2, whereas the HI for cysteine was the lowest in both datasets. The numerical value of these results is consistent with those of Flores et al., though they reported higher p-values for both. Serine, Flores et al.'s most positive HI, was also significantly over-represented in the hinge-bending regions of group 1; serine achieved a positive HI in group 2 but did not quite achieve statistical significance. Glycine, which Flores et al. reported as having a significant high HI, yielded a slightly positive HI without statistical significance in our datasets.

Along with cysteine, group 1's significantly under-represented amino acids were methionine and phenylalanine. In group 2, methionine and phenylalanine did not pass the test of statistical significance; tryptophan, arginine and leucine joined cysteine in the lower end of the HI results in group 2.

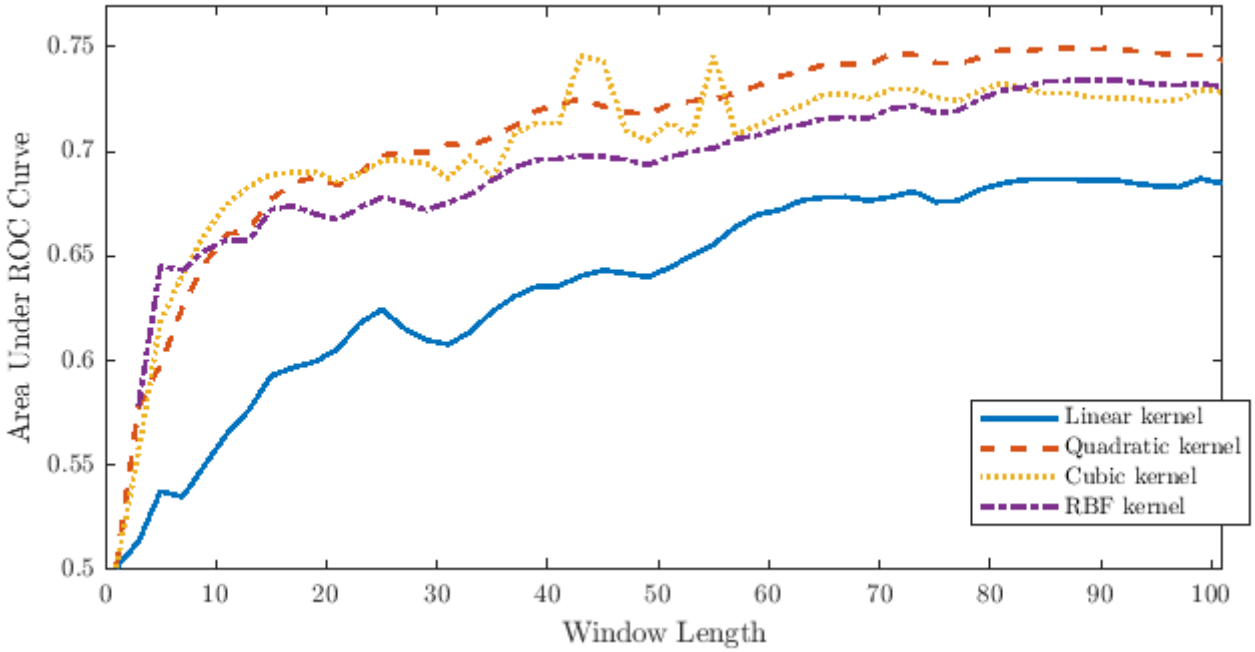


Fig. 6.3 AROC across window length for linear, quadratic, cubic, and RBF KLR models trained on group 1 for CDHit-90.

6.3.2 Accuracy

The areas under the ROC curves (see Section 3.7.3) results from the linear, quadratic, cubic and RBF kernel models on the CDHit-90 filtered group 1 sequences are shown in Figure 6.3. The predictive power increases with the window size beyond the initially considered range; the optimal window length for each of the models is listed in Table 6.2.

To judge the significance of these AROC results, they were compared to the performance of a random classifier with Mann-Whitney U-test [106]. The predicted values from the model are divided into two subsets: one where the true outcome was positive and one where the true outcome was negative. The Mann-Whitney U-test is a test of the hypothesis that the populations have different distributions. In the first result, the input vectors consist of a single amino acid and so the output was the same for each kernel. This failed to reject U-test's null hypothesis; the AROC was 0.501 and the p-value 0.95, showing a result with the same

Kernel	Optimal Window Length	AUC	p-value
Linear	99	0.687	2.9×10^{-20}
Quadratic	87	0.749	9.5×10^{-35}
Cubic	43	0.746	7.0×10^{-34}
RBF	91	0.734	7.0×10^{-31}

Table 6.2 The optimal window length for each kernel trained on group 1, with p-value from Mann-Whitney U-test.

predictive power as a random classification. As the window length increases and the AROC rises, the performance becomes statistically significant. For the linear kernel, this is reached for all window lengths longer than 7 residues. All other kernels fail only when the window length is 1. The values listed in Table 6.3 are the results from each kernel's optimum window length.

The graph appears to show a separation between the linear kernel and the non-linear kernels, which outperform the linear kernels at every window length. We performed a pairwise comparison of the kernels' results using DeLong's algorithm for the comparison of AROC results [39], implemented by Sun and Xu[153]. This algorithm is suitable for such a comparison as it takes into account the correlation of the test and training data between the different kernels' results, as the same partitions were used for all models. Again, results from window length 1 were identical, so no separation was observed. After that, the separation between linear and quadratic results was always significant, with a minimum p-value of 3.53×10^{-12}. Between the quadratic, cubic and RBF kernels, the separation varies across the range of window lengths. The quadratic generally outperforms the RBF kernel, but the cubic kernel is often not significantly different to the performance of the other non-linear kernels. Table 6.3 shows the results for the linear kernel's optimum window length (99).

Figure 6.4 shows the AROC results from group 2. Fewer window lengths were tried due to the increased computational expense of the larger dataset, but the window lengths spanned the same range. The optimal values are listed in Table 6.4 along with the p-values from the

	Linear	**Quadratic**	**Cubic**	**RBF**
Linear	-	1.17×10^{-5}	0.0013	2.22×10^{-5}
Quadratic	1.17×10^{-5}	-	0.0811	0.0091
Cubic	0.0013	0.0811	-	0.8983
RBF	2.22×10^{-5}	0.0091	0.8983	-

Table 6.3 The p-values resulting from statistical comparison of the AUC results from pairs of models trained on group 1, with a window length of 99.

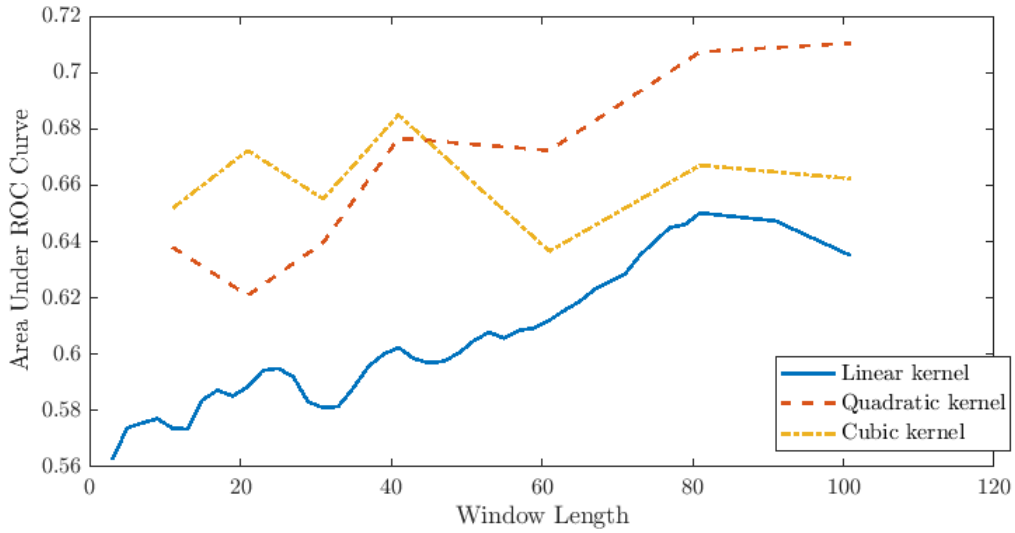


Fig. 6.4 AUC across window length for linear, quadratic, and cubic KLR models trained on group 2.

Mann-Whitney U-test, which all kernels passed at all window lengths. Pairwise comparisons of results using Delong's significance testing are listed in Table 6.5. The linear model showed significantly poorer results than the non-linear models at almost all window lengths.

Each kernel performed worse on the group 2 data than on the data from group 1. A paired t-test was performed on the results at each window length, which rejected the null hypothesis for each kernel; the linear model gave a p-value of 5.3×10^{-8}, the quadratic model 0.0042, and the cubic 2.69×10^{-4}.

Kernel	Optimal Window Length	AUC	p-value
Linear	81	0.650	2.8×10^{-20}
Quadratic	101	0.710	3.3×10^{-40}
Cubic	41	0.685	1.9×10^{-31}

Table 6.4 The optimal window length for each kernel trained on group 2, with p-value from Mann-Whitney U-test.

	Linear	Quadratic	Cubic
Linear	-	4.44×10^{-16}	0.0111
Quadratic	4.44×10^{-16}	-	3.61×10^{-8}
Cubic	0.0111	3.61×10^{-8}	-

Table 6.5 The p-values resulting from statistical comparison of the AUC results from pairs of models trained on group 2, with a window length of 101.

6.3.3 Balancing the Data

The effects of varying the enforced ratio of positively to negatively labelled results can be seen in Figure 6.5, which plots the ROC curves of models trained on data filtered to have different ratios. The effect is minimal, with slightly poorer results from the much larger ratios, possibly due to the lower amount of training data.

6.3.4 Varying the Sequence Identity Threshold

Inspecting the structures allowed in each group indicates that out of the 241 sequences in group 1's CDHit-90 data, a group of 48 antibodies with similar homology dominate the population. This raises the possibility that the models were over-fitting to emphasise features that appear around the hinge-bending regions common to this group.

The weights assigned by the models were extracted as described in detail later in this chapter (see Section 6.3.5), and examined for evidence of such a bias. The highest weighted product feature in this model was the product of serine at 13 residues after the central position, and leucine 9 residues after the central position. A study of the structures where

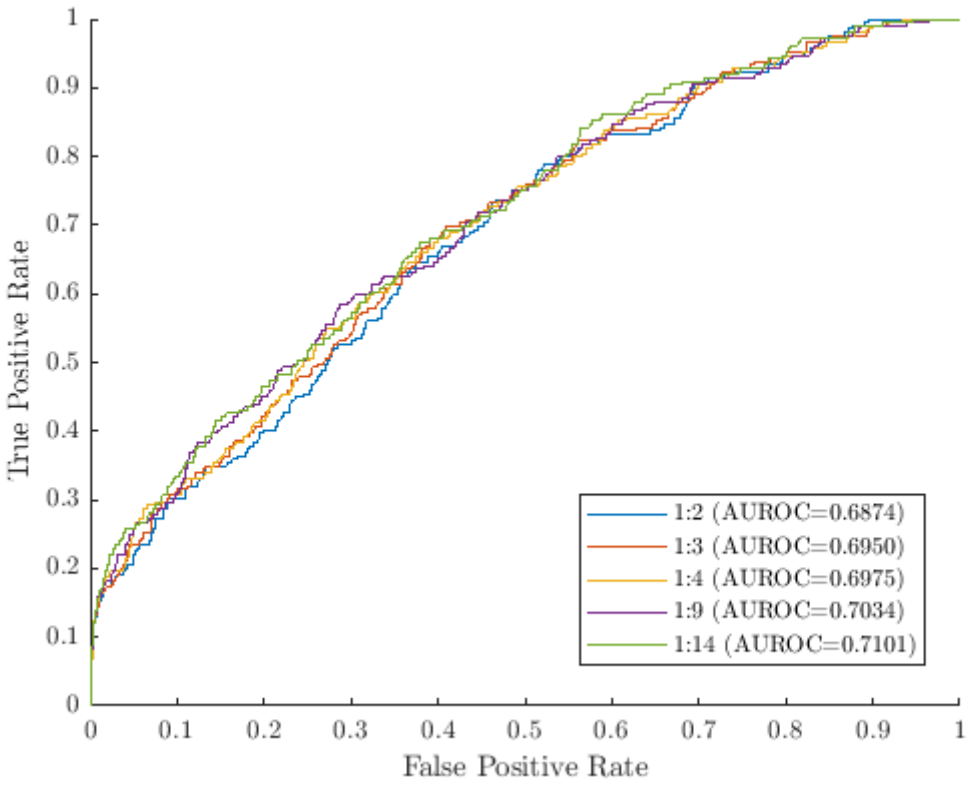


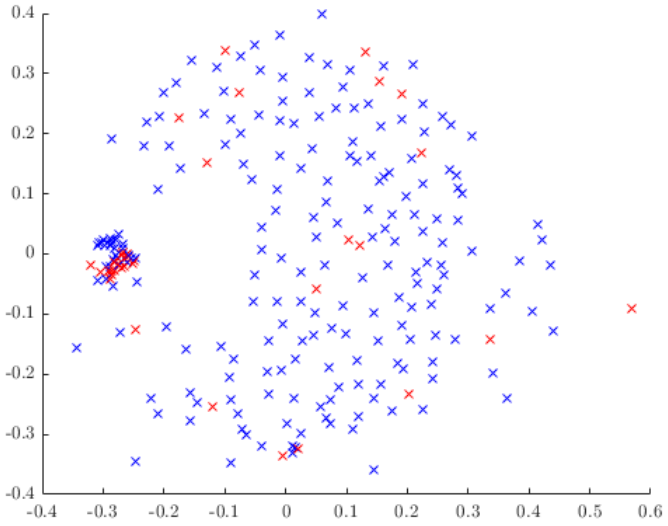
Fig. 6.5 ROC curves for a quadratic model with window length 15 trained on the same dataset randomly discarded down to a range of ratios.

this occurred showed no clear structural reason that this pair would influence the location of the hinge-bending region; in the proteins examined, these residues were at some distance from the bending residues, and would move rigidly as part of their domain with no changes to which other residues they were in contact with. However, the majority of these structures belonged to the group of antibodies.

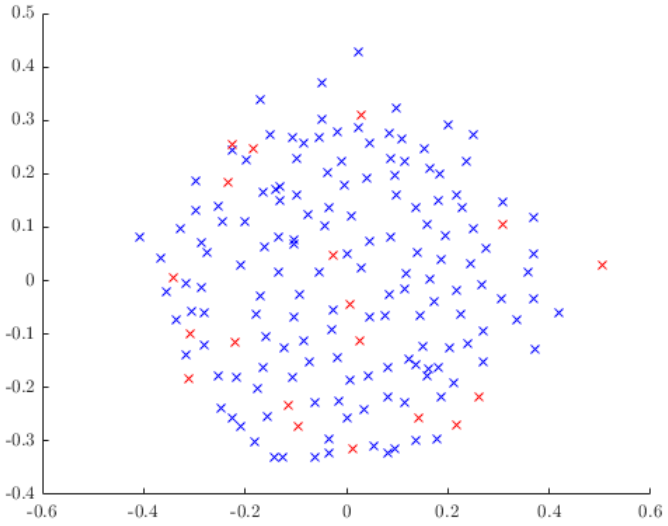
Figure 6.6(a) shows an MDS plot where each point represents a sequence from the CDHit-90 group 1 data. The sequence identity between each pair was calculated, which was used as the similarity matrix for non-metric MDS; more similar pairs of sequences are more closely positioned in the plot. Sequences that exhibited the S13, L9 pair around a hinge-bending region's residue are highlighted in red. The dense cluster on the left consists of antibodies; there is a large subset of these antibodies that contain the feature. Figure 6.6(b) shows the same plot constructed for the CDHit-40 dataset, which shows a more even spread of the sequences. Only 4 of the 171 proteins included in this data are antibodies. Figure 6.6(c) shows the CDHit-20 version of this plot, which shows the most even spread of data of the three plots. 6 out of the 136 sequences are antibodies (the CDHit-20 dataset is not a subset of the CDHit-40 set).

6.3.4.1 40% Sequence Identity

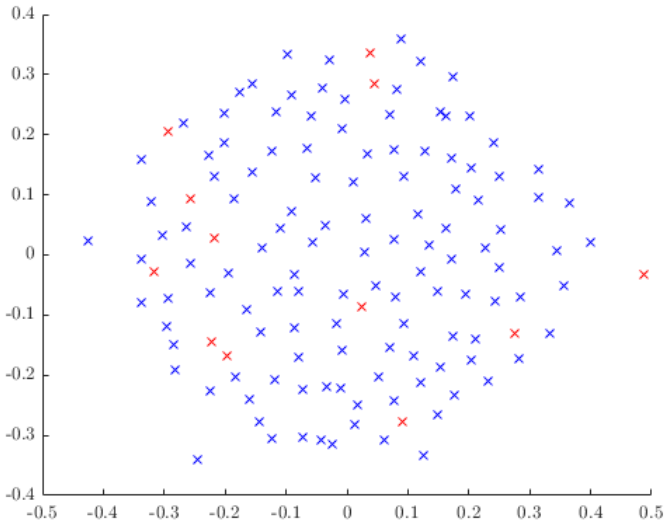
This change in the dataset was influential on the ROCs achieved. CDHit-90's peak result from the quadratic model, 0.749, was considerably higher than the maximum result seen on the equivalent CDHit-40 quadratic peak, 0.61. Similarly, the linear peak was reduced from 0.69 to 0.57. This is not unexpected as the amount of data in the training set was considerably reduced by the increased filtering. Despite the lower peak AROC, the models still exhibited predictive power. The quadratic results are still significantly better than the linear results as shown by a paired t-test of the means of the folds' results; the p-value was 1.0958×10^{-4}.



(a) 90% sequence identity



(b) 40% sequence identity



(c) 20% sequence identity

Fig. 6.6 MDS plot of every sequence in group 1 using pairwise sequence identity to build the similarity matrix. Sequences containing the S13, L9 pair are highlighted in red. Datasets filtered to various sequence identity thresholds.

The results the linear model on each fold of the training data were compared to random chance using the Mann-Whitney U-test, achieving p-values as plotted in Figure 6.8. The required p-value for a confidence level of 95% is indicated with a black dashed line. Most folds of the linear kernel reached the threshold for rejecting the hypothesis of the Mann-Whitney U-test by window length 16, but two of the folds dip above and below this threshold, and one fold only briefly passed below it, indicating that the linear model could not beat random chance when trained on this fold of the training data. Figure 6.9 shows the p-values obtained from the Mann-Whitney U-test performed on the results of the quadratic kernel. Most folds showed no significant difference from the random classifier on the smallest window lengths, but began to reject the null hypothesis after a window length of 20 to 40 residues.

The Hinge Index values and p-values, as plotted for the CDHit-90 dataset in Figure 6.1, were calculated for the CDHit-40 data and plotted in Figure 6.10. Serine and proline are again

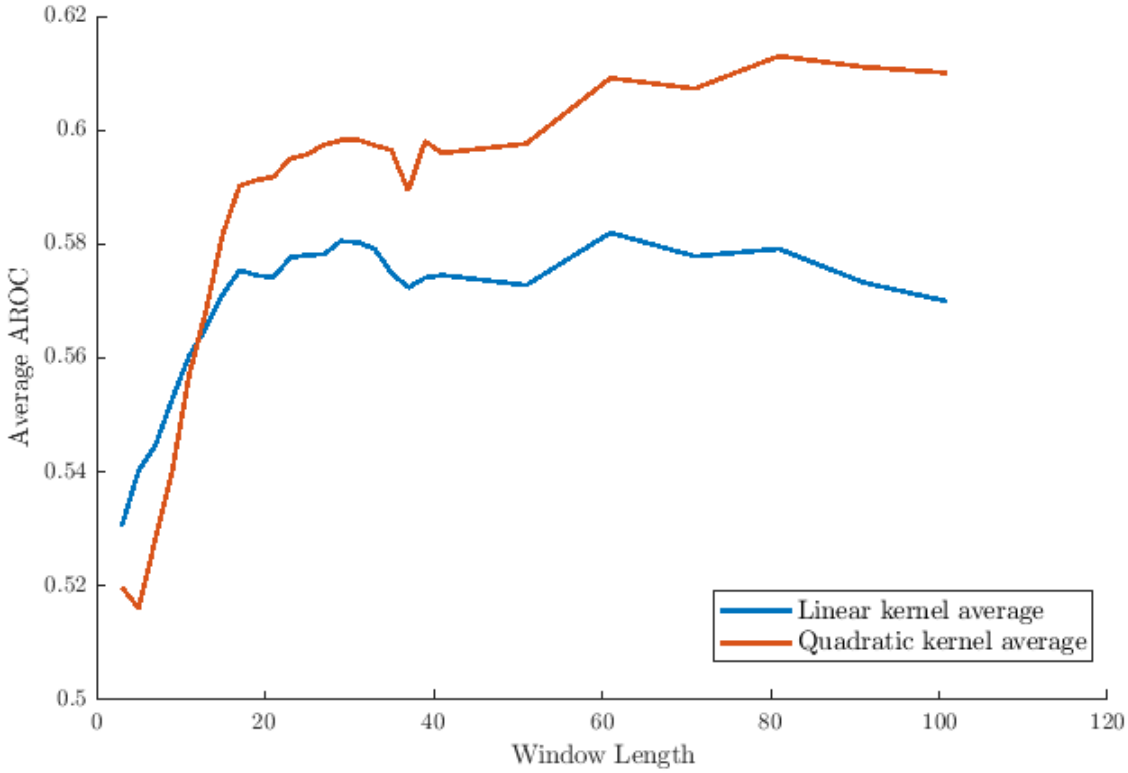


Fig. 6.7 The mean AUC values for 10 folds of results of linear and quadratic models trained on group 1, filtered to CDHit-40.

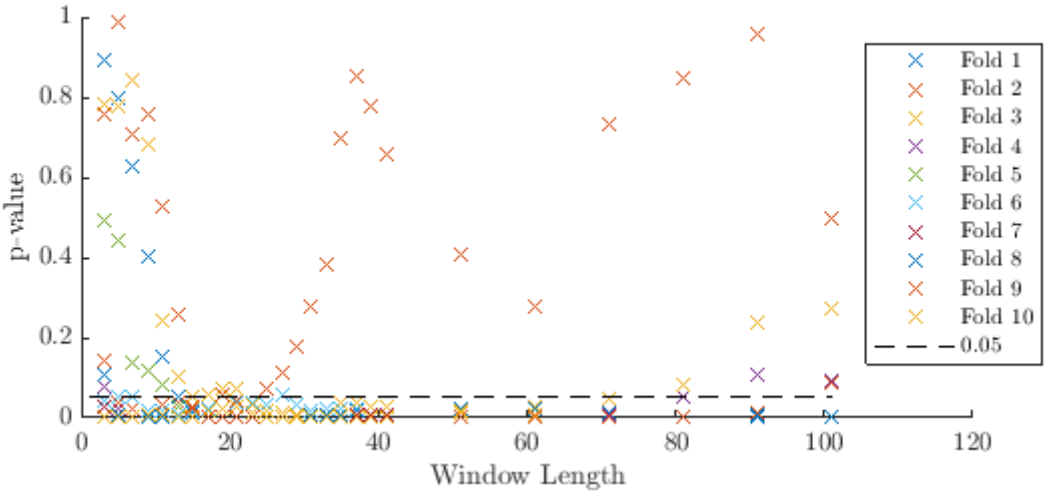


Fig. 6.8 Window length vs p-values for CDHit-40 (linear kernel).

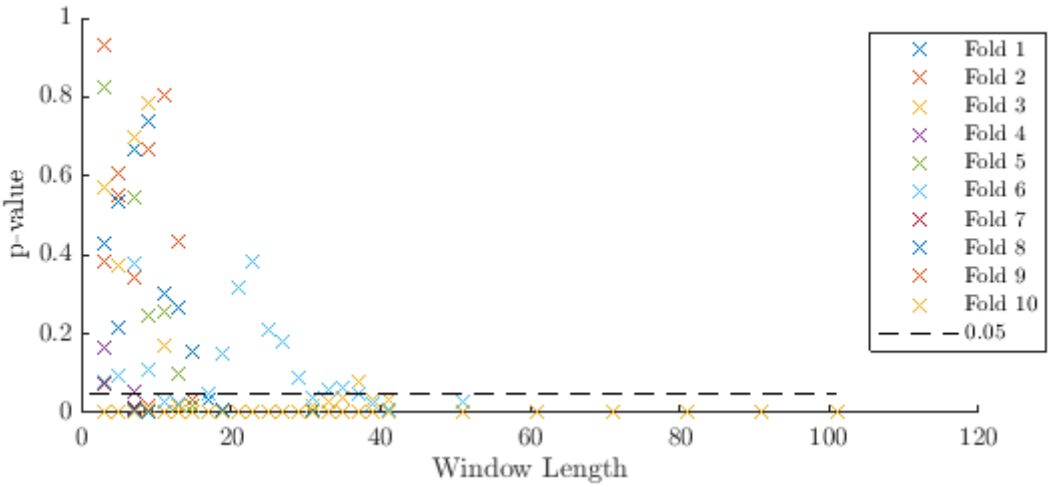


Fig. 6.9 Window length vs p-values for CDHit-40 (quadratic kernel).

significantly over-represented in the hinge-bending region, and cysteine and phenylalanine are under-represented. Methionine receives a similarly negative value for its Hinge Index as for the CDHit-90 group, but in this case it does not achieve a sufficiently low p-value to be considered significant.

6.3.4.2 20% Sequence Identity

Figure 6.11 shows the Hinge Index results calculated from the CDHit-20 data. The HI values for proline, cysteine and phenylalanine continue to support the results found for less strenuously filtered data. The result for serine is still a positive HI value, but a slightly weaker one, with a p-value which does not allow it to be considered significant.

We performed leave-one-out cross validation on the CDHit-90 data, filtering the training data each time to remove sequences with greater than 20% sequence identity with the chosen test case. One window length was used due to the time taken to train each model, 87 was selected as the optimal window length for the CDHit-90 data. The quadratic kernel gave a mean AROC of 0.6058, but the Mann-Whitney U-test showed significance for only 79 out of 241 (32.92%) of the test cases. The same experiment was run using the CDHit-40 data, which gave a mean result for the quadratic kernel of 0.5705. This time, 46 of the 171 cases passed

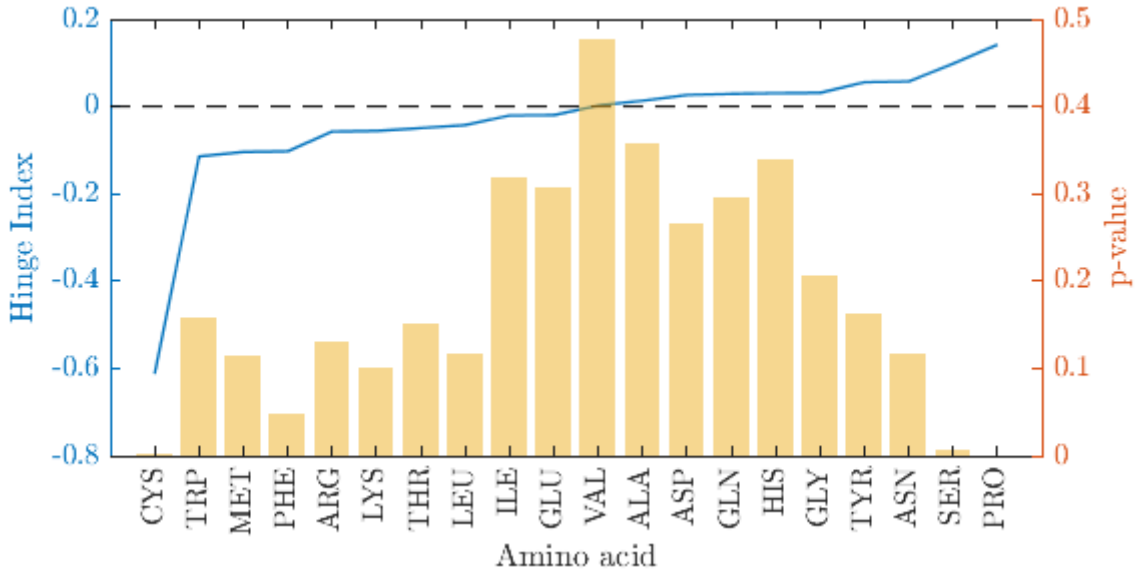


Fig. 6.10 The Hinge Index values calculated on the CDHit-40 group 1 data.

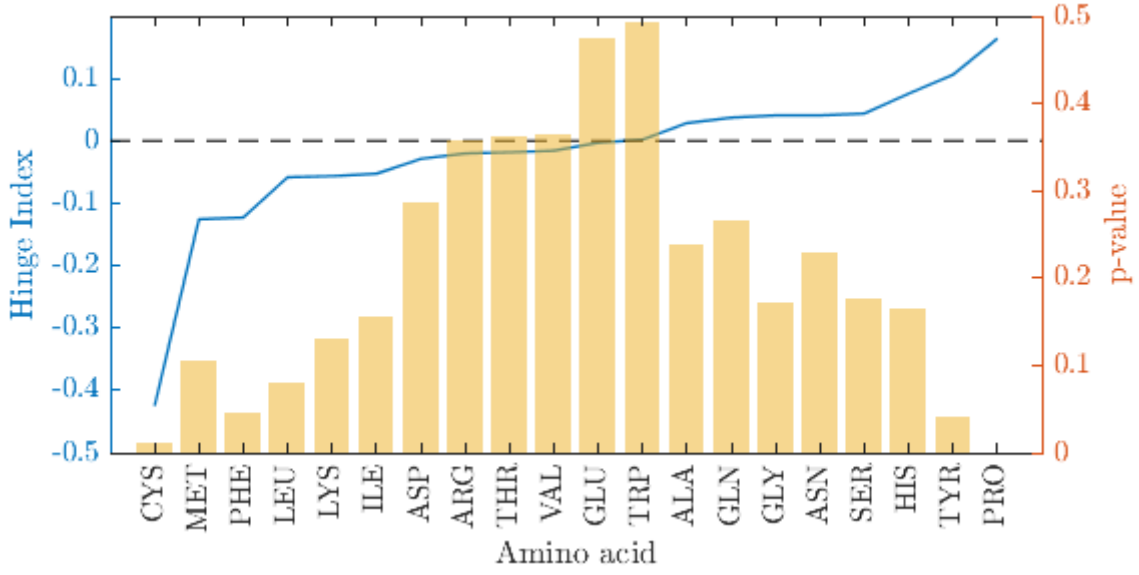


Fig. 6.11 The Hinge Index values calculated on the CDHit-20 group 1 data.

the Mann-Whitney U-test (26.90% of the sequences). Given that each run has the same maximum amount of homology between proteins in training and test data, the difference is performance is potentially due to the reduction in training data. For the linear data, the models trained on the CDHit-90-derived dataset achieved a mean AROC of 0.5875, where 63 of the 241 cases passed the Mann-Whitney U-tests (26.14%), and the models trained on the CDHit-40 data achieved a mean result of 0.5875, with 51 of 171 (29.82%) test cases proving to be significant. While the linear results were poorer for the CDHit-90 data (31.95% were significantly different according to the DeLong significance testing, though this includes both the 130 cases where the quadratic result was higher and the 111 cases where the linear result was higher), they were slightly better for the CDHit-40 cases.

The preceding approach was selected because it would result in more training data than would be obtained by 10-fold cross validation on the CDHit-20 group 1 data. However, in practise it was creating a more difficult problem for the training of each model as every sequence was trained on a maximally different training set, which is reflected in its poor performance. Additionally, the test sets for each fold became very small; each test set consisted of the residues of one sequence, and so contained a few hundred negatively labelled examples and a proportionally very small amount of positively labelled data (on average 8.08 residues per sequence in the CDHit-90 data and 9.04 in the CDHit-40 data), which resulted in low statistical power for the tests of significance. Therefore we also applied the same 10-fold cross validation to the CDHit-20 dataset that was used on the CDHit-40 data.

As with the reduction from CDHit-90 to CDHit-40, the dataset became smaller as the sequence identity restriction became stricter. Again, the performance decreased at this level of sequence identity. Variation between folds was high, and while some folds performed well, others failed to show predictive power. Figures 6.12 and 6.13 show the linear and quadratic kernel models' performance on each of the fold across all tested window lengths. Results from the Mann-Whitney U-tests are listed in Table 6.6, which shows that while some folds

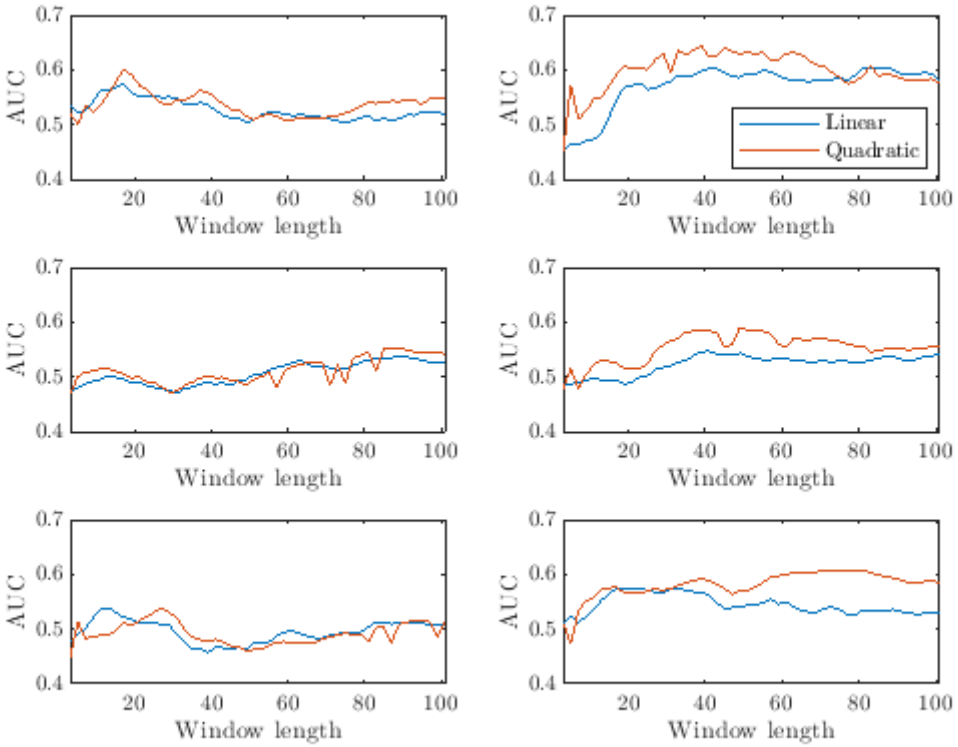


Fig. 6.12 AUC across a range of window lengths for linear and quadratic kernel KLR models trained on folds 1-6 of CDHit-20.

pass this test (and can thus be said to be significantly distinct from the results of a random classifier) on nearly all of their windows for one or both kernels, other folds fail on a majority of windows. The 5th fold in particular resulted in extremely poor performance, wherein no models trained on any window length produced a significant result.

The mean results for each window, combining the performance of all of these folds, are shown in 6.14. The mean results are lower than those of the CDHit-40 and CDHit-90 datasets. The quadratic kernel achieved a maximum average AUC of 0.5722 on window length 89, which supports results from the less stringently filtered datasets where window lengths between 80 and 90 residues were optimal. On the other hand, the linear kernel's maximum result, 0.5570, was achieved with a much shorter window length of 17 residues; AUCs for longer window lengths fluctuated but generally trended downward.

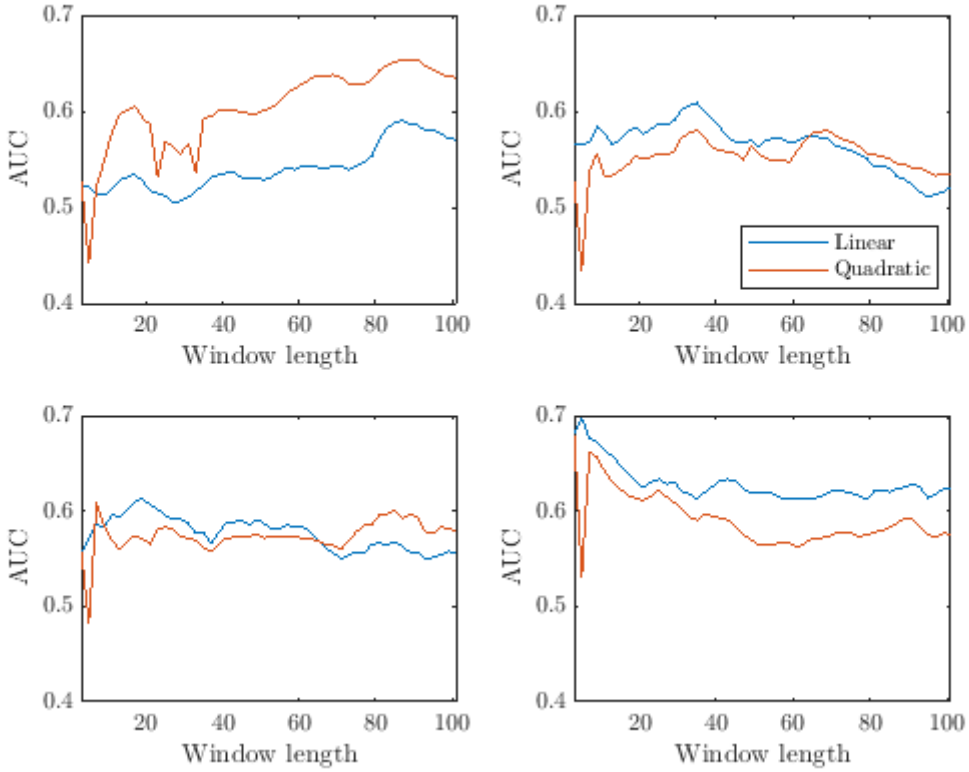


Fig. 6.13 AUC across a range of window lengths for linear and quadratic kernel KLR models trained on folds 7-10 of CDHit-20.

Fold	Linear	Quadratic
1	8.16%	12.24%
2	79.59%	87.76%
3	0.00%	4.08%
4	0.00%	71.43%
5	0.00%	0.00%
6	36.73%	93.88%
7	24.49%	87.76%
8	73.47%	44.90%
9	93.88%	97.96%
10	100.00%	97.96%

Table 6.6 Percentage of window lengths at which models trained on each fold passed a Mann-Whitney U-test.

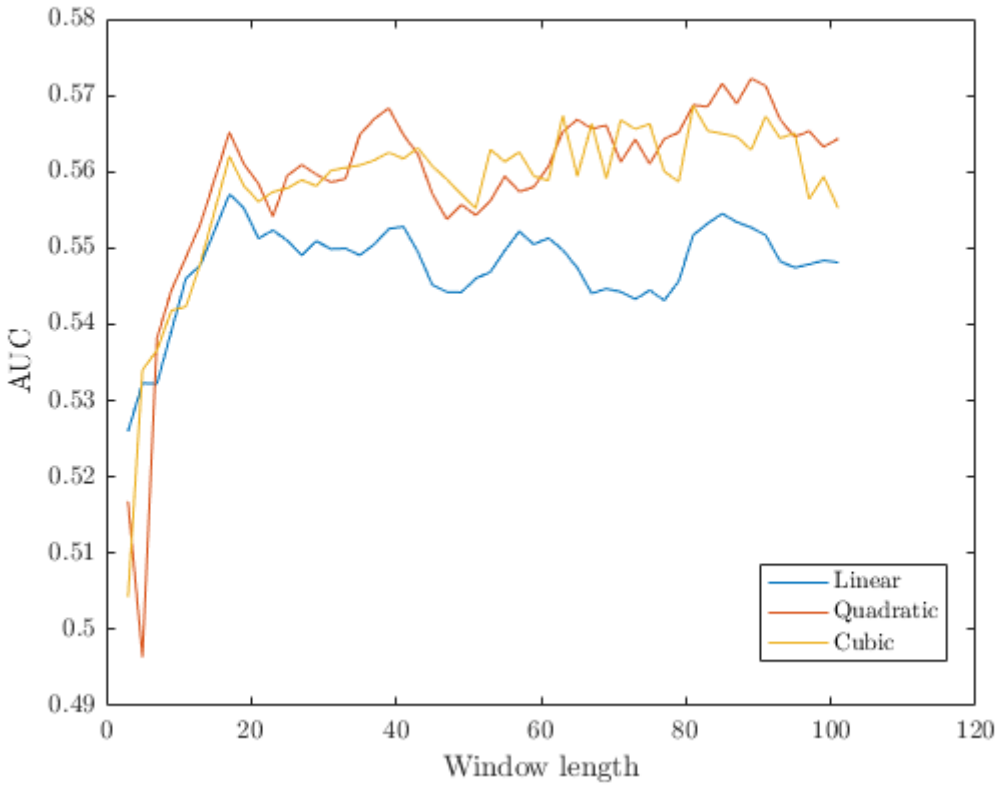


Fig. 6.14 Mean AUC across a range of window lengths for linear, quadratic and cubic kernel KLR models trained on 10 folds of CDHit-20.

While numerically there is a very small difference between the means, at most 0.0216, the quadratic kernel's mean was consistently above the linear kernel's mean. Paired t-testing between the means can reject the null hypothesis with a p-value of 2.5160×10^{-11}. As with previous datasets, there was little separation between the quadratic and cubic kernels.

6.3.5 Weights

We stored the trained models for each of the preceding experiments, including their training samples' input vectors and the α values assigned to them as in the dual model representation of KLR (shown in Equation 3.45). Feature weights can be calculated from these stored values and considered, in order to potentially gain insight into the structure of hinge-bending regions.

6.3.5.1 Linear Weights

The following section describes examples of the information that linear weights can reveal, showing the weights assigned to linear features extracted from KLR models, trained with a linear kernel on CDHit-90 and with a quadratic kernel on CDHit-40 and CDHit-20. All models used the quadratic kernel's optimum window length: 87 on CDHit-90 and CDHit-20 and 81 on CDHit-40. While there are some differences between the models, they are in general agreement, particularly around the strong peaks and troughs. The scale of each model's weights varied, so the weights are plotted as a ratio of each weight divided by the strongest weight assigned by the same model in each graph.

Figure 6.15 shows the weights assigned to proline throughout the window length. As well as the strongest positive HI value, proline was assigned the strongest positive linear weighting in the central residue by all models, indicating a clear preference for being part of the hinge-bending region. The weighting trends negative at either end of the window. The relationship between secondary structure and hinge-bending region makes this an interesting result. Hinge-bending regions have been shown[69] to be more frequent at secondary structure termini, and proline is both suited to terminating secondary structures and unsuited to belonging to α-helices and β-sheets [104]. Proline has also been shown to be overpopulated in linker regions [56], sequences of amino acids that join pairs of domains [62].

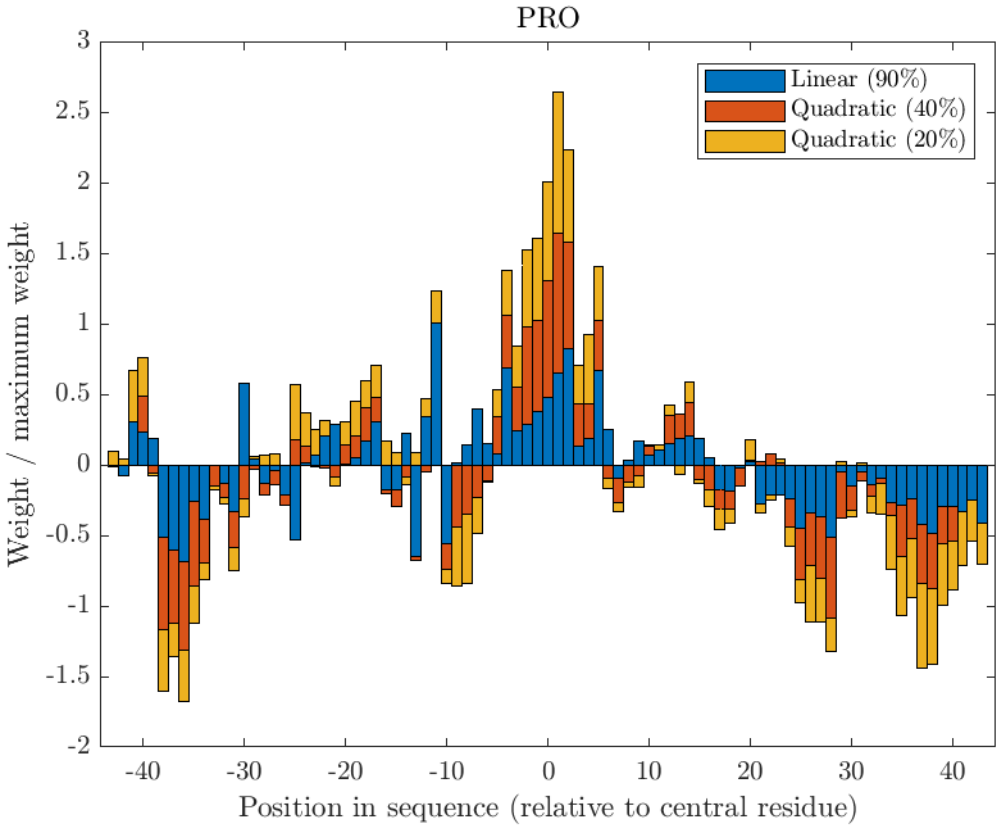


Fig. 6.15 Linear weights for proline residues at various positions in an 87 residue sliding window.

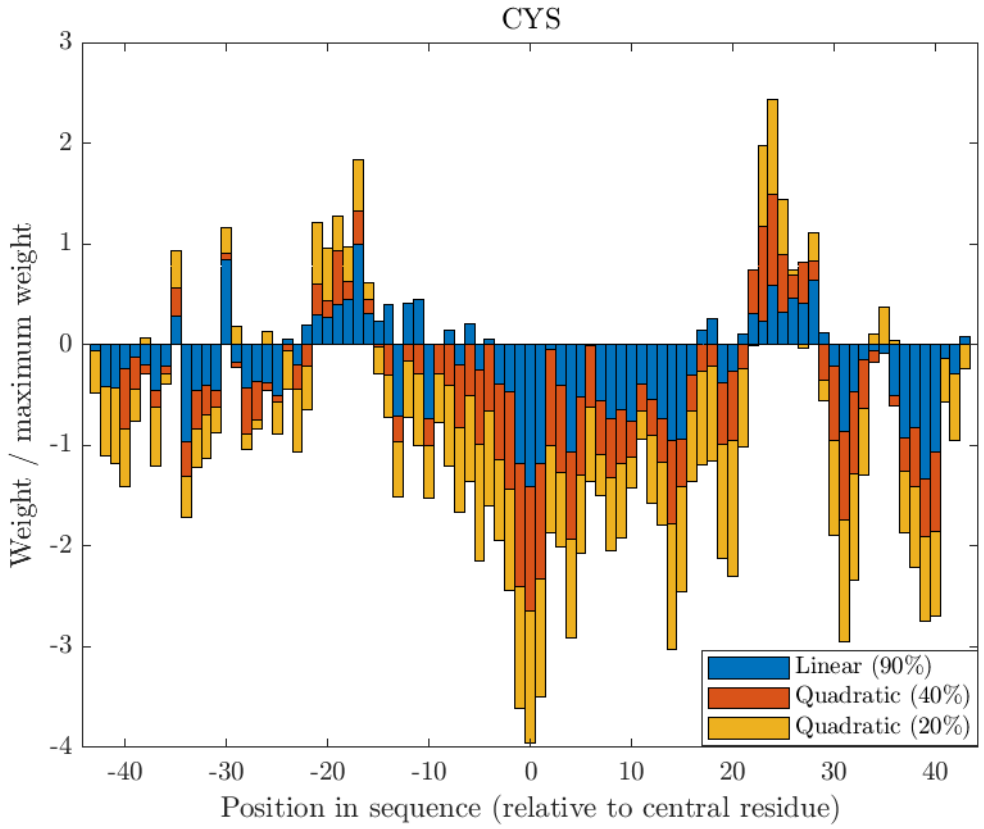


Fig. 6.16 Linear weights for cysteine residues at various positions in an 87 residue sliding window.

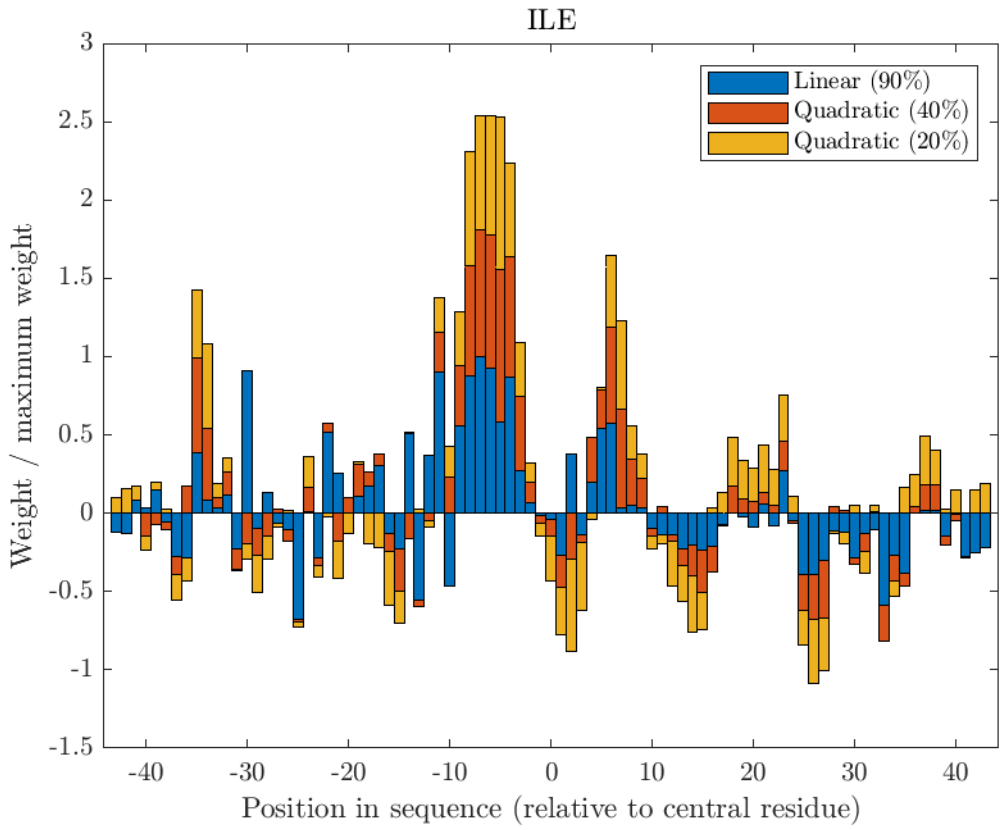


Fig. 6.17 Linear weights for isoleucine residues at various positions in an 87 residue sliding window.

The linear weights assigned to cysteine are plotted in Figure 6.16. Cysteine's disinclination towards being in the hinge-bending region is expressed here as well as in its HI; cysteine received the most negative weighting in the central residue by the linear model, though phenylalanine was more strongly negative in the weights assigned by the quadratic models. Cysteine has positive weightings around 20 residues before and after the centre of the window, indicating a possible preference towards the centre of a domain, which could be due to its ability to form disulphide bonds.

The weights for isoleucine as plotted in Figure 6.17 contain more disagreement than those for proline and cysteine and show either ambivalence or a slight disinclination toward the central residue. The peak around 6 residues before the centre is seen strongly in all three

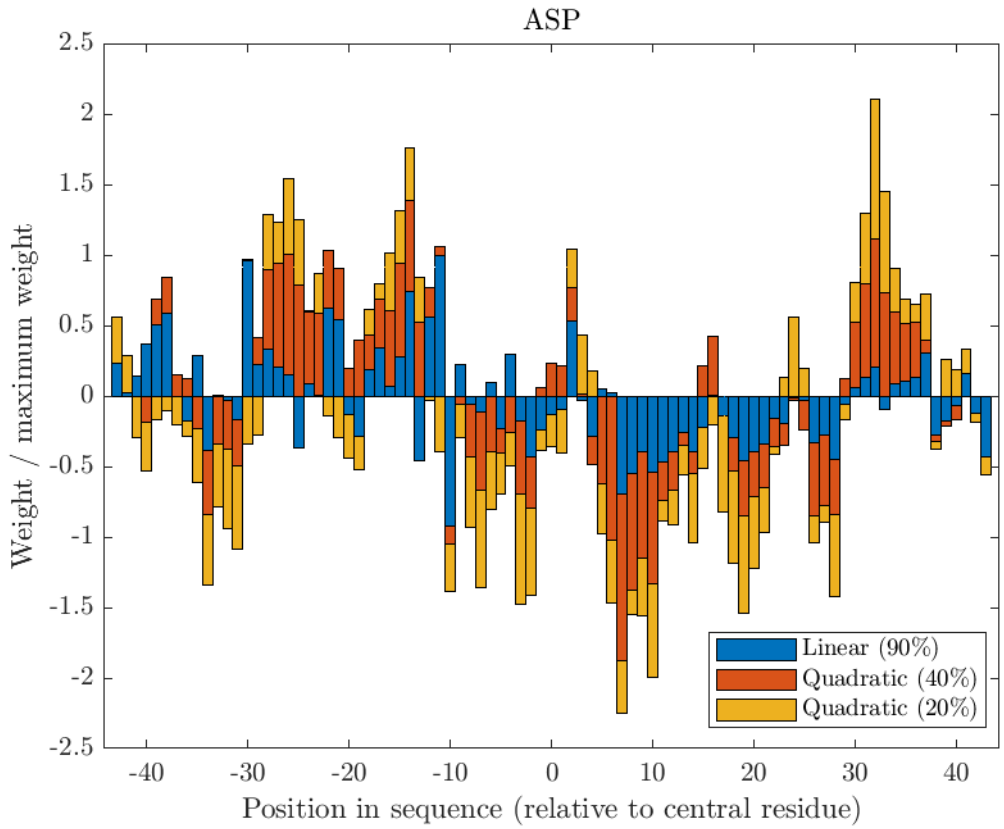


Fig. 6.18 Linear weights for aspartic acid residues at various positions in an 87 residue sliding window.

models, and is among the most strongly weighted linear feature in all three models. A weaker peak can be seen around 6 residues after the central residue.

The results for cysteine, proline and, to a lesser extent, isoleucine display rough symmetry about the central residue. Aspartic acid, as seen in Figure 6.18, displays a less symmetrical pattern of peaks and troughs. It shows ambivalence towards the central residue, with positive peaks around 15 and 25 residues before the centre and 35 residues after, and negative peaks around 35 and 5 residues before the centre and 10, 20 and 30 residues after.

The weights associated with tryptophan, shown in Figure 6.19, are a particularly strong example of asymmetry. Most of the positions are weakly weighted, with disagreement on sign and magnitude between models, but all three are in agreement regarding a large peak around 40 residues after the central residue.

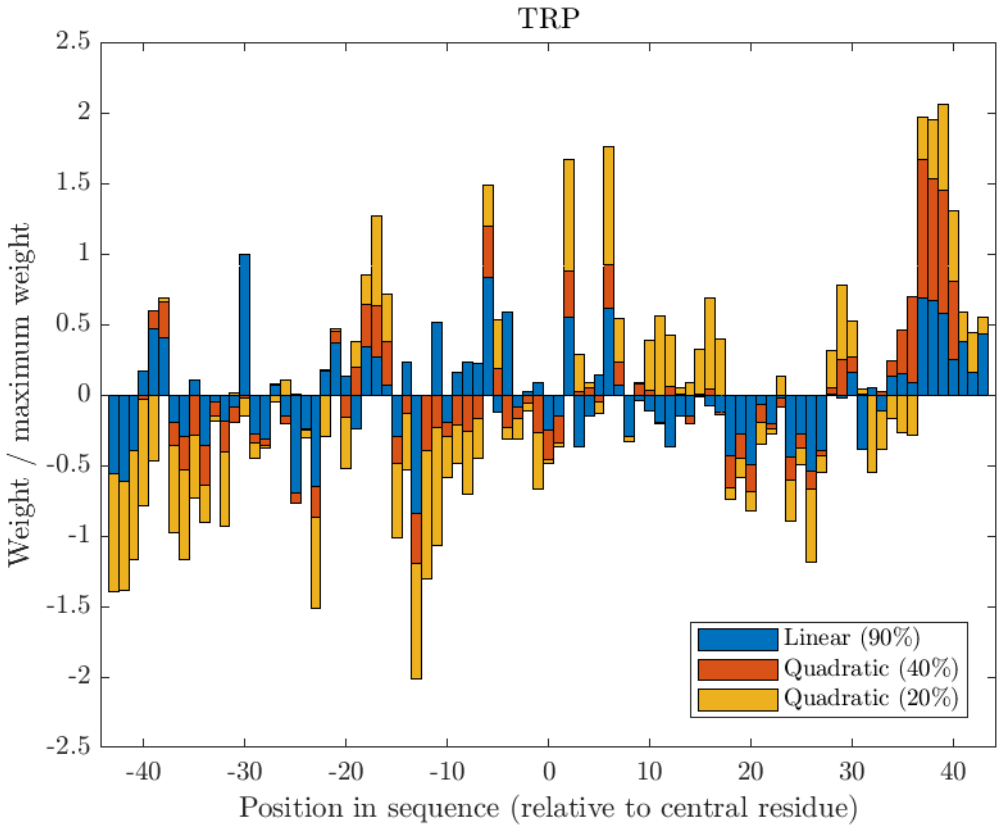


Fig. 6.19 Linear weights for tryptophan residues at various positions in an 87 residue sliding window.

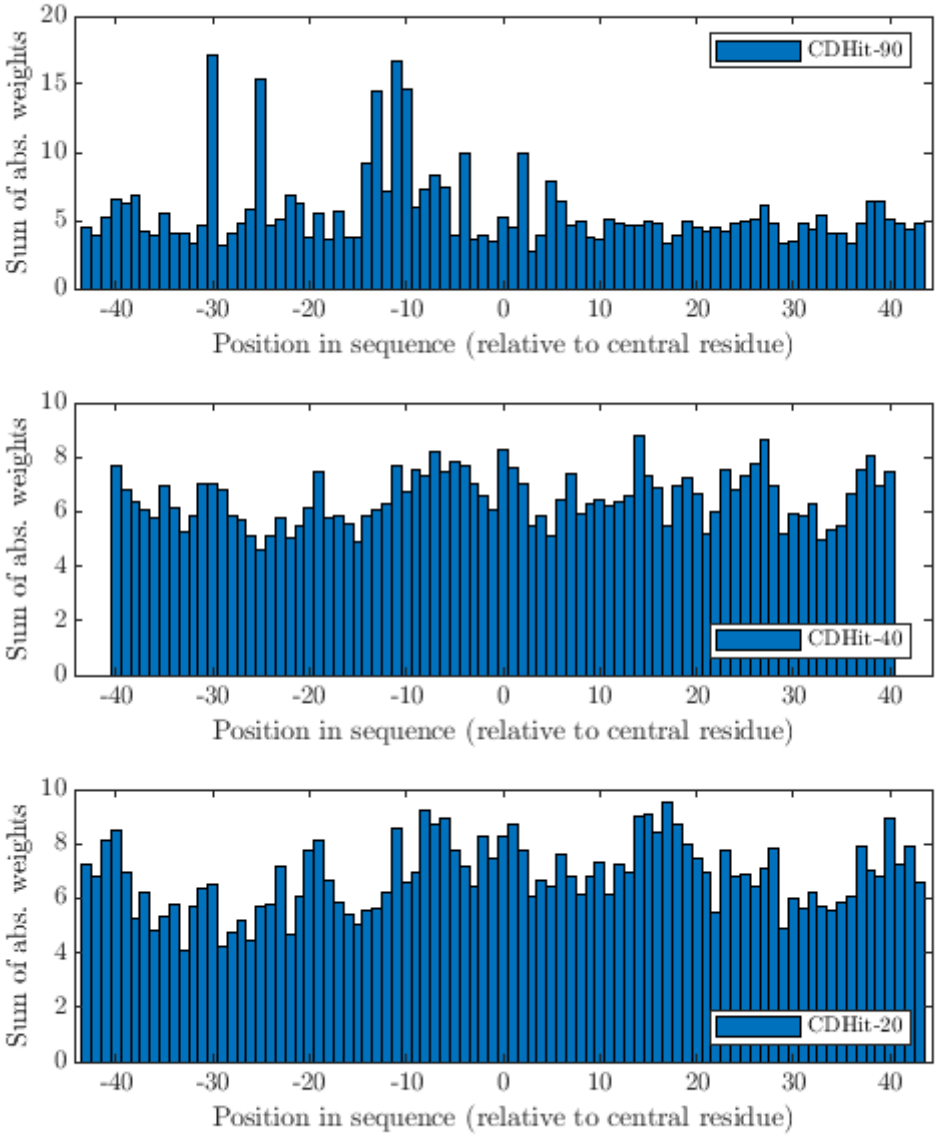


Fig. 6.20 The sum of normalised absolute linear weights from all amino acid.

The results for all amino acids have been combined in Figure 6.20, which shows the sum of absolute weights (normalised by the maximum weight) assigned to all amino acids in each position. These show fluctuation but results in each chart are similar across the entire sequence, indicating that the model's output is equally influenced by the amino acid in the centre of the window as by the distant amino acids up to 43 residues away. The results for the CDHit-90 group are less similar than the corresponding results for the CDHit-40 and CDHit-20 groups, and show strong weights at a small number of positions mostly around or before the centre of the window. Given the finding that a group of proteins within CDHit-90 have high sequence identities, these outlier weights potentially correspond to amino acids in positions that characterise this overrepresented group.

6.3.5.2 Product Weights

In a similar manner, trained quadratic models can be examined to reveal the weights that they associate with product features, indicating the influence on predictions of pairs of amino acids at specific positions.

These extracted weights can be plotted as heat maps, as in Figures 6.21, 6.22, and 6.23. The cells in Figure 6.21 are shaded to represent the weight applied to a pair of amino acids in the sliding window where one of the pair is a cysteine residue and the other is a serine residue. The numbers along the x axis describe the location of the serine residue relative to the central residue of the window, and the y axis gives the location of the cysteine residue relative to the centre of the window. Positive weights are shown in red and negative weights in blue; weights that are close to zero are pale or white. The weights in this graph are mostly dominated by the effects of the cysteine's position, which mirror its linear weights as it shows a generally negative weight, with positive weights when the cysteine residue is around 20 residues before or after the central residue. A patch of strong positive weights around C+25, S+15 suggests that the model has inferred that a serine residue in this location may make

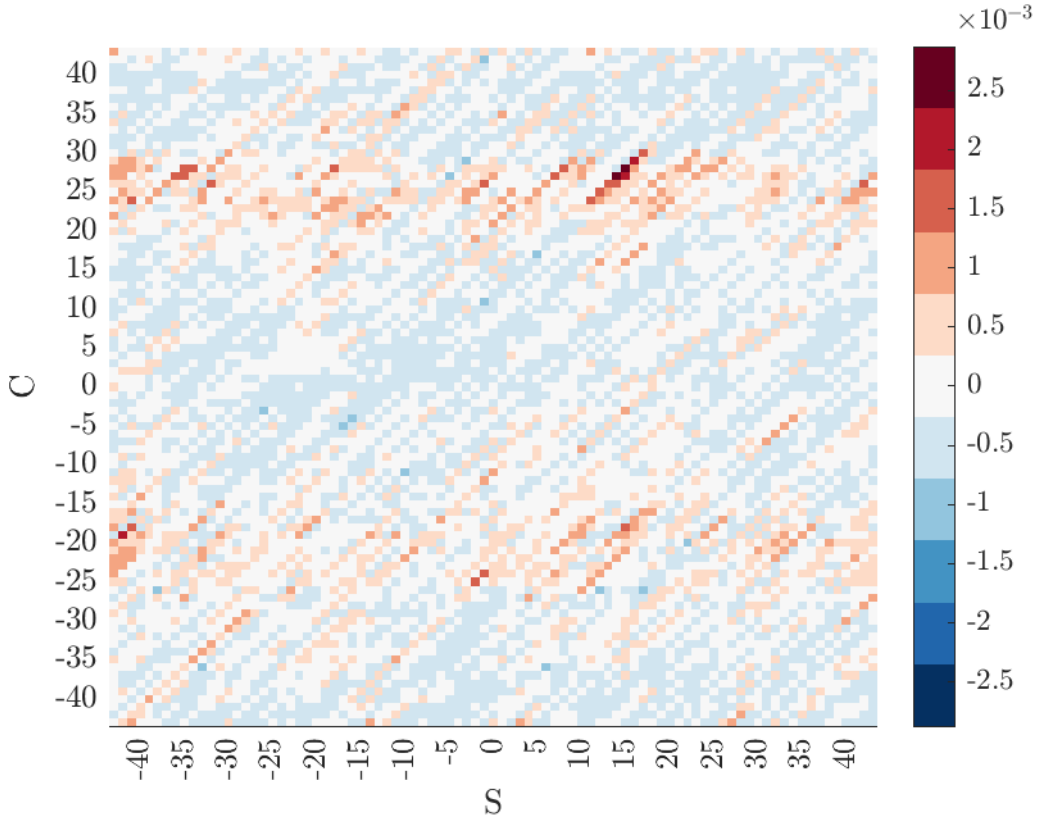


Fig. 6.21 Heat map showing weights extracted from the quadratic model with window length 87, sequence identity 90%, for the product features relating to the positions of cysteine (rows) and serine (columns).

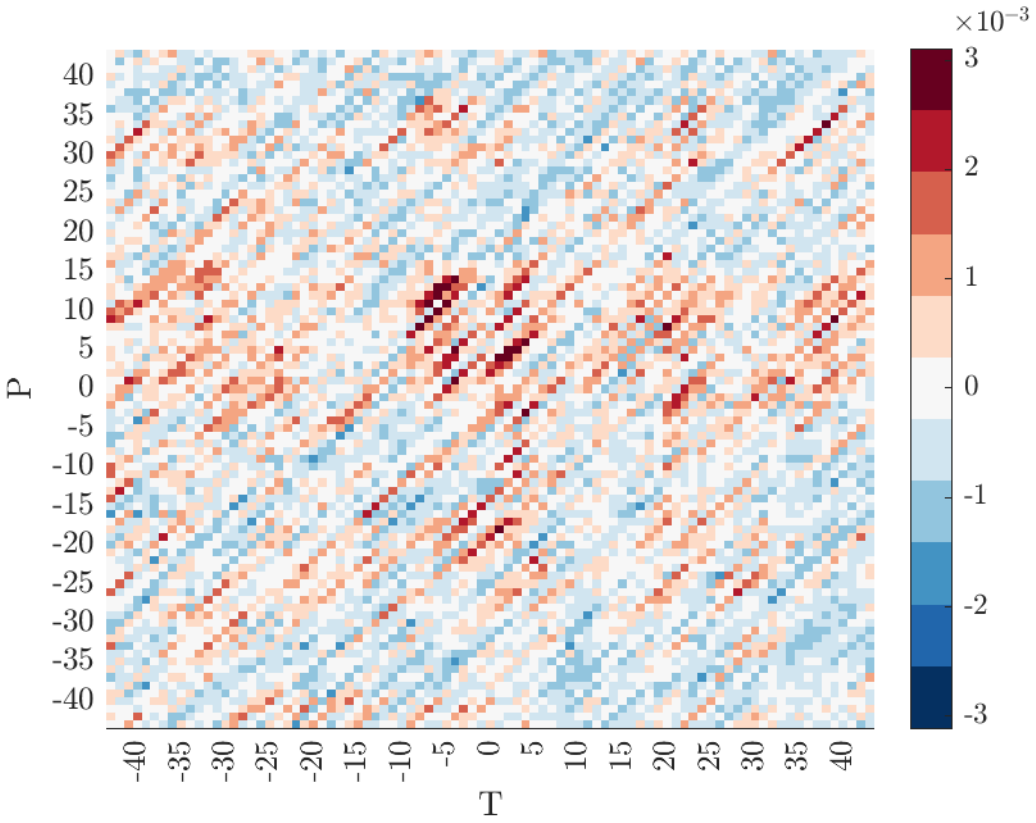


Fig. 6.22 Heat map showing weights extracted from the quadratic model with window length 87, sequence identity 90%, for the product features relating to the positions of proline (rows) and threonine (columns).

the central residue more "hinge-like", though it is hard to say whether this has a structural explanation or whether it is due to being a shared feature in homologous proteins.

Figure 6.22 shows a similar map for proline and threonine. Again, the weights assigned to the product features share similar patterns to the weights assigned to the linear features containing these amino acids; there is a region of mostly positive weighting whenever proline is around the central residue. However, there are areas in the map where the strength or sign of the weight appears to be effected by the presence of a threonine residue: the map has a patch of strong positive weights around the centre, where proline is either the central residue itself or within the next ten residues, and threonine is also around the central residue. Figure 6.23 shows the heat map for the products of proline and asparagine residues, and

while there are some positive weights when proline is around the central residue, the centre of the map is mostly negative, which suggests that the presence of both proline and asparagine around the same residue would cause the model to assign the residue a lower likelihood of belonging to the hinge class. These maps suggest relationships between pairs of amino acids that may offer more subtle discriminatory value available to the quadratic model which would explain its higher performance over the linear model; while proline has generally the highest propensity to being present in a hinge bending region, the presence or absence of other amino acids like threonine or asparagine may support or mitigate this propensity.

The diagonal lines in all heat maps are weighted zero in every cell unless the amino acids in both axes are the same, as two different amino acids cannot both exist at the same position in a sequence.

6.4 Discussion

The location and composition of hinge-bending regions is an important component in understanding a protein domain movement, as it defines how the domains may move relative to one another. The preceding chapter detailed the application of a range of kernel logistic regression models to protein sequence data, labelled by a method which uses solved structural information to identify hinge-bending regions from known conformational change. The trained models were shown to have modest predictive power, showing a statistically significant improvement over a random classifier. While the accuracy of the model is lower than that obtained by machine learning approaches to secondary structure predictions, it showed greater accuracy than the propensity-based method of Flores et al [52] which achieved an AROC of 0.5 when using sequence data alone. KLR models using polynomial kernels were shown to significantly outperform those using linear kernels, indicating that the problem is not linearly separable. The weights assigned by these models to individual and product features were generally consistent across method, window length and sequence identity

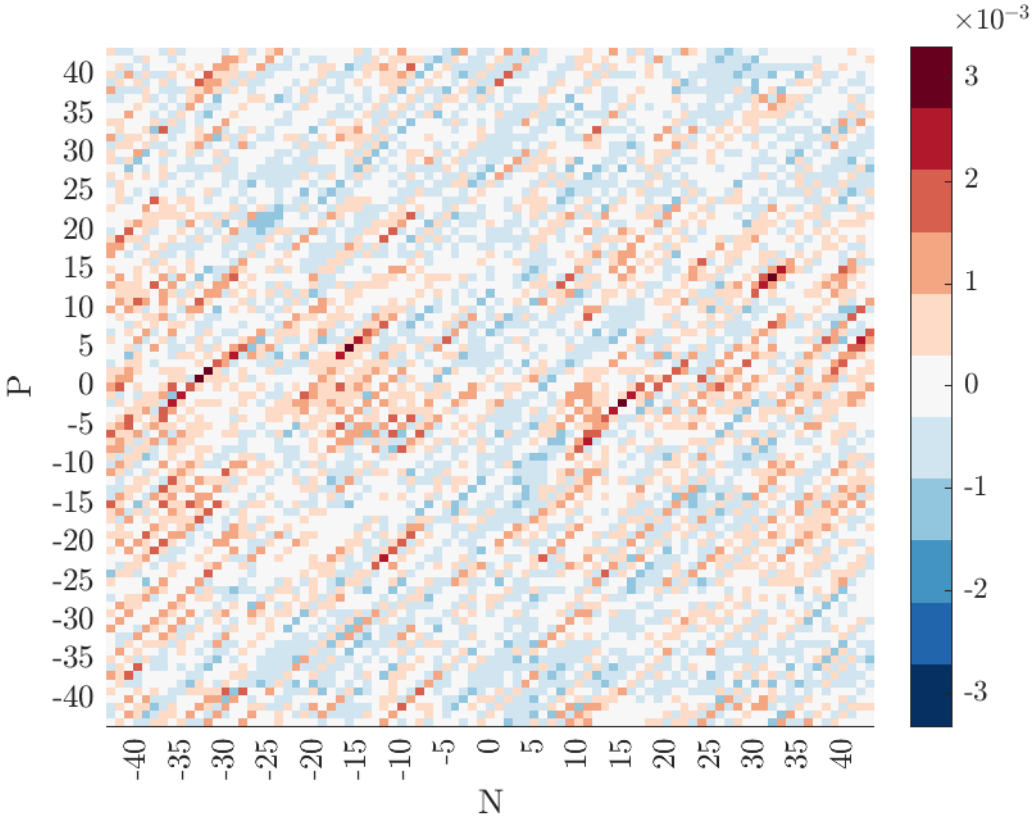


Fig. 6.23 Heat map showing weights extracted from the quadratic model with window length 87, sequence identity 90%, for the product features relating to the positions of proline (rows) and asparagine (columns).

allowed between testing and training data, which could point towards further biological understanding of the composition of hinge-bending regions and their surrounding amino acids.

6.4.1 Accuracy

Several parameters were varied throughout testing, which resulted in a wide range of AUC results.

The length of window used for training and prediction was a key parameter to optimise. Initial values focussed on window lengths between 3 and 30 residues, with an expectation that the highest accuracy would be obtained for window lengths between 11 and 19 following the findings of other works (see Section 3.9) that apply machine learning to other problems of prediction from protein sequence. After these initial tests resulted in optimal window lengths at the higher end of the range, further testing was performed, revealing that the AROC generally increased with window length up to a window between 80 and 90 residues long for the CDHit-90 groups 1 and 2 datasets. These long windows also produced the best performance in the CDHit-40 group 1 data tests, and for the quadratic result of the CDHit-20 dataset. Models trained on the CDHit-20 dataset using a linear kernel had a much smaller optimal window length of 17 residues, which falls more in line with the initially expected results as found in similar works. Given the general poor performance on this dataset, it is unclear whether the linear model's change in behaviour is due to the lack of sufficient data or whether the lower sequence identity threshold has resulted in the removal of homologous proteins that were affecting other datasets' results.

Four kernels were used for the initial KLR tests on the CDHit-90 group 1 data: linear, quadratic, cubic and RBF. The difference between the linear and non-linear kernels had a clear impact on the accuracy on this data, as the quadratic, cubic and RBF kernels significantly outperformed the linear. Among the quadratic, cubic and RBF kernels there was little

difference in accuracy; the quadratic kernel achieved the highest results at longer window lengths, but this difference was not always significant. Due to the similarity in performance, further experiments did not include the RBF and sometimes cubic kernels which were computationally more expensive. The separation between linear and quadratic kernels was seen again in the CDHit-90 group 2 data and in the CDHit-40 data. While performance on the CDHit-20 data was generally poor, a slight but statistically significant separation was found.

As indicated above, the sequence identity allowed between pairs of proteins in the dataset affected accuracy. Lowering this threshold introduces two elements of difficulty: the dataset, which was already stringently filtered, becomes smaller; the proteins within the dataset have less chance of homology. At the 20% sequence identity level we conducted two sets of experiments. In one we created maximally different training sets from larger data sets, which resulted in a generally poor performance, though the statistical power of the significance testing was questionable due to the very small test sets. In the other, we trained the models on folds of a CD-Hit filtered dataset. These models produced lower AROCs than the CDHit-40 and CDHit-90 datasets.

6.4.2 Weight Assignment

A benefit of the KLR method is that the linear and product weights are easy to retrieve from the trained models, which can be examined for potential biological significance. While some strongly weighted pairs appear to be artefacts of overly homologous data, others were consistent across the various experiments.

Linear results, particularly those regarding the central residue, generally supported the Hinge Index findings; proline had the highest HI, and commonly the strongest weighting in the central residue. Cysteine had the lowest HI, and very negative weighting in the central residue, and very positive weighting around 20 residues either side, indication a tendency to

be buried within a domain possibly stemming from its ability to form disulphide bonds. The isoleucine linear weighting provided some information not included in the HI results. While it is ambivalent or weakly disinclined towards the central residue, it is strongly positively weighted in the 5 to 9 residues immediately before it.

6.4.3 Limitations and Further Work

The accuracy of the models is modest, but statistically significant. When filtering the data to no more than 90% sequence identity we achieved a maximum AROC of 0.75, which compares favourably to the results of Flores et al.[52], whose sequence-based predictor had no predictive value after training on data filtered to the same level. A lower accuracy was obtained by models trained on datasets with a lower sequence identity threshold, which may be explained by the smaller amount of examples in these datasets.

The use of one-of-n encoding to build our features could be improved on to potentially improve the accuracy; because our feature vectors are arrays of zeros and ones, the kernel results are based on exact matches, whereas it is known that some pairs of amino acids are more interchangeably than others. The use of string alignment kernels would allow comparisons to incorporate substitution matrices. Additionally, secondary structure prediction using machine learning was greatly improved by using multiple sequence alignments as input, showing possible variation in the sequences. This may have a similar impact here.

The sequences selected for groups 1 and 2 are heavily filtered in order to confidently build a dataset of strong hinge motions, particularly in the requirement that each sequence should contain only two domains. It would be interesting to see whether the findings here are consistent when models are trained or run on sequences from less filtered data.

Chapter 7

Conclusion

The following chapter will summarise how this thesis met its research objectives and outline the key findings revealed. In addition, a number of limitations will be discussed and potential directions for further research will be proposed.

7.1 Thesis Summary

Chapter 1 gave an overview of the motivation behind the study of protein domain movements and explained some of the key concepts that would be required in future chapters. It outlined the aims and objectives for the chapters that followed it and listed those chapters' structure.

Chapters 2 and 3 provided background knowledge and literature review of the biological applications and motivations of the work. Chapter 2 focussed on the structure of proteins and their conformational changes, while Chapter 3 described existing approaches to solving problems like protein morphing, docking, and the assignment of domains and hinge-bending regions. The methods used in later chapters were also covered in this chapter; linear algebra common to several methods was explained, along with various MDS methods. There was a description of the two machine learning algorithms that we used, which were k-means clustering and kernel logistic regression.

In Chapter 4, a method was developed for visualising the conformational changes of biomolecules, either alone or as part of an interaction with another biomolecule. The technique was based on using inter-atomic distances calculated from the solved structures to interpolate ideal distances along the pathway between them. These ideal distances were reconstructed into a set of coordinates using MDS methods. An initial attempt, which used a simple eigenvalue calculation known as classical MDS, achieved results that were no better than a naive interpolation of the Cartesian points, so further "modern" MDS methods were applied to the results of the classical MDS to refine the results. We used a version of MDS that minimises a weighted cost function based on the distance between each atom pair's ideal distance and distance in the created structure, and we applied weights such that the cost function only considers the distances between pairs of atoms that are close enough, applying a cut-off distance that was determined using the MolProbity measure of quality. This resulted in morphs that were much more physically plausible according to the MolProbity validation tool, and which contained less obviously distorted structures. We used a metric proposed by Weiss and Levitt[168] to measure the ability of a morph to recreate a solved intermediate that lies between its start and end confirmations, comparing our method's performance to the published results of other methods. No method scored the highest on every test case, but in a pairwise comparison of the rankings both all-atom and Cα-only MDS morphs outperformed the other methods on more test cases than they were outperformed. We attempted to compare result of an MDS docking morph between a promising cancer drug and its target with a recent and highly accurate MD method, PaCS-MD. Using PaCS-MD we able to simulate a near-native complex with medium accuracy according to the CAPRI guidelines.

Chapter 5 looked at conformational changes as the semi-rigid motions of "dynamic domains", which are parts of the protein that move relative to one another with little internal change. These movements are controlled by hinge-bending regions. The chapter focussed on the development of DynDom6D, a program that finds these dynamic domains by clustering

the components of atoms' motion between one solve structure and another. A grid is constructed over one of the solved structures, and overlapping blocks are constructed at the grid points. The atoms within each block are fitted to the corresponding atoms in the other structure using a quaternion-based least squares best fit procedure. The motion required to bring each block to its second position is expressed as a position in feature space created by the rotation vector, the translation along the axis, and the screw axis location, combined orthogonally. Similar points are gathered using k-means clustering. Each block contributes a vote for its assigned cluster to all of the atoms that it contains; atoms or residues (depending on user preference) are assigned to the domain for which it receives the most votes. The domain movements that the potential domains would produce are examined to determine whether the domains are viable. Atoms or residues with a lot of disagreement during the voting are labelled as hinge-bending regions. A post-processing routine scans to find any points where two residues are adjacent in a chain's sequence but belong to different domains, and creates a small hinge-bending region from the two residues.

Chapter 6 was a description of the development of a tool for predicting the location of hinge-bending regions without including structural information but from sequence alone. Kernel logistic regression was used to train models on labelled sequences taken from the DynDom database. When tested on reserved test sets, the models displayed modest but statistically significant predictive power. It showed that the problem was non-linear, as models that mapped the sequence features into non-linear feature space significantly outperformed the linear method. The effect of window length on the predictions' accuracy was investigated and it was found that the optimum window length for this problem was far longer than that identified by similar machine learning approaches that used a sliding window on protein sequences, applied for the prediction of other properties. The level of redundancy allowed in the testing and training data was examined, and results were poorer when the level of allowed

homology was decreased, but the key findings of the dataset that allowed 90% sequence identity were supported by the results of the more strictly filtered data.

7.2 Objectives and Key Findings

The following are the main objectives of the work, how they were achieved, and the key findings resulting from each objective.

- A **novel visualisation technique** which uses multi-dimensional scaling to construct each frame of a protein morph
  - A protein morphing technique was developed which uses a blend of MDS methods in order to efficiently create intermediate structures between two sets of co-ordinates.
  - The method performed better than comparable techniques, maintaining stronger stereochemical feasibility against the currently most popular technique implemented at the Morph Server, and passing closer to known intermediates on more test cases than any other method.
  - The protein morphing method was extended to work with other types of biomolecules.
  - A further extension was tailored to the problem of visualising protein docking, a novel application of protein morphing that specifically focused on the interaction between two biomolecules.
  - The method was made available as a web server, at http://hingeseek.cmp.uea.ac.uk.
- Development of the next generation of the DynDom **dynamic domain classifier** (DynDom6D).

  - A new program, DynDom6D, was produced which classifies all atoms of a protein, or DNA or RNA molecule, into dynamic domains by clustering on all six dimensions of freedom required to define a rigid body movement.
  - Areas of uncertainty are identified as hinge bending regions.
  - Due to the efficient approach, the program can achieve results on extremely large biomolecules which are becoming more common thanks to next generation methods for solving proteins.

- Machine learning investigation into **finding location of hinge-bending regions from sequence data alone**
  - A series of regression models trained on labelled sequences showed predictive power, albeit limited.
  - Model weights extracted from the most successful KLR models gave some indication towards sequence patterns that occur more prominently around the hinge-bending region.
  - In particular, the discovery of the inflexible proline residue as the amino acid most over-represented in the hinge-bending region is an unexpected and interesting result.

7.3 Limitations and Future Work

7.3.1 Protein Morphing

The MDS protein morphing method makes strict assumptions of linearity which are known to not hold true in all conformational changes. A method that explored unique values for λ was attempted, but the current implementation would approach MD in computational

time. Potential areas of improvement would be either a more efficient implementation of the optimisation process, perhaps assisted with future developments in GPU technology. Or a smaller optimisation problem using one single λ function for each pair of atoms in two rigid areas, perhaps using a version of the DynDom6D program to identify a fixed amount of smaller clusters of rigidity.

In finding the optimal cut-off lengths, it can be observed that varying the cut-off produces very different results for each protein. A pattern of identifying the optimal cut-off from the geometry of the structure was not observed from the experiments conducted, but further investigations with more sophisticated regression techniques may be more successful, and allow a version of the method that selected a set of predicted optimal cut-off lengths based on the input structures.

7.3.2 DynDom6D

While the DynDom6D method performed efficiently, and was able to identify interesting dynamic domains on our test examples, it required some experimentation with parameters to achieve each result. There was no set of parameters which performed well on all test cases. However, some simple patterns were potentially observed, such as larger grid sizes helping for larger molecules. An automated optimisation process would not be easy for this problem, as there is no objective method by which the results can be judged - results of previous iterations of DynDom can only be obtained for single backbone chains (DynDom) or smaller molecules (DynDom3D). Further investigations into parameter selection would make this program more easily usable.

7.3.3 Hinge-Bending Prediction

The results of the KLR investigations into prediction of hinge-bending regions showed some weak predictive power. Further work could be done to improve these results. The current

kernels only take into account exact matches of amino acids in a given position, whereas there are known substitution matrices that describe the interchangeability of pairs of amino acids. String alignment kernels using these substitution matrices may provide better accuracy.

References

[1] Abraham, M. J., Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B., and Lindahl, E. (2015). Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 1:19–25.

[2] Amaro, R. E., Baudry, J., Chodera, J., Demir, Ö., McCammon, J. A., Miao, Y., and Smith, J. C. (2018). Ensemble docking in drug discovery. *Biophysical journal*, 114(10):2271–2278.

[3] Andreeva, A., Howorth, D., Chothia, C., Kulesha, E., and Muzin, A. G. (2014). Scop2 prototype: a new approach to protein structure mining (vol 42, pg d310, 2014). *NUCLEIC ACIDS RESEARCH*, 42(18):11847–11847.

[4] Bahar, I., Lezon, T. R., Bakan, A., and Shrivastava, I. H. (2010). Normal mode analysis of biomolecular structures: functional mechanisms of membrane proteins. *Chemical reviews*, 110(3):1463–1497.

[5] Bai, X.-C., McMullan, G., and Scheres, S. H. (2015). How cryo-em is revolutionizing structural biology. *Trends in biochemical sciences*, 40(1):49–57.

[6] Beckwitt, E. C., Kong, M., and Van Houten, B. (2018). Studying protein-dna interactions using atomic force microscopy. In *Seminars in cell & developmental biology*, volume 73, pages 220–230. Elsevier.

[7] Berg, J., Tymoczko, J., and Stryer, L. (2001). Three-dimensional protein structure can be determined by nmr spectroscopy and x-ray crystallography. *Biochemistry, 5th ed. WH Freeman and Company*.

[8] Berkhin, P. (2006). A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer.

[9] Berman, H., Henrick, K., Nakamura, H., and Markley, J. L. (2007). The worldwide protein data bank (wwpdb): ensuring a single, uniform archive of pdb data. *Nucleic acids research*, 35(suppl_1):D301–D303.

[10] Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., et al. (2002). The protein data bank. *Acta Crystallographica Section D: Biological Crystallography*, 58(6):899–907.

[11] Bhagavan, N. V. (2002). *Medical biochemistry*. Academic press.

[12] Bodén, M. and Bailey, T. L. (2006). Identifying sequence regions undergoing conformational change via predicted continuum secondary structure. *Bioinformatics*, 22(15):1809–1814.

[13] Bodén, M., Yuan, Z., and Bailey, T. L. (2006). Prediction of protein continuum secondary structure with probabilistic models based on nmr solved structures. *BMC bioinformatics*, 7(1):68.

[14] Bonvin, A. M. (2006). Flexible protein–protein docking. *Current opinion in structural biology*, 16(2):194–200.

[15] Borg, I. and Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

[16] Bourne, P. E., Berman, H. M., McMahon, B., Watenpaugh, K. D., Westbrook, J. D., and Fitzgerald, P. M. (1997). [30] macromolecular crystallographic information file. In *Methods in enzymology*, volume 277, pages 571–590. Elsevier.

[17] Boutet, E., Lieberherr, D., Tognolli, M., Schneider, M., and Bairoch, A. (2007). Uniprotkb/swiss-prot. In *Plant bioinformatics*, pages 89–112. Springer.

[18] Boyd, S. and Vandenberghe, L. (2018). *Introduction to applied linear algebra: vectors, matrices, and least squares*. Cambridge university press.

[19] Boyer, P. D. (1997). The atp synthase—a splendid molecular machine. *Annual review of biochemistry*, 66(1):717–749.

[20] Brams, S. J. and Fishburn, P. C. (2002). Voting procedures. *Handbook of social choice and welfare*, 1:173–236.

[21] Bronstein, M. M., Bronstein, A. M., Kimmel, R., and Yavneh, I. (2005). A multigrid approach for multidimensional scaling. In *Proc. Copper Mountain Conf. Multigrid Methods*.

[22] Bronstein, M. M., Bronstein, A. M., Kimmel, R., and Yavneh, I. (2006). Multigrid multidimensional scaling. *Numerical linear algebra with applications*, 13(2-3):149–171.

[23] Busia, A. and Jaitly, N. (2017). Next-step conditioned deep convolutional neural networks improve protein secondary structure prediction. *arXiv preprint arXiv:1702.03865*.

[24] Case, D. A. (1994). Normal mode analysis of protein dynamics. *Current Opinion in Structural Biology*, 4(2):285–290.

[25] Case, D. A., Cheatham, T. E., Darden, T., Gohlke, H., Luo, R., Merz, K. M., Onufriev, A., Simmerling, C., Wang, B., and Woods, R. J. (2005). The amber biomolecular simulation programs. *Journal of computational chemistry*, 26(16):1668–1688.

[26] Cavalli, A., Salvatella, X., Dobson, C. M., and Vendruscolo, M. (2007). Protein structure determination from nmr chemical shifts. *Proceedings of the National Academy of Sciences*, 104(23):9615–9620.

[27] Cawley, G. C., Janacek, G. J., and Talbot, N. L. (2007). Generalised kernel machines. In *2007 International Joint Conference on Neural Networks*, pages 1720–1725. IEEE.

[28] Chasles, M. (1830). Note sur les propriétés génerales du système de deux corps semblables entr'eux et placés d'une manière quelconque dans l'espace; et sur le déplacement fini ou infiniment petit d'un corps solide libre [a note on the general properties of a system of two similar bodies arbitrarily positioned in space; and on the finite or infinitely small displacement of an unconstrained solid body]. *Bulletin des Sciences Mathématiques, Férussac*, 14:321–26.

[29] Chen, V. B., Arendall, W. B., Headd, J. J., Keedy, D. A., Immormino, R. M., Kapral, G. J., Murray, L. W., Richardson, J. S., and Richardson, D. C. (2010). Molprobity: all-atom structure validation for macromolecular crystallography. *Acta Crystallographica Section D: Biological Crystallography*, 66(1):12–21.

[30] Cheng, X., Veverka, V., Radhakrishnan, A., Waters, L. C., Muskett, F. W., Morgan, S. H., Huo, J., Yu, C., Evans, E. J., Leslie, A. J., et al. (2013). Structure and interactions of the human programmed cell death 1 receptor. *Journal of Biological Chemistry*, 288(17):11771–11785.

[31] Chiang, M. M.-T. and Mirkin, B. (2010). Intelligent choice of the number of clusters in k-means clustering: an experimental study with different cluster spreads. *Journal of classification*, 27(1):3–40.

[32] Congreve, M., Murray, C. W., and Blundell, T. L. (2005). Keynote review: Structural biology and drug discovery. *Drug discovery today*, 10(13):895–907.

[33] Cox, M. A. and Cox, T. F. (2008). Multidimensional scaling. In *Handbook of data visualization*, pages 315–347. Springer.

[34] De Leeuw, J. (1988). Convergence of the majorization method for multidimensional scaling. *Journal of classification*, 5(2):163–180.

[35] De Leeuw, J. and Mair, P. (2011). Multidimensional scaling using majorization: Smacof in r.

[36] De Ruyck, J., Brysbaert, G., Blossey, R., and Lensink, M. F. (2016). Molecular docking as a popular tool in drug design, an in silico travel. *Advances and applications in bioinformatics and chemistry: AABC*, 9:1.

[37] De Vivo, M., Masetti, M., Bottegoni, G., and Cavalli, A. (2016). Role of molecular dynamics and related methods in drug discovery. *Journal of medicinal chemistry*, 59(9):4035–4061.

[38] DeLano, W. L. et al. (2002). Pymol: An open-source molecular graphics tool. *CCP4 Newsletter on protein crystallography*, 40(1):82–92.

[39] DeLong, E. R., DeLong, D. M., and Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, 44(3):837–845.

[40] D'incecco, A., Andreozzi, M., Ludovini, V., Rossi, E., Capodanno, A., Landi, L., Tibaldi, C., Minuti, G., Salvini, J., Coppi, E., et al. (2015). Pd-1 and pd-l1 expression in molecularly selected non-small-cell lung cancer patients. *British journal of cancer*, 112(1):95.

[41] Durrant, J. D. and McCammon, J. A. (2011). Molecular dynamics simulations and drug discovery. *BMC biology*, 9(1):71.

[42] Ehrlich, L. P., Nilges, M., and Wade, R. C. (2005). The impact of protein flexibility on protein–protein docking. *Proteins: Structure, Function, and Bioinformatics*, 58(1):126–133.

[43] El-Gebali, S., Mistry, J., Bateman, A., Eddy, S. R., Luciani, A., Potter, S. C., Qureshi, M., Richardson, L. J., Salazar, G. A., and Smart, A. (2018). The pfam protein families database in 2019. *Nucleic acids research*, 47(D1):D427–D432.

[44] Emekli, U., Schneidman-Duhovny, D., Wolfson, H. J., Nussinov, R., and Haliloglu, T. (2008). Hingeprot: automated prediction of hinges in protein structures. *Proteins: Structure, Function, and Bioinformatics*, 70(4):1219–1227.

[45] Engh, R. A. and Huber, R. (1991). Accurate bond and angle parameters for x-ray protein structure refinement. *Acta Crystallographica Section A: Foundations of Crystallography*, 47(4):392–400.

[46] Fang, C., Shang, Y., and Xu, D. (2018). Mufold-ss: New deep inception-inside-inception networks for protein secondary structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 86(5):592–598.

[47] Fatahalian, K., Sugerman, J., and Hanrahan, P. (2004). Understanding the efficiency of gpu algorithms for matrix-matrix multiplication. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 133–137. ACM.

[48] Fawcett, T. (2004). Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 31(1):1–38.

[49] Fernández-Recio, J., Totrov, M., and Abagyan, R. (2003). Icm-disco docking by global energy optimization with fully flexible side-chains. *Proteins: Structure, Function, and Bioinformatics*, 52(1):113–117.

[50] Flores, S. C. and Gerstein, M. B. (2007). Flexoracle: predicting flexible hinges by identification of stable domains. *BMC bioinformatics*, 8(1):215.

[51] Flores, S. C., Keating, K. S., Painter, J., Morcos, F., Nguyen, K., Merritt, E. A., Kuhn, L. A., and Gerstein, M. B. (2008). Hingemaster: normal mode hinge prediction approach and integration of complementary predictors. *Proteins: Structure, Function, and Bioinformatics*, 73(2):299–319.

[52] Flores, S. C., Lu, L. J., Yang, J., Carriero, N., and Gerstein, M. B. (2007). Hinge atlas: relating protein sequence to sites of structural flexibility. *BMC bioinformatics*, 8(1):167.

[53] Francisco, L. M., Sage, P. T., and Sharpe, A. H. (2010). The pd-1 pathway in tolerance and autoimmunity. *Immunological reviews*, 236(1):219–242.

[54] Frank, J. and Agrawal, R. K. (2000). A ratchet-like inter-subunit reorganization of the ribosome during translocation. *Nature*, 406(6793):318.

[55] Franklin, J., Koehl, P., Doniach, S., and Delarue, M. (2007). Minactionpath: maximum likelihood trajectory for large-scale structural transitions in a coarse-grained locally harmonic energy landscape. *Nucleic acids research*, 35(suppl_2):W477–W482.

[56] George, R. A. and Heringa, J. (2002). An analysis of protein domain linkers: their classification and role in protein folding. *Protein Engineering, Design and Selection*, 15(11):871–879.

[57] Gerhart, J. C. and Pardee, A. B. (1963). The effect of the feedback inhibitor, ctp, on subunit interactions in aspartate transcarbamylase. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 28, pages 491–496. Cold Spring Harbor Laboratory Press.

[58] Gerstein, M. and Krebs, W. (1998). A database of macromolecular motions. *Nucleic acids research*, 26(18):4280–4290.

[59] Gerstein, M., Lesk, A. M., and Chothia, C. (1994). Structural mechanisms for domain movements in proteins. *Biochemistry*, 33(22):6739–6749.

[60] Girdlestone, C. and Hayward, S. (2016). The dyndom3d webserver for the analysis of domain movements in multimeric proteins. *Journal of Computational Biology*, 23(1):21–26.

[61] Goh, C.-S., Milburn, D., and Gerstein, M. (2004). Conformational changes associated with protein–protein interactions. *Current opinion in structural biology*, 14(1):104–109.

[62] Gokhale, R. S. and Khosla, C. (2000). Role of linkers in communication between protein modules. *Current opinion in chemical biology*, 4(1):22–27.

[63] Golub, G. H. and Van der Vorst, H. A. (2000). Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1-2):35–65.

[64] Gouaux, J. E. and Lipscomb, W. N. (1990). Crystal structures of phosphonoacetamide ligated t and phosphonoacetamide and malonate ligated r states of aspartate carbamoyltransferase at 2.8-. ang. resolution and neutral ph. *Biochemistry*, 29(2):389–402.

[65] Grosso, J., Horak, C. E., Inzunza, D., Cardona, D. M., Simon, J. S., Gupta, A. K., Sankar, V., Park, J.-S., Kollia, G., Taube, J. M., et al. (2013). Association of tumor pd-l1 expression and immune biomarkers with clinical activity in patients (pts) with advanced solid tumors treated with nivolumab (anti-pd-1; bms-936558; ono-4538).

[66] Hanson, R. M., Prilusky, J., Renjian, Z., Nakane, T., and Sussman, J. L. (2013). Jsmol and the next-generation web-based representation of 3d molecular structure as applied to proteopedia. *Israel Journal of Chemistry*, 53(3-4):207–216.

[67] Harada, R. and Kitao, A. (2013). Parallel cascade selection molecular dynamics (pacsmd) to generate conformational transition pathway. *The Journal of chemical physics*, 139(3):07B611_1.

[68] Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

[69] Hayward, S. (1999). Structural principles governing domain motions in proteins. *Proteins: Structure, Function, and Bioinformatics*, 36(4):425–435.

[70] Hayward, S. and Berendsen, H. J. (1998). Systematic analysis of domain motions in proteins from conformational change: new results on citrate synthase and t4 lysozyme. *Proteins: Structure, Function, and Bioinformatics*, 30(2):144–154.

[71] Hayward, S. and Kitao, A. (2015). Monte carlo sampling with linear inverse kinematics for simulation of protein flexible regions. *Journal of chemical theory and computation*, 11(8):3895–3905.

[72] Hayward, S., Kitao, A., and Berendsen, H. J. (1997). Model-free methods of analyzing domain motions in proteins from simulation: a comparison of normal mode analysis and molecular dynamics simulation of lysozyme. *Proteins: Structure, Function, and Bioinformatics*, 27(3):425–437.

[73] Hayward, S. and Lee, R. A. (2002). Improvements in the analysis of domain motions in proteins from conformational change: Dyndom version 1.50. *Journal of Molecular Graphics and Modelling*, 21(3):181–183.

[74] Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919.

[75] Hinsen, K. (1998). Analysis of domain motions by approximate normal mode calculations. *Proteins: Structure, Function, and Bioinformatics*, 33(3):417–429.

[76] Hogues, H., Gaudreault, F., Corbeil, C. R., Deprez, C., Sulea, T., and Purisima, E. O. (2018). Propose: Direct exhaustive protein–protein docking with side chain flexibility. *Journal of chemical theory and computation*, 14(9):4938–4947.

[77] Huang, J. Z., Ng, M. K., Rong, H., and Li, Z. (2005). Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):657–668.

[78] Huang, S.-Y. (2015). Exploring the potential of global protein–protein docking: an overview and critical assessment of current programs for automatic ab initio docking. *Drug discovery today*, 20(8):969–977.

[79] Ingram, S., Munzner, T., and Olano, M. (2009). Glimmer: Multilevel mds on the gpu. *IEEE Transactions on Visualization and Computer Graphics*, 15(2):249–261.

[80] Ishida, Y., Agata, Y., Shibahara, K., and Honjo, T. (1992). Induced expression of pd-1, a novel member of the immunoglobulin gene superfamily, upon programmed cell death. *The EMBO journal*, 11(11):3887–3895.

[81] Ishima, R. and Torchia, D. A. (2000). Protein dynamics from nmr. *Nature Structural & Molecular Biology*, 7(9):740.

[82] Jacobs, D. J., Rader, A. J., Kuhn, L. A., and Thorpe, M. F. (2001). Protein flexibility predictions using graph theory. *Proteins: Structure, Function, and Bioinformatics*, 44(2):150–165.

[83] Jović, A., Brkić, K., and Bogunović, N. (2015). A review of feature selection methods with applications. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1200–1205. IEEE.

[84] Karaca, E., Melquiond, A. S., de Vries, S. J., Kastritis, P. L., and Bonvin, A. M. (2010). Building macromolecular assemblies by information-driven docking introducing the haddock multibody docking server. *Molecular & Cellular Proteomics*, 9(8):1784–1794.

[85] Karplus, M. and McCammon, J. A. (2002). Molecular dynamics simulations of biomolecules. *Nature structural biology*, 9(9):646–652.

[86] Khan, S. S. and Ahmad, A. (2004). Cluster center initialization algorithm for k-means clustering. *Pattern recognition letters*, 25(11):1293–1302.

[87] Kim, M. K., Jernigan, R. L., and Chirikjian, G. S. (2002). Efficient generation of feasible pathways for protein conformational transitions. *Biophysical journal*, 83(3):1620–1630.

[88] Kleywegt, G. J. and Jones, T. A. (1996). Phi/psi-chology: Ramachandran revisited. *Structure*, 4(12):1395–1400.

[89] Koike, R., Ota, M., and Kidera, A. (2014). Hierarchical description and extensive classification of protein structural changes by motion tree. *Journal of molecular biology*, 426(3):752–762.

[90] Kozakov, D., Hall, D. R., Xia, B., Porter, K. A., Padhorny, D., Yueh, C., Beglov, D., and Vajda, S. (2017). The cluspro web server for protein-protein docking. *Nature protocols*, 12(2):255.

[91] Krebs, W. G. and Gerstein, M. (2000). Survey and summary the morph server: a standardized system for analyzing and visualizing macromolecular motions in a database framework. *Nucleic acids research*, 28(8):1665–1675.

[92] Kuznetsov, I. B. (2008). Ordered conformational change in the protein backbone: Prediction of conformationally variable positions from sequence and low-resolution structural data. *Proteins: Structure, Function, and Bioinformatics*, 72(1):74–87.

[93] Kuznetsov, I. B. and McDuffie, M. (2008). Flexpred: a web-server for predicting residue positions involved in conformational switches in proteins. *Bioinformation*, 3(3):134.

[94] Latchman, Y., Wood, C. R., Chernova, T., Chaudhary, D., Borde, M., Chernova, I., Iwai, Y., Long, A. J., Brown, J. A., Nunes, R., et al. (2001). Pd-l2 is a second ligand for pd-1 and inhibits t cell activation. *Nature immunology*, 2(3):261.

[95] Lee, R. A., Razaz, M., and Hayward, S. (2003). The dyndom database of protein domain motions. *Bioinformatics*, 19(10):1290–1291.

[96] Lensink, M. F., Méndez, R., and Wodak, S. J. (2007). Docking and scoring protein complexes: Capri 3rd edition. *Proteins: Structure, Function, and Bioinformatics*, 69(4):704–718.

[97] Leslie, C., Eskin, E., and Noble, W. S. (2001). The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific.

[98] Li, W. and Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659.

[99] Lindahl, E., Azuara, C., Koehl, P., and Delarue, M. (2006). Nomad-ref: visualization, deformation and refinement of macromolecular structures based on all-atom normal mode analysis. *Nucleic Acids Research*, 34(suppl_2):W52–W56.

[100] Linding, R., Jensen, L. J., Diella, F., Bork, P., Gibson, T. J., and Russell, R. B. (2003). Protein disorder prediction: implications for structural proteomics. *Structure*, 11(11):1453–1459.

[101] Liu, H. and Motoda, H. (2007). *Computational methods of feature selection*. CRC Press.

[102] Loveland, A. B., Demo, G., Grigorieff, N., and Korostelev, A. A. (2017). Ensemble cryo-em elucidates the mechanism of translation fidelity. *Nature*, 546(7656):113.

[103] Lu, Y., Cohen, I., Zhou, X. S., and Tian, Q. (2007). Feature selection using principal feature analysis. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 301–304. ACM.

[104] MacArthur, M. W. and Thornton, J. M. (1991). Influence of proline residues on protein conformation. *Journal of molecular biology*, 218(2):397–412.

[105] Maiti, R., Van Domselaar, G. H., and Wishart, D. S. (2005). Moviemaker: a web server for rapid rendering of protein motions and interactions. *Nucleic acids research*, 33(suppl_2):W358–W362.

[106] Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.

[107] Mejías, C. and Guirola, O. (2019). Pharmacophore model of immunocheckpoint protein pd-l1 by cosolvent molecular dynamics simulations. *Journal of Molecular Graphics and Modelling*.

[108] Meng, F., Uversky, V. N., and Kurgan, L. (2017). Comprehensive review of methods for prediction of intrinsic disorder and its molecular functions. *Cellular and Molecular Life Sciences*, 74(17):3069–3090.

[109] Migden, M. R., Rischin, D., Schmults, C. D., Guminski, A., Hauschild, A., Lewis, K. D., Chung, C. H., Hernandez-Aya, L., Lim, A. M., Chang, A. L. S., et al. (2018). Pd-1 blockade with cemiplimab in advanced cutaneous squamous-cell carcinoma. *New England Journal of Medicine*, 379(4):341–351.

[110] Modha, D. S. and Spangler, W. S. (2003). Feature weighting in k-means clustering. *Machine learning*, 52(3):217–237.

[111] Moreno, B. H. and Ribas, A. (2015). Anti-programmed cell death protein-1/ligand-1 therapy in different cancers. *British journal of cancer*, 112(9):1421.

[112] Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C. (1995). Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540.

[113] Nachar, N. et al. (2008). The mann-whitney u: A test for assessing whether two independent samples come from the same distribution. *Tutorials in quantitative Methods for Psychology*, 4(1):13–20.

[114] Nakamoto, R. K., Ketchum, C. J., and Al-Shawi, M. K. (1999). Rotational coupling in the f0f1 atp synthase. *Annual review of biophysics and biomolecular structure*, 28(1):205–234.

[115] Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

[116] Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313.

[117] Nguyen, M. K., Jaillet, L., and Redon, S. (2017). As-rigid-as-possible molecular interpolation paths. *Journal of computer-aided molecular design*, 31(4):403–417.

[118] O'Donovan, C., Martin, M. J., Gattiker, A., Gasteiger, E., Bairoch, A., and Apweiler, R. (2002). High-quality protein knowledge resource: Swiss-prot and trembl. *Briefings in bioinformatics*, 3(3):275–284.

[119] Orengo, C. A., Michie, A. D., Jones, S., Jones, D. T., Swindells, M. B., and Thornton, J. M. (1997). Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109.

[120] Parkinson, G., Vojtechovsky, J., Clowney, L., Brünger, A. T., and Berman, H. (1996). New parameters for the refinement of nucleic acid-containing structures. *Acta Crystallographica Section D: Biological Crystallography*, 52(1):57–64.

[121] Pearl, F. M. G., Bennett, C. F., Bray, J. E., Harrison, A. P., Martin, N., Shepherd, A., Sillitoe, I., Thornton, J., and Orengo, C. A. (2003). The CATH database: an extended protein family resource for structural and functional genomics. *Nucleic Acids Research*, 31(1):452–455.

[122] Petsko, G. A. and Ringe, D. (2004). *Protein structure and function*. New Science Press.

[123] Pettersen, E. F., Goddard, T. D., Huang, C. C., Couch, G. S., Greenblatt, D. M., Meng, E. C., and Ferrin, T. E. (2004). Ucsf chimera—a visualization system for exploratory research and analysis. *Journal of computational chemistry*, 25(13):1605–1612.

[124] Pietal, M. J., Bujnicki, J. M., and Kozlowski, L. P. (2015). Gdfuzz3d: a method for protein 3d structure reconstruction from contact maps, based on a non-euclidean distance function. *Bioinformatics*, 31(21):3499–3505.

[125] Ponting, C. P. and Russell, R. R. (2002). The natural history of protein domains. *Annual review of biophysics and biomolecular structure*, 31(1):45–71.

[126] Poornam, G. P., Matsumoto, A., Ishida, H., and Hayward, S. (2009). A method for the analysis of domain movements in large biomolecular complexes. *Proteins: Structure, Function, and Bioinformatics*, 76(1):201–212.

[127] Porter, K. A., Desta, I., Kozakov, D., and Vajda, S. (2019). What method to use for protein–protein docking? *Current opinion in structural biology*, 55:1–7.

[128] Qi, G., Lee, R., and Hayward, S. (2005). A comprehensive and non-redundant database of protein domain movements. *Bioinformatics*, 21(12):2832–2838.

[129] Quarteroni, A., Sacco, R., and Saleri, F. (2010). *Numerical mathematics*, volume 37. Springer Science & Business Media.

[130] Remington, S., Wiegand, G., and Huber, R. (1982). Crystallographic refinement and atomic models of two different forms of citrate synthase at 2· 7 and 1· 7 å resolution. *Journal of molecular biology*, 158(1):111–152.

[131] Remy, I., Wilson, I. A., and Michnick, S. W. (1999). Erythropoietin receptor activation by a ligand-induced conformation change. *Science*, 283(5404):990–993.

[132] Ritchie, D. W. (2008). Recent progress and future directions in protein-protein docking. *Current protein and peptide science*, 9(1):1–15.

[133] Rost, B. (1999). Twilight zone of protein sequence alignments. *Protein Engineering, Design and Selection*, 12(2):85–94.

[134] Rost, B. and Sander, C. (1993). Prediction of protein secondary structure at better than 70% accuracy. *Journal of molecular biology*, 232(2):584–599.

[135] Roux, M. (2018). A comparative study of divisive and agglomerative hierarchical clustering algorithms. *Journal of Classification*, 35(2):345–366.

[136] Sambongi, Y., Iko, Y., Tanabe, M., Omote, H., Iwamoto-Kihara, A., Ueda, I., Yanagida, T., Wada, Y., and Futai, M. (1999). Mechanical rotation of the c subunit oligomer in atp synthase (f0f1): direct observation. *Science*, 286(5445):1722–1724.

[137] Samish, I., Gu, J., and Klein, M. L. (2009). Protein motion: Simulation. *Structural Bioinformatics*, 2:907–936.

[138] Sander, C. and Schneider, R. (1991). Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins: Structure, Function, and Bioinformatics*, 9(1):56–68.

[139] Savaresi, S. M., Boley, D. L., Bittanti, S., and Gazzaniga, G. (2002). Cluster selection in divisive clustering algorithms. In *Proceedings of the 2002 SIAM International Conference on Data Mining*, pages 299–314. SIAM.

[140] Sayle, R. A. and Milner-White, E. J. (1995). Rasmol: biomolecular graphics for all. *Trends in biochemical sciences*, 20(9):374–376.

[141] Schrauber, H., Eisenhaber, F., and Argos, P. (1993). Rotamers: to be or not to be?: an analysis of amino acid side-chain conformations in globular proteins. *Journal of molecular biology*, 230(2):592–612.

[142] Schueler-Furman, O., Wang, C., and Baker, D. (2005). Progress in protein–protein docking: Atomic resolution predictions in the capri experiment using rosettadock with an improved treatment of side-chain flexibility. *Proteins: Structure, Function, and Bioinformatics*, 60(2):187–194.

[143] Secundo, F. (2013). Conformational changes of enzymes upon immobilisation. *Chemical Society Reviews*, 42(15):6250–6261.

[144] Shatsky, M., Nussinov, R., and Wolfson, H. J. (2002). Flexible protein alignment and hinge detection. *Proteins: Structure, Function, and Bioinformatics*, 48(2):242–256.

[145] Shatsky, M., Nussinov, R., and Wolfson, H. J. (2004). Flexprot: alignment of flexible protein structures without a predefinition of hinge regions. *Journal of Computational Biology*, 11(1):83–106.

[146] Shi, D., Zhou, S., Liu, X., Zhao, C., Liu, H., and Yao, X. (2018). Understanding the structural and energetic basis of pd-1 and monoclonal antibodies bound to pd-l1: A molecular modeling perspective. *Biochimica et Biophysica Acta (BBA)-General Subjects*, 1862(3):576–588.

[147] Shores, T. S. (2007). *Applied linear algebra and matrix analysis*, volume 2541. Springer.

[148] Sillitoe, I., Dawson, N., Thornton, J., and Orengo, C. (2015). The history of the cath structural classification of protein domains. *Biochimie*, 119:209–217.

[149] Skalniak, L., Zak, K. M., Guzik, K., Magiera, K., Musielak, B., Pachota, M., Szelazek, B., Kocik, J., Grudnik, P., Tomala, M., et al. (2017). Small-molecule inhibitors of pd-1/pd-l1 immune checkpoint alleviate the pd-l1-induced exhaustion of t-cells. *Oncotarget*, 8(42):72167.

[150] Smith, T. F., Waterman, M. S., et al. (1981). Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197.

[151] Smyth, M. and Martin, J. (2000). x ray crystallography. *Molecular Pathology*, 53(1):8.

[152] Sonnhammer, E. L., Eddy, S. R., and Durbin, R. (1997). Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins: Structure, Function, and Bioinformatics*, 28(3):405–420.

[153] Sun, X. and Xu, W. (2014). Fast implementation of delong's algorithm for comparing the areas under correlated receiver operating characteristic curves. *IEEE Signal Processing Letters*, 21(11):1389–1393.

[154] Taylor, D., Cawley, G., and Hayward, S. (2014). Quantitative method for the assignment of hinge and shear mechanism in protein domain movements. *Bioinformatics*, 30(22):3189–3196.

[155] Toyoshima, C., Nakasako, M., Nomura, H., and Ogawa, H. (2000). Crystal structure of the calcium pump of sarcoplasmic reticulum at 2.6 å resolution. *Nature*, 405(6787):647.

[156] Trefethen, L. N. and Bau III, D. (1997). *Numerical linear algebra*, volume 50. Siam.

[157] Tuckerman, M. E. (2002). Ab initio molecular dynamics: basic concepts, current trends and novel applications. *Journal of Physics: Condensed Matter*, 14(50):R1297.

[158] Van Zundert, G., Rodrigues, J., Trellet, M., Schmitz, C., Kastritis, P., Karaca, E., Melquiond, A., van Dijk, M., De Vries, S., and Bonvin, A. (2016). The haddock2. 2 web server: user-friendly integrative modeling of biomolecular complexes. *Journal of molecular biology*, 428(4):720–725.

[159] Veevers, R., Cawley, G., and Hayward, S. (2020). Investigation of sequence features of hinge-bending regions in proteins with domain movements using kernel logistic regression. *BMC bioinformatics*, 21:1–18.

[160] Veevers, R. and Hayward, S. (2018). Morphing and docking visualisation of biomolecular structures using multi-dimensional scaling. *Journal of Molecular Graphics and Modelling*, 82:108–116.

[161] Veevers, R. and Hayward, S. (2019). Methodological improvements for the analysis of domain movements in large biomolecular complexes. *Biophysics and Physicobiology*.

[162] Viricel, C., Ahmed, M., and Barakat, K. (2015). Human pd-1 binds differently to its human ligands: a comprehensive modeling study. *Journal of Molecular Graphics and Modelling*, 57:131–142.

[163] Vonrhein, C., Schlauderer, G. J., and Schulz, G. E. (1995). Movie of the structural changes during a catalytic cycle of nucleoside monophosphate kinases. *Structure*, 3(5):483–490.

[164] Vreven, T., Moal, I. H., Vangone, A., Pierce, B. G., Kastritis, P. L., Torchala, M., Chaleil, R., Jiménez-García, B., Bates, P. A., Fernandez-Recio, J., et al. (2015). Updates to the integrated protein–protein interaction benchmarks: docking benchmark version 5 and affinity benchmark version 2. *Journal of molecular biology*, 427(19):3031–3041.

[165] Wang, S., Peng, J., Ma, J., and Xu, J. (2016). Protein secondary structure prediction using deep convolutional neural fields. *Scientific reports*, 6:18962.

[166] Weidmann, J. (2012). *Linear operators in Hilbert spaces*, volume 68. Springer Science & Business Media.

[167] Weiss, D. R. and Koehl, P. (2014). Morphing methods to visualize coarse-grained protein dynamics. *Protein Dynamics: Methods and Protocols*, pages 271–282.

[168] Weiss, D. R. and Levitt, M. (2009). Can morphing methods predict intermediate structures? *Journal of molecular biology*, 385(2):665–674.

[169] Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer.

[170] Woolfson, M. (2018). The development of structural x-ray crystallography. *Physica Scripta*, 93(3):032501.

[171] Yang, J. and Hu, L. (2019). Immunomodulators targeting the pd-1/pd-l1 protein-protein interaction: From antibodies to small molecules. *Medicinal research reviews*, 39(1):265–301.

[172] Yaseen, A. and Li, Y. (2014). Context-based features enhance protein secondary structure prediction accuracy. *Journal of chemical information and modeling*, 54(3):992–1002.

[173] Ye, Y. and Godzik, A. (2004). Fatcat: a web server for flexible structure comparison and structure similarity searching. *Nucleic acids research*, 32(suppl_2):W582–W585.

[174] Zhang, N., Tu, J., Wang, X., and Chu, Q. (2019). Programmed cell death-1/programmed cell death ligand-1 checkpoint inhibitors: differences in mechanism of action. *Immunotherapy*, 11(5):429–441.

[175] Zhu, J. and Hastie, T. (2005). Kernel logistic regression and the import vector machine. In *Advances in neural information processing systems*, pages 1081–1088.

Appendices

Appendix A

PDB Codes for MDS Morph Test Cases

PDB Code 1	Chain ID 1	PDB Code 2	Chain ID 2
2a4n	B	1b87	A
1i9b	A	1i9b	B
1cjk	A	2gvd	A
1xjb	A	1j96	A
1b8g	B	1ynu	A
1i8m	A	1kb5	L
2b48	A	2p1l	A
1at1	A	1raa	A
1wp9	E	1wp9	C
4tnw	A	4tnv	A
1g3j	A	2bct	
1ux5	A	1ux4	A
1k8t	A	1xfx	D
4jz7	C	4jz9	C
2gjz	B	2gk0	H
1yy9	C	1yy8	A
1zu0	A	1zty	A
2i7v	A	2i7t	A
1rid	A	1y8e	B
1ghq	C	1ly2	A
1tij	A	1r4c	G
1k1q	A	1k1q	B
2gd1	R	1nq5	A
1y1v	I	1r9s	I
1mmi	A	1jql	A
2iuu	F	2iut	A

PDB Code 1	Chain ID 1	PDB Code 2	Chain ID 2
2a1t	R	1efv	A
1k3x	A	1q3b	A
2b7m	A	2b1e	A
1yuh	A	1sm3	L
1ezf	B	1ezf	A
1r19	D	1r17	A
1jbw	A	2gc5	A
2i02	A	2i02	B
2j6h	A	1jxa	C
1xw5	B	1hnb	A
1axk	B	1axk	A
3nyn	A	2acx	A
1qcf	A	2c0t	B
2pjr	A	1pjr	
1gp9	C	1gmo	C
1a9e	A	2bvp	A
3loc	A	4jyk	A
1ptm	B	1r8k	A
1vdw	A	1vdw	B
1jt0	A	1jum	A
2gfb	J	1kno	D
1za6	B	1za6	D
1mju	L	1uz8	A
1i40	A	1i6t	A
1ic1	B	1ic1	A
1t09	A	1t0l	D
1zi7	A	1zhx	A
1y1x	A	1y1x	B
8adh		1a71	A
1kwh	A	1j1n	B
2obg	A	1peb	A
1ewk	B	1ewk	A
2gid	K	2gia	B
1nc2	D	1nc2	B
2azo	A	2azo	B
2nmt	A	1iic	A
2f8v	C	2a38	C
1df1	A	1df1	B
1ho5	A	1oid	A
1ttt	A	1ob5	A
1ynp	A	1ynp	B
1mwr	B	1vqq	B

PDB Code 1	Chain ID 1	PDB Code 2	Chain ID 2
2pah	A	2pah	B
1quk		1oib	A
1au7	A	1au7	B
2oqg	B	2oqg	C
2ff4	B	2ff4	A
1ks2	A	1ks2	B
1y0z	B	1y0z	A
4hfi	A	4hfi	B
1t33	A	1t33	B
2g50	E	1aqf	H
2gil	B	2ffq	A
3g56	A	3frq	A
1x86	A	1txd	A
2h23	A	2h21	B
2c9o	A	2c9o	B
1hf2	C	1hf2	A
1o9x	A	1e7e	A
1kyq	B	1kyq	C
2qkm	B	2qkm	D
1o9l	D	1o9l	A
1e4w	H	1e4x	I
1kei	A	1kkk	A
1qxx	A	1u07	A
1rfy	A	1rfy	B
3ice	C	1pv4	F
1aro	P	1h38	A
2c5j	A	2c5k	T
2ag6	A	1u7d	B
1gqq	A	1p3d	A
1ufq	C	1uei	B
1rke	A	1rkc	A
1o89	A	1o8c	B

Appendix B

Protein Sequences Used in KLR Test and Training Data

B.1 Group 1

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Dna-Directed RNA Polymerase II Largest Subunit	1i50	A	2nvq	A
Damage-Specific DNA Binding Protein 1	2b5m	A	2hye	A
Pullulanase	2yoc	A	2yoc	B
Glucansucrase	3klk	A	4amc	A
Ubiquitin-Activating Enzyme E1 1	4ii3	A	4ii2	A
Leucyl-Trna Synthetase	1wz2	B	1wkb	A
DNA Polymerase	1ig9	A	2dy4	C
Ns5 Polymerase	4k6m	A	5ccv	A
Pyruvate,orthophosphate Dikinase	1vbh	A	2r82	A
T7 Lysozyme	1h38	A	1s77	D
Importin Beta-1 Subunit	2bku	B	2bpt	A
Protein Translocase Subunit Seca	3jv2	A	1m6n	A
Isocitrate Dehydrogenase [Nadp]	3mbc	A	1j1w	A
Programmed Cell Death 6-Interacting Protein	2oev	A	4jjy	A
Lactoferrin	1bka	A	1cb6	A
Argonaute	3hk2	A	3f73	A
Atp-Dependent DNA Helicase Rep	1uaa	A	1uaa	B

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Polyphosphate Kinase	2o8r	A	2o8r	B
Calpain 2, Large [Catalytic] Subunit Precursor	1u5i	A	2ary	A
Secreted Effector Protein	2qyu	A	2qza	A
Glucosamine-Fructose-6-Phosphate Amino-transferase	2j6h	A	1jxa	C
Protein Phosphatase Pp2a	2nym	D	2ie4	A
Flavocytochrome C Fumarate Reductase	1d4e	A	1qo8	A
Phosphoenolpyruvate-Protein Phosphotrans-ferase	2hwg	A	1zym	A
Acyl-Coenzyme A Synthetase Acsm2a, Mito-chondrial Precursor	3b7w	A	3c5e	A
Long Chain Fatty Acid-Coa Ligase	1ult	A	1ult	B
Chitin Oligosaccharide Binding Protein	1zu0	A	1zty	A
D-3-Phosphoglycerate Dehydrogenase	1ygy	B	1ygy	A
5”-Nucleotidase	1hp1	A	1hpu	C
Groel	1aon	H	2c7e	A
5”-Nucleotidase	4h2g	A	4h2i	A
Diphtheria Toxin	1f0l	B	1tox	B
Luciferase	1lci	A	2d1r	A
Pyruvate Kinase Isozymes M1/m2	4fxj	A	3srh	A
Periplasmic Oligopeptide-Binding Protein	1rkm	A	2rkm	A
Lethal Factor	1yqy	A	1jky	A
Nitrite Reductase	1hzv	A	1nir	B
Metabotropic Glutamate Receptor Subtype 1	2e4u	A	3sm9	A
2,3-Bisphosphoglycerate-Independent Phospho-glycerate Mutase	1o98	A	2ify	A
Ba3-Type Cytochrome-C Oxidase	2ify	A	4my4	A
Dipeptide-Binding Protein	1dpe	A	1dpp	A
4-Chlorobenzoyl Coa Ligase	3cw8	X	3cw9	A
Pyruvate Kinase	1pkl	A	3hqp	A
Nickel-Binding Periplasmic Protein	2noo	A	1zlq	B
Macromolecule-Binding Periplasmic Protein	1kwh	A	1j1n	B
Algq1	1y3q	A	1y3n	A
Fimbrin-Like Protein	1pxy	B	1pxy	A
Chaperone Protein Htpg	1y4s	B	2iop	A
Malonyl Coa Synthetase	4fut	A	4fuq	A
Intermedilysin	1s3r	A	4bik	A
2-Succinylbenzoate–Coa Ligase	5buq	B	5bur	A
Atp Synthase Beta Chain, Mitochondrial	2hld	N	2hld	M
F1-Atpase	1sky	E	1bmf	F
Son Of Sevenless Protein Homolog 1	1xd2	C	2ii0	A

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Atp Synthase Subunit Beta	2jdi	E	1h8e	D
Metabotropic Glutamate Receptor Subtype 1	1ewk	B	1ewk	A
Udp-N-Acetylmuramate-L-Alanine Ligase	1p3d	A	1gqq	B
3-Phosphoshikimate 1-Carboxyvinyltransferase	2gg4	A	2gg6	A
Integrin Beta-3	1tye	B	1jv2	B
3-Phosphoshikimate 1-Carboxyvinyltransferase	3roi	A	3slh	A
Protein (Udp-N-Acetylmuramoyl-L-Alanine:d-Glutamate Ligase)	3uag	A	1e0d	A
5-Enolpyruvylshikimate-3-Phosphate Synthase	1rf6	C	1rf5	B
Type I Restriction-Modification Enzyme, S Subunit	1yf2	B	1yf2	A
Elongation Factor 1-Alpha	1jny	A	1f60	A
Isocitrate Dehydrogenase	1sjs	A	1hj6	A
3-Phosphoglycerate Kinase	13pk	A	1php	A
Isocitrate Dehydrogenase [Nadp] Cytoplasmic	1t09	A	1t0l	D
Isocitrate Dehydrogenase	1lwd	A	3mas	A
Putative Fimbrial Subunit	4hss	A	4hss	B
Gamma-Aminobutyric Acid Type B Receptor Subunit 1	4mqe	A	4ms3	A
Heat Shock Locus U	1do0	A	1do0	B
D-3-Phosphoglycerate Dehydrogenase (Phosphoglycerate 3 Dehydrogenase) (E.C.1.1.1.95)	1psd	A	1sc6	B
Molybdopterin Biosynthesis Protein Moea	2nqq	C	2nqq	A
47 Kda Membrane Antigen	1o75	B	1o75	A
Elongation Factor Tu	1ob5	C	2c78	A
Protein (Eukaryotic Peptide Chain Release Factor Subunit 1)	1dt9	A	3e1y	A
Aminopeptidase T	2ayi	D	2ayi	B
Succinyl-Coa Synthetase, Beta Chain	1eud	B	2fp4	B
Acarbose/maltose Binding Protein Gach	3k01	A	3jzj	A
Polymerase (Dna Directed) Kappa	1t94	B	2oh2	B
Molybdopterin Biosynthesis Moea Protein	1wu2	A	1wu2	B
Udp-N-Acetylglucosamine 2-Epimerase	3beo	A	1o6c	A
Actin	2zwh	A	1j6z	A
Maltose Abc Transporter, Periplasmic Maltose-Binding Protein	2gha	A	2ghb	B
D-Maltodextrin Binding Protein	1anf	A	1jw5	A
Chemotaxis Protein Chea	1b3q	A	2ch4	A
Collybistin II	2dfk	A	2dfk	C
Rho Guanine Nucleotide Exchange Factor 12	1x86	A	1txd	A
Uncharacterized Protein	4xe8	A	4xe7	A

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Udp-N-Acetylglucosamine 2-Epimerase	4neq	A	4nes	A
3-Isopropylmalate Dehydrogenase	2y3z	A	4f7i	A
Choline Kinase Alpha	2ckq	B	2i7q	A
Dbh Protein	1k1q	A	2rdi	A
Leu/ile/val-Binding Protein	1z15	A	1z16	A
Nagk Protein	2ch6	D	2ch6	B
DNA Polymerase IV	3qz7	A	3bq1	A
Purine Nucleotide Synthesis Repressor	1jft	A	1dbq	A
Atp-Dependent Hsl Protease Atp-Binding Subunit Hslu	1im2	A	1qg4	A
Twitching Motility Protein Pilt	2gsz	A	2gsz	E
DNA Polymerase III, Delta Subunit	1jqj	D	1xxh	F
D-Lactate Dehydrogenase	1j49	B	1j4a	D
Ovotransferrin	1tfa	A	1iej	A
Rfcs	1iqp	C	1iqp	D
DNA Polymerase Beta	1bpd	A	2bpg	B
Igg Heavy Chain	1za6	B	1za6	D
Serotransferrin	1ryo	A	1bp5	C
Atp-Dependent Clp Protease Atp-Binding Subunit Clpx	3hws	A	3hws	B
Parm	1mwk	A	1mwm	A
M-Calpain	1kxr	A	1ziv	A
Thioredoxin Reductase	1tde	A	1f6m	A
Translation Initiation Factor Eif-2b, Delta Subunit	3a9c	A	3vm6	A
Xylanase J	2dck	A	2dcj	A
Calpain 9	1ziv	A	2p0r	A
Ig Epsilon Chain C Region	4j4p	B	1o0v	B
Spectrin Alpha Chain, Brain	1u4q	B	1cun	B
Sugar Transport Protein	1tjy	A	1tm2	A
Nuclear Factor Nf-Kappa-B P105 Subunit	1ooa	A	2i9t	B
Manganese-Dependent Inorganic Pyrophosphatase	1k20	B	1k23	B
Interleukin-1 Receptor	1itb	B	1g0y	R
Manganese-Dependent Inorganic Pyrophosphatase	1k23	A	1wpm	A
Guanine Nucleotide Exchange Factor Dbs [Fragment]	1rj2	G	1rj2	J
D-Galactose-Binding Periplasmic Protein	2fw0	A	2hph	A
Sugar Abc Transporter, Periplasmic Sugar-Binding Protein	3c6q	A	3c6q	C

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
D-3-Phosphoglycerate Dehydrogenase, Putative	4nfy	A	4njm	A
Nf-Kappa-B P65	1nfi	C	2ram	B
Mrna Decapping Enzyme	1xmm	B	1xml	B
Probable Transcriptional Regulator	2esn	A	2esn	C
Virb11 Homolog	1nlz	F	1nlz	E
Hexokinase	2e2n	A	2e2o	A
Type IIe Restriction Endonuclease Naei	1ev7	A	1iaw	A
2-Dehydropantoate 2-Reductase	1ks9	A	2ofp	B
Ribose Abc Transporter, Periplasmic Ribose-Binding Protein	2fn9	A	2fn8	A
Titin	2ill	A	2nzi	B
Serine/threonine-Protein Kinase Pak 4	2cdz	A	2c30	A
D-Allose-Binding Periplasmic Protein	1gub	A	1rpj	A
Glutamate [Nmda] Receptor Subunit Zeta 1	1y20	A	1pbq	A
Nuclear Factor Of Activated T-Cells, Cytoplasmic 2	1owr	Q	1owr	M
Potassium Channel	2wln	A	2wlk	A
Pantothenate Synthetase	3ag5	A	3ag6	A
Diaminopimelate Epimerase	2q9h	A	2gke	A
Mhc Class I H-2dd Heavy Chain	1qo3	A	1ddh	A
Glutamate Receptor Ionotropic, Nmda 2A	4nf5	B	3oel	A
D-Ribose-Binding Protein	1ba2	A	1urp	C
Osmoprotection Protein (Prox)	1sw4	A	1sw5	C
Udp-2,3-Diacylglucosamine Pyrophosphatase Lpxi	4ggm	X	4j6e	A
N-Methyl-D-Aspartate Receptor Subunit 1	1pbq	B	1pb7	A
Pectocin M2	4n58	A	4n59	A
Spac19a8.12 Protein	2qkm	B	2qkm	D
Phosphate-Binding Protein Psts 1	4exl	A	4lat	A
Probable Translation Initiation Factor 2 Alpha Subunit	1yz6	A	1yz7	A
Glutamate Receptor Subunit 2	1ftj	B	1fto	A
Glutamate Receptor 3	3dln	A	1fto	A
Acetylglutamate Kinase	2wxb	A	1gs5	A
Glutamate Receptor, Ionotropic Kainate 1	2f34	B	1ycj	B
Betaine Abc Transporter Permease And Substrate Binding Protein	3l6g	A	3l6h	A
Glutamate Receptor, Ionotropic Kainate 1	1s7y	B	1fto	A
Vinculin Isoform Vcl	1ydi	A	1rke	A
Glutamate Receptor, Ionotropic Kainate 3	4e0w	A	1fto	A
Nopaline-Binding Periplasmic Protein	4pow	A	4p0i	A

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Angiostatin	1ki0	A	2doh	X
Endonuclease VIII	1k3x	A	1q3b	A
Major Surface Antigen P30	1kzq	A	1ynt	G
Alpha-1 Catenin	1h6g	A	1l7c	C
Mg2+ Transporter Mgte	2yvy	A	2yvz	A
Tight Junction Protein Zo-1	3lh5	A	3kfv	A
Replication Protein A 70 Kda Dna-Binding Subunit	1fgu	B	1fgu	A
Dihydrodipicolinate Reductase	1yl7	C	1p9l	A
Hypothetical Protein	2i76	A	2i76	B
Dihydrodipicolinate Reductase	3qy9	B	3qy9	D
Putative Abc Transporter, Periplasmic Binding Protein, Amino Acid	2yln	A	3zsf	A
Lysine, Arginine, Ornithine-Binding Protein	2lao	A	1lst	A
Glutamate Receptor Delta-2 Subunit	2v3t	A	2v3u	A
Windbeutel Protein	2c1y	A	2c0e	A
Dna-Directed RNA Polymerase Alpha Chain	1ynj	A	1ynn	B
Dna-Directed RNA Polymerase Alpha Chain	2a6h	A	1iw7	L
Fab 17B Heavy Chain	1rz8	B	2i60	R
Igg1 Antibody 58.2 (Heavy Chain)	1f58	H	3f58	H
Abc-Type Transporter, Periplasmic Subunit Family 3	4psh	A	4prs	A
Fab-Ysd1 Heavy Chain	1za3	H	1za3	B
Glutamine Binding Protein	1wdn	A	1ggg	B
Hyb3 Heavy Chain	1w72	H	1dfb	H
Calcium-Gated Potassium Channel Mthk	2fy8	C	2fy8	H
Chimera Of Fab2c4: "Humanized" Murine Monoclonal Antibody	1l7i	H	1s78	F
Igg1 Fab Fragment	1igc	H	2aab	H
Catalytic Elimination Antibody 13G5 Heavy Chain	2gjz	B	2gk0	H
Chimeric Germline Precursor Of Oxy-Cope Catalytic Antibody Az-28 (Heavy Chain)	1d5i	H	1d5b	B
Fab Fragment, Heavy Chain	2h2s	E	2htl	C
Igg1 Fab Fragment (Hc19)	1gig	H	2vir	B
Antibody Light Chain	1jgu	L	1baf	L
Fab Fragment, Antibody A5b7	1ad0	B	1rmf	H
Igg Heavy Chain	2dd8	H	1rzi	F
Fab Fragment Of 8F5 Antibody Against Human Rhinovirus 3 Serotype 2 4	1bbd	L	1hin	L
K42-41l Fab Light Chain	1mju	L	1uz8	A

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
28B4 Fab	1kem	H	1q9l	B
Pc283 Immunoglobulin	1kcr	H	1kcu	H
Anti-Idiotypic Monoclonal Antibody (Light Chain)	2aab	L	1iqw	L
Igg 5C8	15c8	H	1fgn	H
Antibody M41	1gpo	L	1keg	L
Fab Heavy Chain	1xf3	H	1i8m	B
Immunoglobulin Gamma-1 Heavy Chain Constant Region	1e4k	B	2iwg	A
Igg1 Antibody 58.2 (Light Chain)	1f58	L	3f58	L
Immunoglobulin G1 (Igg1)	2mcg	1	2mcg	2
Loc - Lambda 1 Type Light-Chain Dimer	3bjl	B	1bjm	A
Mature Metal Chelatase Catalytic Antibody, Heavy Chain	1ngy	B	1n7m	L
Dihydrodipicolinate Reductase	1vm6	C	1vm6	B
Igg2a Fab Fragment (50.1)	1ggi	M	1ai1	L
Igm-Kappa Cold Agglutinin (Light Chain)	1dn0	C	1rhh	A
Immunoglobulin Lambda Light Chain	1jvk	B	1jvk	A
Monoclonal Antibody 2D12.5, Igg1 Gamma Heavy Chain	1gig	L	1q0x	L
Antibody Light Chain 11K2	2bdn	L	1osp	L
Immunoglobulin 48G7 Germline Fab	1gaf	L	1gpo	L
Fab 17B Light Chain	1rz8	A	2ny1	C
Fab Fragment Of Murine Monoclonal Antibody An02 Complex 3 With Its Hapten (2,2,6,6-Tetramethyl-1-Piperidinyloxy- 4 Dinitrophenyl)	1baf	L	1cz8	L
33H1 Fab Light Chain	1ors	A	1fig	L
Humanized Antibody Hfe7a, Light Chain	1it9	L	2gcy	A
Igg Antibody (Light Chain)	1emt	L	2a6i	A
Igg2b (Kappa)	1cgs	H	2cgr	H
Monoclonal Anti-Estradiol 10G6d6 Immunoglobulin Gamma-1 Chain	1jn6	B	1jnh	B
Pc287 Immunoglobulin	1kcu	L	1fsk	K
Antibody Light Chain Fab	1i8m	A	1qbm	L
Erythropoietin Receptor	1eer	B	1ern	B
17E8	1eap	A	1a0q	L
Fab E51 Light Chain	1rzf	L	1q1j	M
Fab Fragment, Antibody A5b7	1ad0	A	1rmf	L
Germline Metal Chelatase Catalytic Antibody, Chain H	1n7m	H	1ngy	A
Humanized Antibody D3h44	1pg7	H	1jps	H

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Immunoglobulin	1ce1	L	1t04	C
Lambda III Bence Jones Protein Cle	1lil	A	1lil	B
Chimeric Germline Precursor Of Oxy-Cope Catalytic Antibody Az-28 (Light Chain)	1d6v	L	1axs	L
Hyb3 Light Chain	1w72	M	1adq	L
Fibroblast Growth Factor Receptor 2	1e0o	D	1djs	A
Septum Site-Determining Protein Minc	1hf2	C	1hf2	A
Vascular Cell Adhesion Molecule-1	1vsc	B	1vca	A
P58-Cl42 Kir	1nkr	A	2dli	A
N2b-Titin Isoform	2f8v	C	2a38	C
Transcriptional Regulator, Tetr Family	1zkg	A	1z77	A
Hypothetical Transcriptional Regulator In Qaca 5”Region	1jt0	A	1jtx	A
Muscle-Specific Kinase Receptor	2iep	B	2iep	A
Transcriptional Regulator	3vok	A	3vp5	A
Tenascin	1qr4	B	1qr4	A
Fatty Acid-Binding Protein, Epidermal	4azr	B	1b56	A

B.2 Group 2

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Dna-Directed RNA Polymerase II Largest Subunit	1i50	A	2nvq	A
Damage-Specific DNA Binding Protein 1	2b5m	A	2hye	A
Pullulanase	2yoc	A	2yoc	B
Glucansucrase	3klk	A	4amc	A
Ubiquitin-Activating Enzyme E1 1	4ii3	A	4ii2	A
Leucyl-Trna Synthetase	1wz2	B	1wkb	A
DNA Polymerase	1ig9	A	2dy4	C
Ns5 Polymerase	4k6m	A	5ccv	A
Pyruvate,orthophosphate Dikinase	1vbh	A	2r82	A
T7 Lysozyme	1h38	A	1s77	D
Importin Beta-1 Subunit	2bku	B	2bpt	A
Protein Translocase Subunit Seca	3jv2	A	1m6n	A
Isocitrate Dehydrogenase [Nadp]	3mbc	A	1j1w	A
Programmed Cell Death 6-Interacting Protein	2oev	A	4jjy	A
Lactoferrin	1bka	A	1cb6	A
Argonaute	3hk2	A	3f73	A
Atp-Dependent DNA Helicase Rep	1uaa	A	1uaa	B
Polyphosphate Kinase	2o8r	A	2o8r	B
Calpain 2, Large [Catalytic] Subunit Precursor	1u5i	A	2ary	A
Secreted Effector Protein	2qyu	A	2qza	A
Glucosamine-Fructose-6-Phosphate Aminotransferase	2j6h	A	1jxa	C
Protein Phosphatase Pp2a	2nym	D	2ie4	A
Flavocytochrome C Fumarate Reductase	1d4e	A	1qo8	A
Phosphoenolpyruvate-Protein Phosphotransferase	2hwg	A	1zym	A
Acyl-Coenzyme A Synthetase Acsm2a, Mitochondrial Precursor	3b7w	A	3c5e	A
Long Chain Fatty Acid-Coa Ligase	1ult	A	1ult	B
Chitin Oligosaccharide Binding Protein	1zu0	A	1zty	A
D-3-Phosphoglycerate Dehydrogenase	1ygy	B	1ygy	A
5”-Nucleotidase	1hp1	A	1hpu	C
Groel	1aon	H	2c7e	A
5”-Nucleotidase	4h2g	A	4h2i	A
Diphtheria Toxin	1f0l	B	1tox	B
Luciferase	1lci	A	2d1r	A
Pyruvate Kinase Isozymes M1/m2	4fxj	A	3srh	A

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Periplasmic Oligopeptide-Binding Protein	1rkm	A	2rkm	A
Lethal Factor	1yqy	A	1jky	A
Nitrite Reductase	1hzv	A	1nir	B
Metabotropic Glutamate Receptor Subtype 1	2e4u	A	3sm9	A
2,3-Bisphosphoglycerate-Independent Phosphoglycerate Mutase	1o98	A	2ify	A
Ba3-Type Cytochrome-C Oxidase	2ify	A	4my4	A
Dipeptide-Binding Protein	1dpe	A	1dpp	A
4-Chlorobenzoyl Coa Ligase	3cw8	X	3cw9	A
Pyruvate Kinase	1pkl	A	3hqp	A
Nickel-Binding Periplasmic Protein	2noo	A	1zlq	B
Macromolecule-Binding Periplasmic Protein	1kwh	A	1j1n	B
Algq1	1y3q	A	1y3n	A
Fimbrin-Like Protein	1pxy	B	1pxy	A
Chaperone Protein Htpg	1y4s	B	2iop	A
Malonyl Coa Synthetase	4fut	A	4fuq	A
Intermedilysin	1s3r	A	4bik	A
2-Succinylbenzoate–Coa Ligase	5buq	B	5bur	A
Atp Synthase Beta Chain, Mitochondrial	2hld	N	2hld	M
F1-Atpase	1sky	E	1bmf	F
Son Of Sevenless Protein Homolog 1	1xd2	C	2ii0	A
Atp Synthase Subunit Beta	2jdi	E	1h8e	D
Metabotropic Glutamate Receptor Subtype 1	1ewk	B	1ewk	A
Udp-N-Acetylmuramate-L-Alanine Ligase	1p3d	A	1gqq	B
3-Phosphoshikimate 1-Carboxyvinyltransferase	2gg4	A	2gg6	A
Integrin Beta-3	1tye	B	1jv2	B
3-Phosphoshikimate 1-Carboxyvinyltransferase	3roi	A	3slh	A
Protein (Udp-N-Acetylmuramoyl-L-Alanine:d-Glutamate Ligase)	3uag	A	1e0d	A
5-Enolpyruvylshikimate-3-Phosphate Synthase	1rf6	C	1rf5	B
Type I Restriction-Modification Enzyme, S Subunit	1yf2	B	1yf2	A
Elongation Factor 1-Alpha	1jny	A	1f60	A
Isocitrate Dehydrogenase	1sjs	A	1hj6	A
3-Phosphoglycerate Kinase	13pk	A	1php	A
Isocitrate Dehydrogenase [Nadp] Cytoplasmic	1t09	A	1t0l	D
Isocitrate Dehydrogenase	1lwd	A	3mas	A
Putative Fimbrial Subunit	4hss	A	4hss	B
Gamma-Aminobutyric Acid Type B Receptor Subunit 1	4mqe	A	4ms3	A
Heat Shock Locus U	1do0	A	1do0	B

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
D-3-Phosphoglycerate Dehydrogenase (Phosphoglycerate 3 Dehydrogenase) (E.C.1.1.1.95)	1psd	A	1sc6	B
Molybdopterin Biosynthesis Protein Moea	2nqq	C	2nqq	A
47 Kda Membrane Antigen	1o75	B	1o75	A
Elongation Factor Tu	1ob5	C	2c78	A
Protein (Eukaryotic Peptide Chain Release Factor Subunit 1)	1dt9	A	3e1y	A
Aminopeptidase T	2ayi	D	2ayi	B
Succinyl-Coa Synthetase, Beta Chain	1eud	B	2fp4	B
Acarbose/maltose Binding Protein Gach	3k01	A	3jzj	A
Polymerase (Dna Directed) Kappa	1t94	B	2oh2	B
Molybdopterin Biosynthesis Moea Protein	1wu2	A	1wu2	B
Udp-N-Acetylglucosamine 2-Epimerase	3beo	A	1o6c	A
Actin	2zwh	A	1j6z	A
Maltose Abc Transporter, Periplasmic Maltose-Binding Protein	2gha	A	2ghb	B
D-Maltodextrin Binding Protein	1anf	A	1jw5	A
Chemotaxis Protein Chea	1b3q	A	2ch4	A
Collybistin II	2dfk	A	2dfk	C
Rho Guanine Nucleotide Exchange Factor 12	1x86	A	1txd	A
Uncharacterized Protein	4xe8	A	4xe7	A
Udp-N-Acetylglucosamine 2-Epimerase	4neq	A	4nes	A
3-Isopropylmalate Dehydrogenase	2y3z	A	4f7i	A
Choline Kinase Alpha	2ckq	B	2i7q	A
Dbh Protein	1k1q	A	2rdi	A
Leu/ile/val-Binding Protein	1z15	A	1z16	A
Nagk Protein	2ch6	D	2ch6	B
DNA Polymerase IV	3qz7	A	3bq1	A
Purine Nucleotide Synthesis Repressor	1jft	A	1dbq	A
Atp-Dependent Hsl Protease Atp-Binding Subunit Hslu	1im2	A	1qg4	A
Twitching Motility Protein Pilt	2gsz	A	2gsz	E
DNA Polymerase III, Delta Subunit	1jqj	D	1xxh	F
D-Lactate Dehydrogenase	1j49	B	1j4a	D
Ovotransferrin	1tfa	A	1iej	A
Rfcs	1iqp	C	1iqp	D
DNA Polymerase Beta	1bpd	A	2bpg	B
Igg Heavy Chain	1za6	B	1za6	D
Serotransferrin	1ryo	A	1bp5	C
Atp-Dependent Clp Protease Atp-Binding Subunit Clpx	3hws	A	3hws	B

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Parm	1mwk	A	1mwm	A
M-Calpain	1kxr	A	1ziv	A
Thioredoxin Reductase	1tde	A	1f6m	A
Translation Initiation Factor Eif-2b, Delta Subunit	3a9c	A	3vm6	A
Xylanase J	2dck	A	2dcj	A
Calpain 9	1ziv	A	2p0r	A
Ig Epsilon Chain C Region	4j4p	B	1o0v	B
Spectrin Alpha Chain, Brain	1u4q	B	1cun	B
Sugar Transport Protein	1tjy	A	1tm2	A
Nuclear Factor Nf-Kappa-B P105 Subunit	1ooa	A	2i9t	B
Manganese-Dependent Inorganic Pyrophosphatase	1k20	B	1k23	B
Interleukin-1 Receptor	1itb	B	1g0y	R
Manganese-Dependent Inorganic Pyrophosphatase	1k23	A	1wpm	A
Guanine Nucleotide Exchange Factor Dbs [Fragment]	1rj2	G	1rj2	J
D-Galactose-Binding Periplasmic Protein	2fw0	A	2hph	A
Sugar Abc Transporter, Periplasmic Sugar-Binding Protein	3c6q	A	3c6q	C
D-3-Phosphoglycerate Dehydrogenase, Putative	4nfy	A	4njm	A
Nf-Kappa-B P65	1nfi	C	2ram	B
Mrna Decapping Enzyme	1xmm	B	1xml	B
Probable Transcriptional Regulator	2esn	A	2esn	C
Virb11 Homolog	1nlz	F	1nlz	E
Hexokinase	2e2n	A	2e2o	A
Type IIe Restriction Endonuclease Naei	1ev7	A	1iaw	A
2-Dehydropantoate 2-Reductase	1ks9	A	2ofp	B
Ribose Abc Transporter, Periplasmic Ribose-Binding Protein	2fn9	A	2fn8	A
Titin	2ill	A	2nzi	B
Serine/threonine-Protein Kinase Pak 4	2cdz	A	2c30	A
D-Allose-Binding Periplasmic Protein	1gub	A	1rpj	A
Glutamate [Nmda] Receptor Subunit Zeta 1	1y20	A	1pbq	A
Nuclear Factor Of Activated T-Cells, Cytoplasmic 2	1owr	Q	1owr	M
Potassium Channel	2wln	A	2wlk	A
Pantothenate Synthetase	3ag5	A	3ag6	A
Diaminopimelate Epimerase	2q9h	A	2gke	A
Mhc Class I H-2dd Heavy Chain	1qo3	A	1ddh	A

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Glutamate Receptor Ionotropic, Nmda 2A	4nf5	B	3oel	A
D-Ribose-Binding Protein	1ba2	A	1urp	C
Osmoprotection Protein (Prox)	1sw4	A	1sw5	C
Udp-2,3-Diacylglucosamine Pyrophosphatase Lpxi	4ggm	X	4j6e	A
N-Methyl-D-Aspartate Receptor Subunit 1	1pbq	B	1pb7	A
Pectocin M2	4n58	A	4n59	A
Spac19a8.12 Protein	2qkm	B	2qkm	D
Phosphate-Binding Protein Psts 1	4exl	A	4lat	A
Probable Translation Initiation Factor 2 Alpha Subunit	1yz6	A	1yz7	A
Glutamate Receptor Subunit 2	1ftj	B	1fto	A
Glutamate Receptor 3	3dln	A	1fto	A
Acetylglutamate Kinase	2wxb	A	1gs5	A
Glutamate Receptor, Ionotropic Kainate 1	2f34	B	1ycj	B
Betaine Abc Transporter Permease And Substrate Binding Protein	3l6g	A	3l6h	A
Glutamate Receptor, Ionotropic Kainate 1	1s7y	B	1fto	A
Vinculin Isoform Vcl	1ydi	A	1rke	A
Glutamate Receptor, Ionotropic Kainate 3	4e0w	A	1fto	A
Nopaline-Binding Periplasmic Protein	4pow	A	4p0i	A
Angiostatin	1ki0	A	2doh	X
Endonuclease VIII	1k3x	A	1q3b	A
Major Surface Antigen P30	1kzq	A	1ynt	G
Alpha-1 Catenin	1h6g	A	1l7c	C
Mg2+ Transporter Mgte	2yvy	A	2yvz	A
Tight Junction Protein Zo-1	3lh5	A	3kfv	A
Replication Protein A 70 Kda Dna-Binding Subunit	1fgu	B	1fgu	A
Dihydrodipicolinate Reductase	1yl7	C	1p9l	A
Hypothetical Protein	2i76	A	2i76	B
Dihydrodipicolinate Reductase	3qy9	B	3qy9	D
Putative Abc Transporter, Periplasmic Binding Protein, Amino Acid	2yln	A	3zsf	A
Lysine, Arginine, Ornithine-Binding Protein	2lao	A	1lst	A
Glutamate Receptor Delta-2 Subunit	2v3t	A	2v3u	A
Windbeutel Protein	2c1y	A	2c0e	A
Dna-Directed RNA Polymerase Alpha Chain	1ynj	A	1ynn	B
Dna-Directed RNA Polymerase Alpha Chain	2a6h	A	1iw7	L
Fab 17B Heavy Chain	1rz8	B	2i60	R
Igg1 Antibody 58.2 (Heavy Chain)	1f58	H	3f58	H

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Abc-Type Transporter, Periplasmic Subunit Family 3	4psh	A	4prs	A
Fab-Ysd1 Heavy Chain	1za3	H	1za3	B
Glutamine Binding Protein	1wdn	A	1ggg	B
Hyb3 Heavy Chain	1w72	H	1dfb	H
Calcium-Gated Potassium Channel Mthk	2fy8	C	2fy8	H
Chimera Of Fab2c4: "Humanized" Murine Monoclonal Antibody	1l7i	H	1s78	F
Igg1 Fab Fragment	1igc	H	2aab	H
Catalytic Elimination Antibody 13G5 Heavy Chain	2gjz	B	2gk0	H
Chimeric Germline Precursor Of Oxy-Cope Catalytic Antibody Az-28 (Heavy Chain)	1d5i	H	1d5b	B
Fab Fragment, Heavy Chain	2h2s	E	2htl	C
Igg1 Fab Fragment (Hc19)	1gig	H	2vir	B
Antibody Light Chain	1jgu	L	1baf	L
Fab Fragment, Antibody A5b7	1ad0	B	1rmf	H
Igg Heavy Chain	2dd8	H	1rzi	F
Fab Fragment Of 8F5 Antibody Against Human Rhinovirus 3 Serotype 2 4	1bbd	L	1hin	L
K42-41l Fab Light Chain	1mju	L	1uz8	A
28B4 Fab	1kem	H	1q9l	B
Pc283 Immunoglobulin	1kcr	H	1kcu	H
Anti-Idiotypic Monoclonal Antibody (Light Chain)	2aab	L	1iqw	L
Igg 5C8	15c8	H	1fgn	H
Antibody M41	1gpo	L	1keg	L
Fab Heavy Chain	1xf3	H	1i8m	B
Immunoglobulin Gamma-1 Heavy Chain Constant Region	1e4k	B	2iwg	A
Igg1 Antibody 58.2 (Light Chain)	1f58	L	3f58	L
Immunoglobulin G1 (Igg1)	2mcg	1	2mcg	2
Loc - Lambda 1 Type Light-Chain Dimer	3bjl	B	1bjm	A
Mature Metal Chelatase Catalytic Antibody, Heavy Chain	1ngy	B	1n7m	L
Dihydrodipicolinate Reductase	1vm6	C	1vm6	B
Igg2a Fab Fragment (50.1)	1ggi	M	1ai1	L
Igm-Kappa Cold Agglutinin (Light Chain)	1dn0	C	1rhh	A
Immunoglobulin Lambda Light Chain	1jvk	B	1jvk	A
Monoclonal Antibody 2D12.5, Igg1 Gamma Heavy Chain	1gig	L	1q0x	L

Protein Name	Conformer 1		Conformer 2	
	PDB Code	Chain ID	PDB Code	Chain ID
Antibody Light Chain 11K2	2bdn	L	1osp	L
Immunoglobulin 48G7 Germline Fab	1gaf	L	1gpo	L
Fab 17B Light Chain	1rz8	A	2ny1	C
Fab Fragment Of Murine Monoclonal Antibody An02 Complex 3 With Its Hapten (2,2,6,6-Tetramethyl-1-Piperidinyloxy- 4 Dinitrophenyl)	1baf	L	1cz8	L
33H1 Fab Light Chain	1ors	A	1fig	L
Humanized Antibody Hfe7a, Light Chain	1it9	L	2gcy	A
Igg Antibody (Light Chain)	1emt	L	2a6i	A
Igg2b (Kappa)	1cgs	H	2cgr	H
Monoclonal Anti-Estradiol 10G6d6 Immunoglobulin Gamma-1 Chain	1jn6	B	1jnh	B
Pc287 Immunoglobulin	1kcu	L	1fsk	K
Antibody Light Chain Fab	1i8m	A	1qbm	L
Erythropoietin Receptor	1eer	B	1ern	B
17E8	1eap	A	1a0q	L
Fab E51 Light Chain	1rzf	L	1q1j	M
Fab Fragment, Antibody A5b7	1ad0	A	1rmf	L
Germline Metal Chelatase Catalytic Antibody, Chain H	1n7m	H	1ngy	A
Humanized Antibody D3h44	1pg7	H	1jps	H
Immunoglobulin	1ce1	L	1t04	C
Lambda III Bence Jones Protein Cle	1lil	A	1lil	B
Chimeric Germline Precursor Of Oxy-Cope Catalytic Antibody Az-28 (Light Chain)	1d6v	L	1axs	L
Hyb3 Light Chain	1w72	M	1adq	L
Fibroblast Growth Factor Receptor 2	1e0o	D	1djs	A
Septum Site-Determining Protein Minc	1hf2	C	1hf2	A
Vascular Cell Adhesion Molecule-1	1vsc	B	1vca	A
P58-Cl42 Kir	1nkr	A	2dli	A
N2b-Titin Isoform	2f8v	C	2a38	C
Transcriptional Regulator, Tetr Family	1zkg	A	1z77	A
Hypothetical Transcriptional Regulator In Qaca 5”Region	1jt0	A	1jtx	A
Muscle-Specific Kinase Receptor	2iep	B	2iep	A
Transcriptional Regulator	3vok	A	3vp5	A
Tenascin	1qr4	B	1qr4	A
Fatty Acid-Binding Protein, Epidermal	4azr	B	1b56	A