The Dissertation Committee for Jason A. Vertrees

certifies that this is the approved version of the following dissertation:

A Thermodynamic Definition of Protein Folds

Committee:

Vincent Hilser, PhD, Supervisor

Wlodek Bujalowski, PhD

Montgomery Pettitt, PhD

Robert Fox, PhD

Henry Epstein, MD

Dean, Graduate School

A Thermodynamic Definition of Protein Folds

by

Jason A. Vertrees, B.A., B.A.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas Medical Branch

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas Medical Branch

May 2008

To my father, for he has thoroughly impressed upon me the nature and value of knowledge and a sound education. For this I am deeply indebted.

To my wife, a paragon of perseverence and strength, without whom life would be less fulfilling or enjoyable.

Acknowledgments

This work would not have been possible without the help of many great people serving various roles. The support of my friends and family is very important. My wife and best friend, Stephanie Vertrees, has supported me throughout graduate school; and, as she promised through the good times and the bad. My parents—all three—were all very supportive throughout my time in graduate school.

My mentor, Vincent Hilser, was always an energetic source of inspiration and information. His incredible understanding of science, ability to absorb new knowledge at breakneck speeds, and intellectual curiosity are all enviable.

I also thank my friends: Rodrigo Maillard, MariaFe Lanfranco, Keerthi Gottipati, Brian Lemire, Meghan Keedy, Sergio Santa Maria, Miguel de Valdenebro, and Ashley Rioux. Without them, life outside the laboratory would have been quite drab. We enjoyed good times together and I am sure we have many more good times ahead.

Other faculty aside from Dr. Hilser played a significant role in my success. Dr. Jim Lee is a very strong student advocate. He supported me from day one, and I appreciate it very much. I have learned alot from him. Dr. Bryan Sutton had been secretly pushing me to learn PyMOL; his surreptitious actions helped me greatly. Without this, I would not have began the PyMolWiki or supported the PyMOL

program, which in the end helped me secure the first PyMOL Fellowship. I would also like to thank Dr. Wayne Bolen, Dr. Andres Oberhauser and the rest of the BSCB faculty from whom I took classes.

The administrators play a big role in enabling our success as well. Debora Botting has been an invaluable resource for many years. Other administrators and technical staff that were very helpful were Angelina Johnson, Lisa Pipper, Phil Barritt, Karen Jones, Lori Blackwell, Lindie Nanninga, Laura Teed, and Shirley Broz.

JASON A. VERTREES

The University of Texas Medical Branch

May 2008

A Thermodynamic Definition of Protein Folds

Publication No. _____

Jason A. Vertrees, Ph.D.
The University of Texas Medical Branch, 2008

Supervisor: Vincent Hilser

Modern techniques in structural biology, like homology modeling, protein threading, protein fold classification, and homology detection have proven extremely useful. For example, they have provided us with evolutionary information about protein homology which has in some many cases lead directly to therapeutics. Due to the importance of these methods, augmenting or improving them may lead to significant advances in understanding proteins. These methods treat the high-resolution structure as a static entity upon which they operate, however we know that proteins are not static entities—they are polymers that exist in an enormous array of conformational states. Therefore, we propose to model the proteins from a statistical thermodynamic viewpoint based upon their average energetic properties. We show

that this model can be used to (1) better characterize the partial unfolding process of proteins, and (2) reclassify the protein fold space from a new perspective.

Contents

List of Tables

List of Figures

List of Source Code

List of Abbreviations

3D	Three-Dimensional
4D	Four-Dimensional
$\forall$	Universal Quantification ("For All")
A	Alanine
AA	Amino Acid
Å	Angstrom, $(1 \times 10^{-10}\ m)$
ΔASA	Change in Accessible Surface Area
ap, apol	Apolar
ΔASA_{apol}	Apolar Change in Accessible Surface Area
ΔASA_{pol}	Polar Change in Accessible Surface Area
BLAST	Basic Local Alignment Search Tool
$^{\circ}C$	Degrees Celcius
C	Cysteine
cAMP	Cyclic Adenosine Mononucleotide (binding)
CATH	Class, Architecture, Topology, Homologous Family
CD	Circular Dichroism

CE	Combinatorial Extension
`CLARA`	Clustering Large Applications
`D`	Aspartic Acid
DALI	Distance Matrix Alignment
ΔC_p	Heat Capacity Change (at Constant Pressure)
$\det$	Determinate of a Matrix
DSC	Differential Scanning Calorimetry
`E`	Glutamic Acid
`F`	Phenylalanine
$\mathcal{F}$	Folded Subensemble
$\langle \circ \rangle_{\mathcal{F}}$	Folded Subensemble Energetic Property $\circ$
`G`	Glycine
ΔG	Gibbs' Free Energy
`H`	Histidine
ΔH	Enthalpy Change
HDex	Hydrogen-Deuterium Exchange
`I`	Isoleucine
$\in$	In (set inclusion)
$^{\circ}K$	Degrees Kelvin (absolute)
`K`	Lysine
$(kcal/mol)$	Kilocalories per mol
K_i	Statistical Weight for Microstate i
κ_i	COREX Stability Constant for Residue i

λ	Eigenvalue
L	Leucine
M	Methionine
MC	Monte Carlo
MD	Molecular Dynamics
MDF	Mean Deviation Form
N	Asparagine
NMR	Nuclear Magnetic Resonance
NPC	Non-deterministic Polynomial Time Complete
P	Proline
PCA	Principal Components Analysis
PDB	Protein Data Bank
PDBID	Protein Data Bank Accession Number
Pfam	Protein Family (database)
pol	Polar
PROT	Protein
Q	Glutamine
Q	Partition Function
R	R Project's R Program[1]
R	Arginine
R	Gas Constant
$\mathbb{R}^3$	Three-Dimensional Basis on the Real Number Line

[1] Ihaka and Gentleman (1996)

$\mathbb{R}^4$	Four-Dimensional Basis on the Real Number Line
RBP	Retinol Binding Protein
RESI	Residue
RMSD	Root Mean Square Deviation
S	Serine
ΔS	Entropy Change
SCOP	Structural Classification of Proteins
sd	Standard Deviation
Σ	Singular Values (matrix)
SS	Secondary Structure
SVD	The Singular Value Decomposition
T	Threonine
T	Temperature
TE	Thermodynamic Environment
Θ	Big-O, Execution Time
TIM	Triosephosphate Isomerase
TNF	Tumor Necrosis Factor
U	Left singular vectors
$\mathcal{U}$	Unfolded Subpartition
$\langle \circ \rangle_{\mathcal{U}}$	Unfolded Subensemble Energetic Property $\circ$
V	Valine
V	Right Singular Values
W	Tryptophan

Y	Tyrosine
Z^+	The Set of Positive Integers

Chapter 1

Introduction

Current methods in structural biology like homology modeling, threading, *ab initio* modeling, and fold space classification have proven extremely useful. They have allowed the elucidation of heretofore unknown evolutionary relationships between proteins (Holm and Sander, 1993), and in many cases have even led directly to treating disease (Zhao et al., 2000). While incredibly useful, inherent in all of these methods is the lack of dynamical information about the proteins. All of these models consider the protein as a static entity. But, a static structure cannot thoroughly account for all the properties of a dynamic macromolecule that samples an amazingly large array of conformers. Simply put, biologically relevant information is lost when one models the protein based solely upon a single static structure. To illustrate the importance of this point, consider the triosephosphate isomerase (TIM) barrel family of proteins (see Figure 1.1). The TIM barrels are classified into one family of structures due to their composition and organization of observed secondary structures, but the collection of TIM barrel proteins has well over two dozen

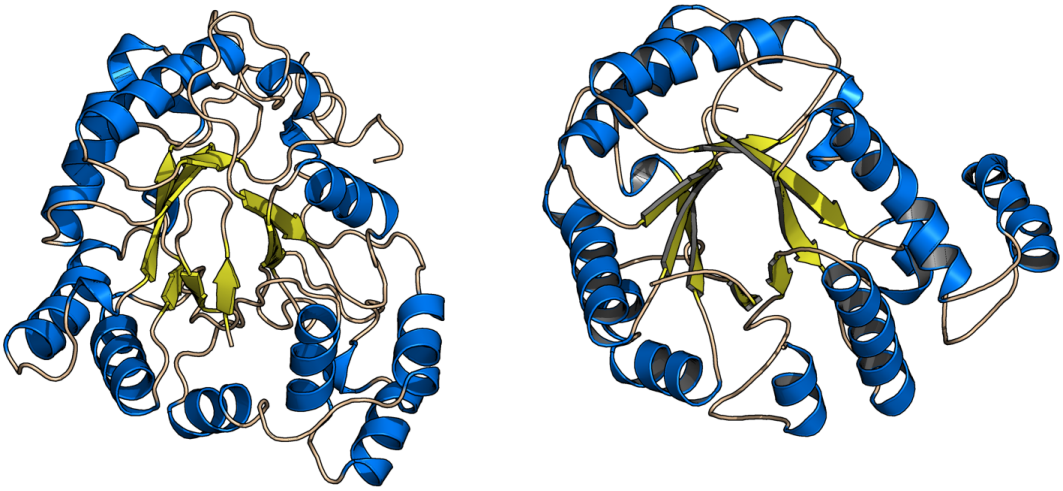


Figure 1.1: Two representative proteins (PDBID: 1CEN (Domìnguez et al., 1996), right; 1AH4 (Urzhumtsev et al., 1997), left) from the TIM-barrel protein family. They have similar structure, but different function. 1CEN is an aldose reductase while 1AH4 is a cellulase. The TIM-barrel family of proteins has over two dozen known functions, from one similar scaffold.

disparate functional characteristics (Qian et al., 2001). This is one example where observations belie current dogma: how can a protein that looks very similar to another have completely unrelated function? Consider another important problem: the evolutionary progression of protein folds. When the protein fold space—the organized collection of all protein folds—is clustered into a complete hierarchy, we can surmise that proteins at the very lowest levels (most similar) of the hierarchy are evolutionarily related by some ancestral fold, but we cannot make any credible statements about the evolutionary organization of the fold space from the point of families of folds (higher up in the tree). Figure 1.2 illustrates this concept. At the lowest parts of the hierarchy, the proteins are related by some common ancestral fold. However, at the level of grouping families, we cannot make any assumptions as to the evolutionary history of the families.

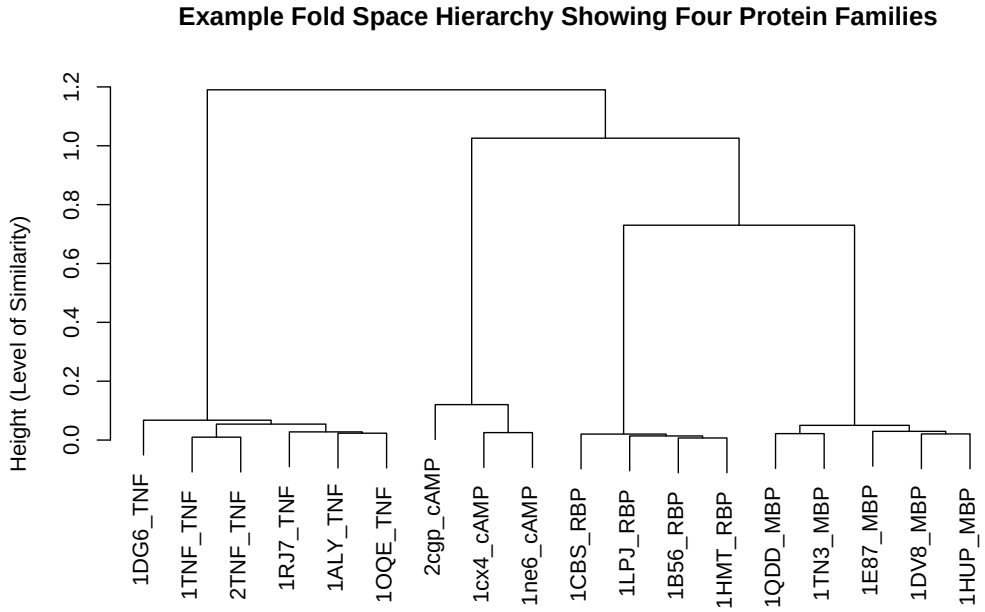


Figure 1.2: Example hierarchy showing four protein families. Each of the four protein families can be seen as descending from one common ancestral protein. However, we cannot make any credible statements about the relationship between the tumor necrosis factor (TNF) family and the cyclic adenosine monophosphate binding protein (cAMP), or the retinol binding (RBP) proteins.

Protein fold progression and the evolutionary relationship between folds are part of the foundation used throughout traditional methods of protein comparison. Homology modeling is successful if the target sequence is similar enough to the source. The success of protein threading also is directly related to the evolutionary distance between the target and source. Because the current definition of protein fold space is undergoing refinement (Honig, 2007; Kolodny et al., 2006), novel information concerning the relationship between protein folds or their constituent

pieces will be of value to the field.

Hypothesis and Goals Noting the lack of dynamical information in current methods in structural biology, we believe that by representing proteins as dynamic entities that we can provide novel information, augmenting traditional methods of structural biology. To that end the goals of my project are to:

- represent proteins as thermodynamic "structures";
- elucidate the biophysical meaning of the dynamic structures and fragments;
- redefine the protein fold space based upon the dynamic structure model.

Chapter 2

Background and Significance

Introduction Before discussing our hybrid structural-thermodynamic representation of proteins, it may be necessary to briefly introduce statistical mechanics and thermodynamics. Exact mechanics or simply, mechanics, is the science of moving objects. For example, Newton studied the laws of mechanics and devised his very well-known three laws of motion. For describing simple objects, mechanics is helpful. However, when one wants to study a large collection of objects, say for example gas molecules, or proteins, or even money, one cannot rely simply on exact mechanics due to computational intractability of calculating the exact solution[1]. When dealing with large collections of objects, or ensembles, one then takes advantage of statistical mechanics. Statistical mechanics is the application of probability theory to mechanics.

[1] Admittedly much progress has been made using exact mechanics for macromolecular research, called molecular dynamics (MD), but the field is limited by the inherent exponential running time of the calculations. Thus, biologically relevant time scales are difficult to realize with MD.

Statistical Thermodynamics Statistical thermodynamics is, similarly, the application of probability theory to many-bodied systems under thermodynamic control. The goal of statistical thermodynamics is to understand the observable macroscopic properties of these systems by linking statistical and probabalistic tools to the thermodynamic equations defining the system. The problem of statistical thermodynamics as one of the founders, Schrödinger writes,

> . . . is, essentially, only one problem [in statistical thermodynamics]: the distribution of a given amount of energy E over N identical systems. (Schrödinger, Erwin, 1952)

There is a basic recipe for most statistical thermodynamic problems. Given some object to study: (1) make many virtual copies of the object (called a *Gibbs' Ensemble*); (2) perturb each copy in some small, but measurable way, (this slightly perturbed state is called a *microstate*); (3) calculate an energy for each microstate; (4) calculate relative probabilities for each microstate by comparing individual microstates to the sum of all microstates in the ensemble. The sum of all microstates is also known as the partition function. The energy-weighted properties of the microstates are then summed up over the entire ensemble. This yields "ensemble-averaged" macroscopic results—a probabalistic representation of the entire collection of states. The power of statistical thermodynamics lies in its ability to allow interpretation of macroscopic observables from microscopic properties.

2.1 COREX: A Structural Thermodynamic Model of Proteins

2.1.1 History

Linking Statistical Mechanical Theory to Observation In the late 1970s, Freire and Biltonen demonstrated a method to predict the existence of, and thermodynamically characterize, equilibrium folding-unfolding intermediates (Freire and Biltonen, 1978). They discovered that this statistical mechanical model accurately represented the observed differential scanning calorimetry (DSC) unfolding profile of various proteins. In fact the theoretical framework was general enough to be applied to any two-state transition in single domain proteins. The apparent excess enthalpy, $\langle \Delta H \rangle$, was derived by integrating the excess heat capacity, $\langle \Delta C_p \rangle$, measured from the DSC unfolding experiments. The partition function Q was then derived as:

$$Q_T = \exp\left(\int_{T_0}^{T} \frac{\langle \Delta H \rangle}{RT^2} \right) dT \ ,$$

where $\langle \Delta H \rangle$ is the calculable excess enthalpy from DSC. This links the theoretical partition function to the observable enthalpy and allowed the theory to be applied to various systems and extended to include structural thermodynamics.

Structural Thermodynamics A few years after the theoretical framework of Freire and Biltonen had been devised, two important experimental observations arose. The first was the discovery that one could linearly relate the energetics of unfolding (ΔC_p, ΔH, ΔS) of small polypeptides called "model compounds"

to the amount of surface area being exposed in the unfolding event (Murphy and Gill, 1990). And, the next important observation arose from nuclear magnetic resonance (NMR) based hydrogen-deuterium exchange (HDex) experiments. This experimental methodology brought to light the nature of "local unfolding," showing the existence of thermodynamically predicted intermediate equilibrium states in proteins (Baum et al., 1989). In HDex experiments one measures the protection factor or the rate at which solvent deuteriums exchange with the protein's hydrogens. The residues with more protection—those that are folded away from solvent or considered more "stable"—have lower rates of exchange.

Knowing that there exist experimentally verifiable equilibrium intermediate states and a method to link the change in surface area of the intermediate states to the energetics of the transition, the theoretical model posited by Freire and Biltonen could be further advanced to model the energetics of the transition state intermediates through estimable calculations of their surface area changes (Freire and Murphy, 1991; Murphy and Freire, 1992). For each intermediate state, called a "microstate", i, the Gibbs-Helmholtz equation can be used to calculate the free energy, ΔG_i, of the microstate:

$$\Delta G_i = \Delta H_i - T\Delta S_i \ ,$$

which with non-zero ΔC_p expands to

$$\Delta G_i = \Delta H_i(T_{0,H}) - T\Delta S_i(T_{0,S}) + \Delta C_{p,i}\left[(T - T_{0,H}) - T\ln\frac{T}{T_{0,S}}\right] \ . \qquad (2.1)$$

I shall explain the derivation of each term in Equation 2.1. First, the change in heat

capacity, $\Delta C_{p,i}$ is related to the ΔASA by the linear function:

$$\Delta C_{p,i} = 0.45\Delta ASA_{apol} - 0.26\Delta ASA_{pol} \ . \tag{2.2}$$

This is the best-fit equation derived from a dataset of model compounds (Murphy and Gill, 1991; Murphy et al., 1990). Next, it was shown (Xie and Freire, 1994) that the enthalpy of the reaction, ΔH_i, could also be linearly related to ΔASA by

$$\Delta H_i(T = 60) = -8.44\Delta ASA_{apol} + 31.4\Delta ASA_{pol} \ . \tag{2.3}$$

Lastly, the entropy, ΔS_i, at the temperature T is related to the ΔASA as:

$$\Delta S_i(T) = \Delta C_{p,apol} \ln\left(\frac{T}{385}\right) - \Delta C_{p,pol} \ln\left(\frac{T}{335}\right) \ , \tag{2.4}$$

where the $385°K$ and $335°K$ the temperatures at which the apolar and polar solvation entropies are zero, respectively (D'Aquino et al., 1996; Murphy and Freire, 1992).

It is worth noting that because the energy function ΔG_i is based upon the apolar and polar surface area changes that each thermodynamic quantity discussed, ΔG_i, ΔH_i, ΔS_i, can be further decomposed into their apolar and polar contributions. For example, $\Delta G_{i,\text{total}} = \Delta G_{i,\text{apolar}} + \Delta G_{i,\text{polar}}$.

Now that the energetics of transition states has been expounded upon, we can consider the statistical probability of any given microstate being populated in

the Gibbs' ensemble. Each microstate was given a statistical weight,

$$K_i = \exp\left(-\Delta G_i / RT\right) \ .$$

The statistical weight when related to the partition function defines the probability that the given microstate will be populated in the ensemble of states. Symbolically, the probability of a given microstate i in relation to the partition function Q is:

$$P_i = \frac{K_i}{Q} = \frac{K_i}{\sum_{i=1}^{N} K_i} \ .$$

Once probabilities of microstates are known, they are weighted against a signal emitted by that microstate. This is exactly how equilibrium based experiments, like circular dichroism (CD), HDex, NMR, and X-ray crystallography work. For example, imagine that we want to monitor the progress of protein unfolding through the measurement of tryptophan fluorescence. Because the protein is in equilibrium, sometimes the tryptophan is protected from solvent (emitting) and other times in solvent (quenched). Ignoring for now the complexities involved in the amount of photon emission define I_i as the level of emission from state i. Then, as we monitor the reaction as a function of some variable, we observe that the fluorescence can be modeled as: $\langle I \rangle = \sum P_i I_i$. That is, the observed signal from the states that are more dominant (higher P_i) in the ensemble are more highly represented that those states with a lower probability.

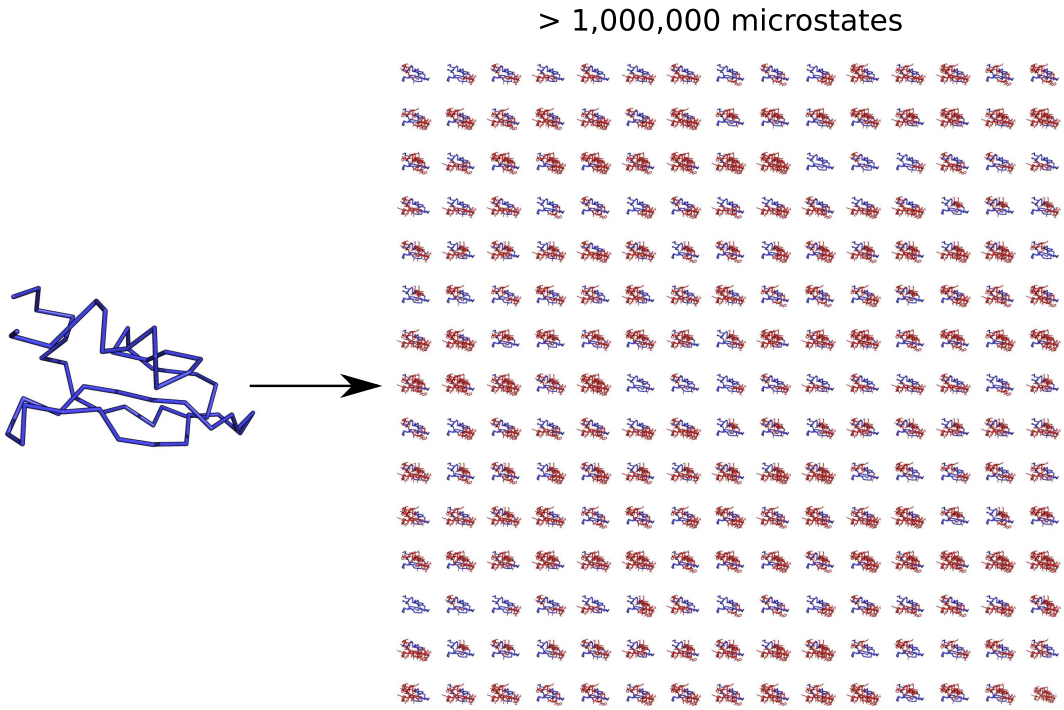


Figure 2.1: A schematic diagram showing how COREX creates the Gibbs' ensemble. COREX performs combinatorial local unfolding on the native protein structure. All possible states in the equilibrium folding-unfolding ensemble are sampled. Each state is assigned a probability in the ensemble. Then, the thermodynamic properties at each state are weighted by that state's probability. Ensembles typically contain more than 1,000,000 microstates.

2.1.2 The COREX Model Defined

In 1991, Freire and Murphy defined a method to partition the protein sequence into windows of unfolding (Freire and Murphy, 1991). For each window a ΔASA would be calculated and given a Gibbs' free energy as explained above. They defined their partition boundaries at secondary structure boundaries. The assumption was that secondary structural elements would act cooperatively and typically be more stable. This increased the speed of the calculations of the structural thermo-

dynamics, at the cost of fine-grained accuracy.

Hilser and Freire, in 1996, devised the sliding-window concept to define windows of unfolding (Hilser and Freire, 1996; Hilser et al., 1996). This algorithm, while computationally more costly, allowed more precise calculations of the structural thermodynamics at the residue-level. Instead of fixed window boundaries aligned with secondary structure boundaries, COREX calculates the local unfolding of fixed-sized windows with sliding boundaries across the entire protien. It then unfolds each window, combinatorially, and calculates an energy for the microstate. The protein ensemble is created by sampling all possible states in the equilibrium, from fully folded to fully unfolded. Figure 2.1 illustrates the partially unfolded ensemble. COREX unfolds windows, instead of single residues to aide in the problem of the exponential nature of the calculation. That is, to sample a protein of N residues by locally unfolding all combinations of a single residue requires $2^N - 1$ calculations, whereas with a window size of w the calculation is pared down to $2^{N/w} - 1$ calculations. COREX also ensures that a residue is sampled close to and far from boundaries by sliding the windows. This removes edge-effects from boundary conditions that the residue may experience.

Ensemble-Average Residue-Level Energetics COREX is able to determine ensemble-average energetics for microstates, as well as for residues. This allows COREX to calculate local viewpoints of residue energetics across the protein. Figure 2.2 shows an example stability profile $[\Delta G]$ for the Kunitz Type Domain C5 Protein (PDBID: 1KTH). The ordinate axis units are in energetic units of (cal/mol). Calculation of the ensemble-average residue-level energetics for a

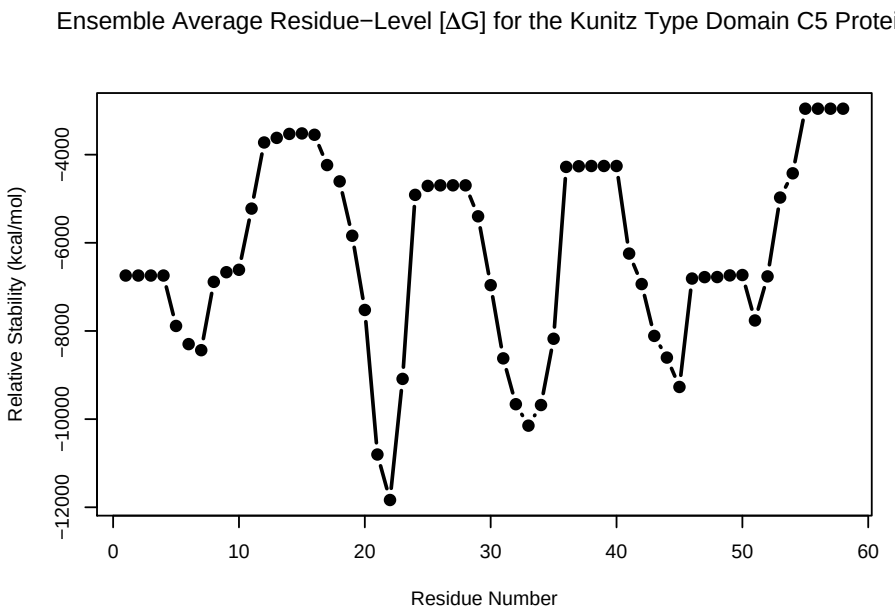


Figure 2.2: Ensemble average stability profile $[\Delta G]$ for the Kunitz Type Domain C5 Protein. This plot was made by using Equation 2.13 on the thermodynamic output data from COREX.

residue is done as follows. For some residue under investigation, r, and the entire ensemble of states, S, let

$$\mathcal{F}_r = \{\forall \text{ microstates } s \in S \mid \text{residue } r \text{ is folded in microstate } s\} \ ,$$

and

$$\mathcal{U}_r = \{\forall \text{ microstates } s \in S \mid \text{residue } r \text{ is unfolded in microstate } s\}$$

be the sets of microstates (subensembles) where residue r is folded and unfolded, respectively. Then, the energetics associated with the folded subensemble are

$$\langle \Delta G \rangle_{\mathcal{F}_r} = \sum_{\mathcal{F}_r} P \cdot \Delta G \tag{2.5}$$

$$\langle \Delta H_{apol} \rangle_{\mathcal{F}_r} = \sum_{\mathcal{F}_r} P \cdot \Delta H_{apol} \tag{2.6}$$

$$\langle \Delta H_{pol} \rangle_{\mathcal{F}_r} = \sum_{\mathcal{F}_r} P \cdot \Delta H_{apol} \tag{2.7}$$

$$\langle \Delta S_r \rangle_{\mathcal{F}_r} = \sum_{\mathcal{F}_r} P \cdot \Delta S \, . \tag{2.8}$$

The unfolded-ensemble averages are likewise,

$$\langle \Delta G \rangle_{\mathcal{U}_r} = \sum_{\mathcal{U}_r} P \cdot \Delta G \tag{2.9}$$

$$\langle \Delta H_{apol} \rangle_{\mathcal{U}_r} = \sum_{\mathcal{U}_r} P \cdot \Delta H_{apol} \tag{2.10}$$

$$\langle \Delta H_{pol} \rangle_{\mathcal{U}_r} = \sum_{\mathcal{U}_r} P \cdot \Delta H_{apol} \tag{2.11}$$

$$\langle \Delta S_r \rangle_{\mathcal{U}_r} = \sum_{\mathcal{U}_r} P \cdot \Delta S \, . \tag{2.12}$$

The differences in the folded and unfolded weighted subensembles for each thermodynamic property yield the residue-level ensemble-average thermodynamic proper-

ties. Symbolically,

$$
\begin{aligned}
[\Delta G]_r &= \langle \Delta G \rangle_{\mathcal{F}_r} - \langle \Delta G \rangle_{\mathcal{U}_r} && (2.13)\\
[\Delta H_{apol}]_r &= \langle \Delta H_{apol} \rangle_{\mathcal{F}_r} - \langle \Delta H_{apol} \rangle_{\mathcal{U}_r} && (2.14)\\
[\Delta H_{pol}]_r &= \langle \Delta H_{pol} \rangle_{\mathcal{F}_r} - \langle \Delta H_{pol} \rangle_{\mathcal{U}_r} && (2.15)\\
[\Delta S]_r &= \langle \Delta S \rangle_{\mathcal{F}_r} - \langle \Delta S \rangle_{\mathcal{U}_r} && (2.16)
\end{aligned}
$$

The residue-level energetics in Equations 2.13–2.16 are the basis for amino acid specificity (Wrabl et al., 2002) and fold discrimination (Larson and Hilser, 2004). The elucidation of the biophysical meaning of the residue-level ensemble-average thermodynamics is a major part of this project, and is discussed at length in Section 3.3.

Ensemble-based Polar/Apolar Enthalpy with Respect to Polar/Apolar Accessible Surface Area Changes Using Equations 2.2 and 2.3 we can rearrange terms and solve for $\Delta ASA_{\text{apolar}}$ and $\Delta ASA_{\text{polar}}$ given ΔH_{apolar} and ΔH_{polar}. Through collection and rearrangement of terms we see that,

$$
\begin{aligned}
\Delta H(25) &= \Delta H(60) + \Delta C_p(T - 60) \\
&= -8.44\Delta ASA_{\text{apol}} + 31.4\Delta ASA_{\text{pol}} + \\
&\quad [0.45\Delta ASA_{\text{apol}} - 0.26\Delta ASA_{\text{pol}}]\,(T - 60)\ .
\end{aligned} \qquad (2.17)
$$

Let $T^0 = T - 60$. Which leads to,

$$\Delta ASA_{\text{apol}} = \frac{\Delta H_{\text{apol}}(25)}{a_H + a_{Cp} * T^0} \ , \tag{2.18}$$

$$\Delta ASA_{\text{pol}} = \frac{\Delta H_{\text{pol}}(25)}{b_H + b_{Cp} * T^0} \ . \tag{2.19}$$

We must be cautious about the interpretation of Equations 2.18 and 2.19. The ΔH in the numerators are ensemble averages of the difference in enthalpy between the unfolded subensemble and the folded subensemble. These equations will be used later on to investigate the biophysical meaning of the thermodynamic environment space.

2.1.3 Applications and Validation

COREX is founded in the formal theory of statistical thermodynamics. Furthermore, the energy function is derived from experimental observation of ΔASA and how it relates to K_i. Here I will show that this foundation provides it with the means to model various experiments that serve as validation of the methods. Further applications and validation of the COREX algorithm are discussed below.

Hydrogen-Deuterium Exchange COREX was shown to accurately recreate the HDex protection factors for hen egg-white lysozyme, equine lysozyme, bovine pancreatic trypsin inhibitor, staphylococcal nuclease, and turkey ovomucoid third

domain (Hilser and Freire, 1996). The natural log of the stability constant,

$$\ln \kappa_r = \ln \frac{\sum_{\mathcal{F}_r} P_{\mathcal{F}_r}}{\sum_{\mathcal{U}_r} P_{\mathcal{U}_r}} ,$$

defined for each residue in the protein models the HDex protection factors mentioned above.

Fold Recognition Using COREX it was shown that amino acids had propensities for certain *thermodynamic environments* in proteins (Wrabl et al., 2001). Then, it was suggested that the thermodynamic environments could be used for discrimination of fold specificity (Wrabl et al., 2002). Larson et al then demonstrated this, when position-specific scoring matrices were trained with strictly thermodynamic information instead of amino acids sequence information and then used to detect the amino acid sequence from a decoy data set (Larson and Hilser, 2004). This method is similar to the one used by Gribskov (Gribskov et al., 1987), but used strictly thermodynamic parameters. The results of this study indicated that: (1) when the entire space in which the residues lie is partitioned into eight regions, fold recognition is achieved with 84% success. Interestingly, the eight partitions are independent of secondary structure. Larson showed that when the PSSM was trained using only alpha proteins, that when tested against all beta proteins, the fold recognition was nearly as successful. This indicated the same information was encoded into both alpha and beta proteins.

This project reaches one step further, to define an energetic fold, not just discriminate between them.

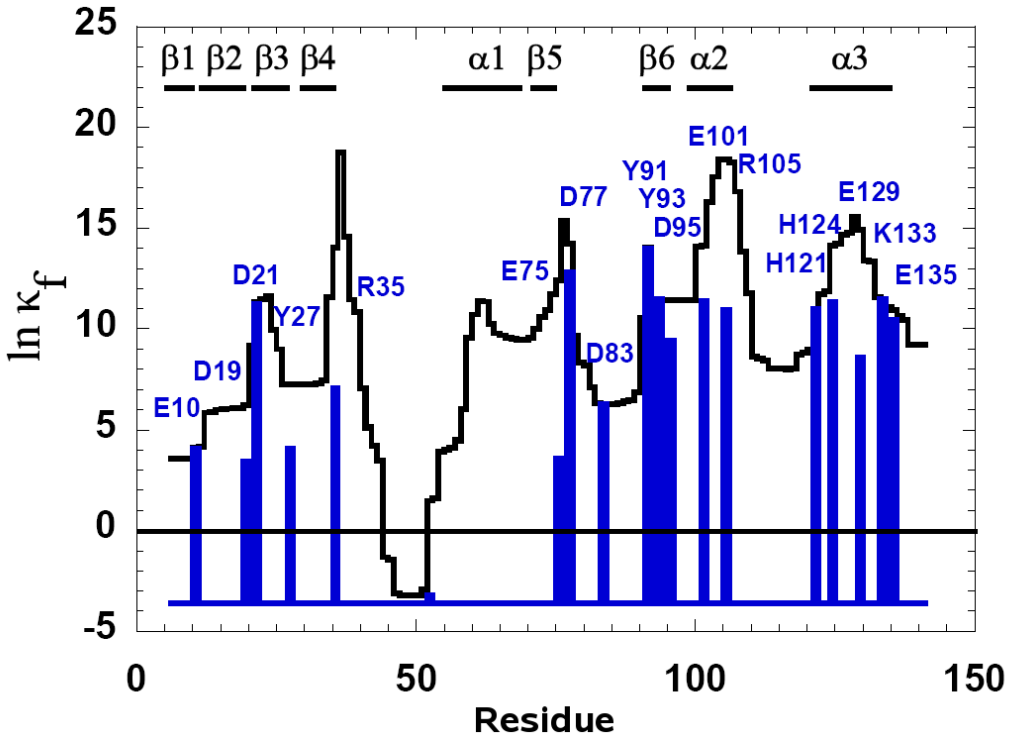


Figure 2.3: The natural logarithm of the residue stability constants (black lines) and residue protection constants (blue lines) calculated for the PHS ensemble. The COREX model identifies the residues responsible for the observed difference in proton binding upon acid unfolding with respect to the static calculations. The residues of interest are those in which the residue stability constants does not match the residue protection constants. Image courtesy of Steve Whitten. Used with permission.

pH Effects Recently, COREX has been used to represent local pH effects in proteins (Whitten et al., 2005). Local conformational fluctuations have been correlated to proton binding in staphylococcal nuclease. See Figure 2.3, for details.

Functional Residue Detection Liu et al have shown that COREX can be used to detect functional residues in proteins. She discovered that residues that are functionally relevant have statistically significant higher effects upon mutation of affect-

ing the calculated ensemble (Liu et al., 2006).

2.2 Current Methods in Protein Structure Comparison and Fold Space Classification

Before discussing the dissection of the protein fold space, one needs to understand how to quantitatively compare proteins. I start with expounding on protein comparison using sequence methods then move to structural methods. After the quantitative methods are introduced, then I discuss how the field puts these methods to use in protein fold space classification.

2.2.1 Sequence Alignments

The first methods of protein homology detection were through sequence alignment techniques. In protein sequence alignment, each amino acid is assigned a symbol from some alphabet. Each protein sequence is then treated as an array of symbols. Two proteins sequences are considered related if their two sequences have a high number of overlapping symbols (Altschul et al., 1990) or if a properly weighted matrix could be used to detect homologous sequences from a database of decoy sequences (Gribskov et al., 1987). Typically, statistics are used to identify uncommonly high matches. For example, one can relate a series of sequences by pairing together the two sequences with the highest Z-score. The Z-score is a statistical measure of how far away from the mean a data point is in a normal distribution, in units of the standard deviation (sd). By choosing paired sequences with a high

PDBID	Amino Acid Sequence	Last
1CLL	----EFLTMMARKMKDTDSEEEIREAFR----------VFDK	94
	|||||||||||||||||||||||| ||||	
1GGZ	----EFLGMMARKMKDTDNEEEIREAFR----------VFDK	94
	| || | |	
1PXY	DGKYAFAEMMTEDVETCRDERCYRLWINSLGIDSYVNNVFED	300

Table 2.1: An example of a multiple protein sequence alignment. The first two sequences are from homologous proteins (PDBIDs: 1CLL, 1GGZ) (top two rows). The last is a protein sequence (PDBID: 1PXY) chosen at random to illustrate low sequence identity. The PDBIDs appear at the left, the sequence in the middle, and the residue number of the last residue on the right. The vertical bar (|) between two rows indicates an identity between two sequences.

Z-score we can be statistically assured of a close relationship between the two sequences. Table 2.1 shows a sample multiple sequence alignment. Extensions of these method were introduced like allowing gaps and recalculating penalty scores for mismatched but biophysically similar residues (Henikoff and Henikoff, 1992; Schwartz and Dayhoff, 1979). Soon after, multiple sequence alignments (Bateman et al., 2002; Thompson et al., 1994) were introduced as a way of classifying many sequences into fold families.

The problem of sequence alignment (for two sequences) is in the class of polynomial time solvable problems, P. This means exact solutions are essentially guaranteed and are fast. The standard running time for two protein sequences of lengths m and n, is $\Theta(m \cdot n)$. Most algorithms solve the problem through dynamic programming (Needleman and Wunsch, 1970; Smith and Waterman, 1981).

Protein sequence alignment is still tremendously useful. And, because there are far more known protein sequences than high resolution structures, protein se-

quence alignment will continue to be used in structural biology. But, protein sequence alignment suffers from that fact that protein sequences are far less conserved than are protein structures. Therefore, protein structure alignment exploded in popularity soon after protein sequence alignment.

2.2.2 Protein Structure Alignments

It was observed early on that protein structures have redundant structural features at both the secondary-structure (Chothia and Levitt, 1977), and tertiary-structure levels (Holm and Sander, 1996). As an example to illustrate the point, Figure 2.4 shows four randomly selected helices from the PDB. The four helical segments matched each other to no more than 0.484 Å RMSD. The four helical structures are so similar that their four cartoon backbones when superimposed look like one. In fact there is so much redundancy, that many protocols for protein classification involve a first step of removing redundancies (Holm and Sander, 1999). Also, structure alignments can be improved through consideration of the amino acid sequence (Brenner et al., 2000).

Structure Definition The problem of structure alignment is important in structural biology: it is believed that a "high degree" of protein structure implies evolutionary consanguinity. So to attack the problem of protein structure alignment, the problem should be stated precisely.

Structure alignment has two steps to it. The first step is matching which residues from the first protein are to be matched to which residues from the second

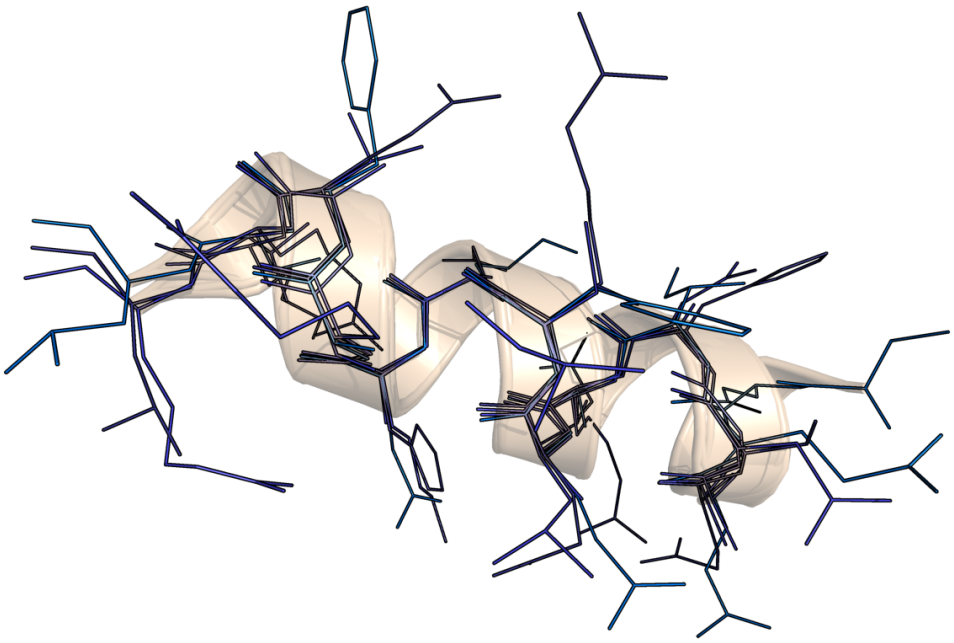


Figure 2.4: Image showing four randomly selected helices from the PDB. The helical segments were then aligned using the "super" command in PyMOL (DeLano, 2002). The maximum deviation between the four structures is 0.484 Å RMSD. The regularity of the helical structure is exemplified in the overlap of the amino acid sidechains from truly randomly selected proteins.

protein. This step is computationally very expensive. In fact it's been proven to be in the set of non-deterministic polynomial time complete (NPC) solvable problems (Goldman et al., 1999). Therefore, one has to relax the problem or perform some heuristic optimization to approach the exact solution. The second step is optimally aligning and scoring the deviation of the matched residues. This requires a rigid-body rotation and translation of one set of points onto the other. A beautiful, closed form solution to the problem of optimal superposition was published Wolfgang Kabsch, in 1976 (Kabsch, 1976) and made more accessible in 1978 (Kabsch, 1978). The details of Kabsch's algorithm are discussed in Section 6.2.4. Since then

other methods have been established (Coutsias et al., 2003).

2.2.3 Protein Fold Space Dissection and Organization

Once a method of protein alignments has been devised, one can compare each protein in a data set to each other protein. This yields a (square, symmetric) matrix of scores identifying the level of similarity among each protein. This similarity matrix can then be clustered, or otherwise partitioned to define fold families, or classes. Various methods have been used to cluster the protein fold space (Day et al., 2003; Holm and Sander, 1993; Hou et al., 2003; Shindyalov and Bourne, 1998). The concept of protein fold space has been advancing from a discrete definition to a continuous definition (Kolodny et al., 2006). As a visual example of what it means to partition fold space, consider Figure 2.5. This image shows the clustering results of structurally comparing seven families of proteins. Each protein is colored by which class it has come from. Note how the colors cluster appropriately in the projected space.

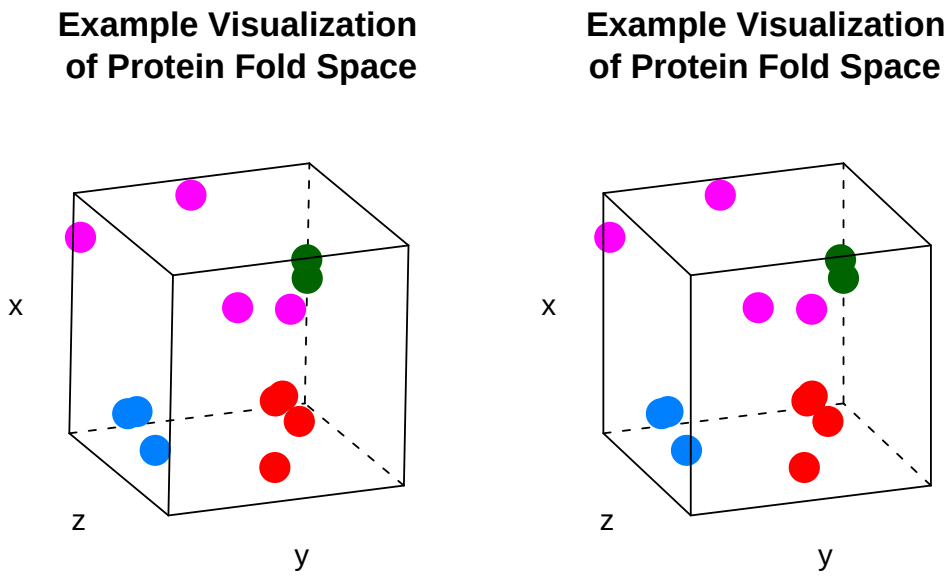


Figure 2.5: Stereo image showing the virtual (projected) protein fold space. Each protein is represented by a dot colored according to the family of the protien. This original space is 13-dimensional. To create this image, I used a linear subspace decomposition (isoMDS) technique to organize and project the data into a 3D representation. Note how each class indeed clusters around some central location in the projected space. Boundaries for this data would be very easy to draw.

Chapter 3

The Thermodynamic Structure Definition of Proteins

3.1 Thermodynamic Structure Definition of Proteins

The Thermodynamic Definition of a Protein As we know, the typical protein structure is defined by the atomic coordinates of each atom in the high-resolution structure file. So, for each atom we have its corresponding (x, y, z) coordinates as a vector in $\mathbb{R}^3$. In this project, we define the **thermodynamic structure** of a protein by assigning to each amino acid a vector in $\mathbb{R}^4$ defining its location in some space. The coordinates that define the vectors are the ensemble averaged residue-level thermodynamic properties from the COREX output $[\Delta G]$, $[\Delta H_{\text{apol}}]$, $[\Delta H_{\text{pol}}]$, $[\Delta S]$ (Equations 2.13–2.16). To contrast the ideas and provide an example ther-

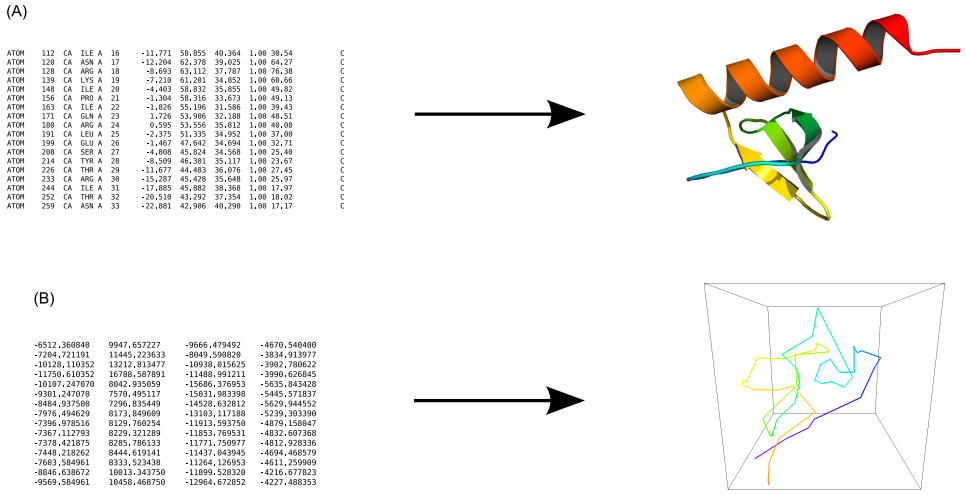
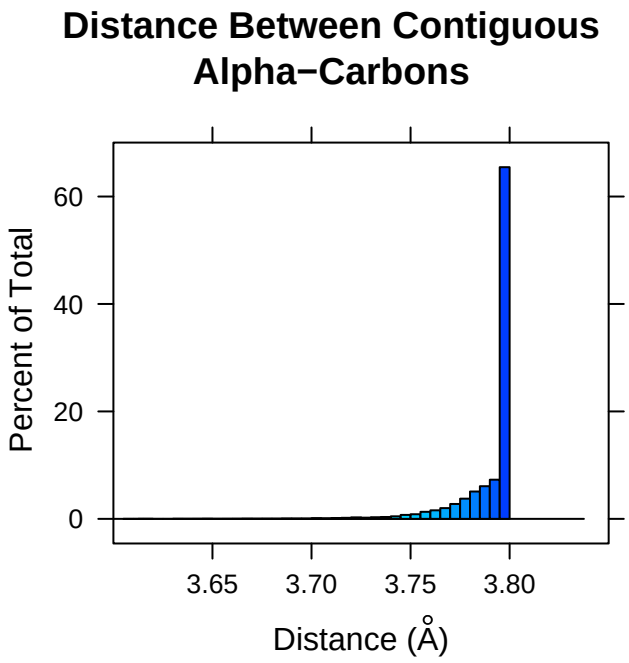


Figure 3.1: Comparison of Protein Structure and Thermodynamically Defined Protein Structure. (A) A table is shown (top left) with a snapshot of the atomic coordinates for the Monocyte Chemotactic Protein 2 (PDBID: 1ESR). When these coordinates are plotted into a 3D space, the result is an image of the structure (top right). (B) A table (bottom left) is shown with a snapshot of the 4D thermodynamic coordinates for the same protein as above. The first three dimensions of the data are plotted and the resultant structure shown (bottom right). Both proteins are colored along their sequence.

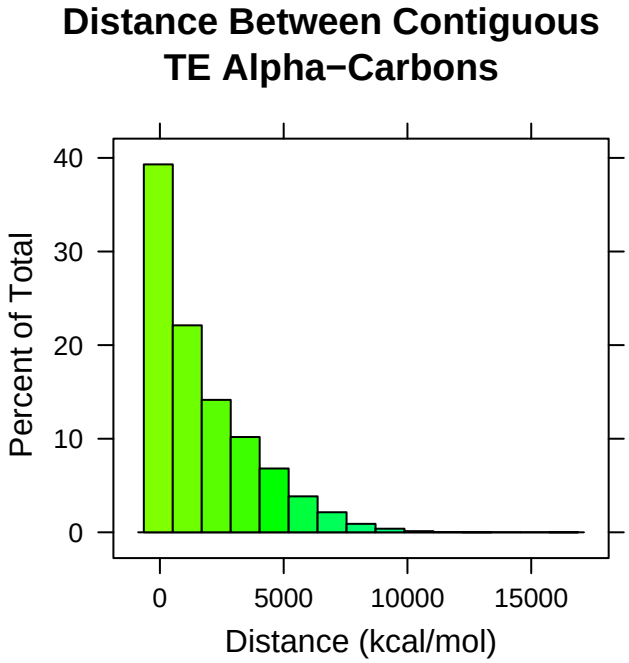
modynamic structure, Figure 3.1 shows a comparison of typical structure definition with our thermodynamic definition.

3.1.1 Thermodynamic Structure Data do not Mirror Typical Protein Structure Data

Our data set has various attributes that are very different with respect to that of normal protein structure space. Typical protein structure data observe rules enforced by physics and chemistry. First, two atoms within a structure cannot occupy the



(a)



(b) Histogram of Distances Between Contiguous

Figure 3.2: Comparison of Distances Between Contiguous Residues. Notice how more than 99% of the typical structure distances are within the range of 3.4–3.8 Å. TE structure definition has no such constraint. Over 17,000 residue-residue distances were calculated for these two plots.

same place at the same time. Our energetic data can, and do. Second, two alpha-carbon residues contiguous in sequence must be within about 3.4–3.8 Å due to bond lengths. Our data show that sequence neighbors can have very large energetic jumps in TE space. A comparison of the contiguous residue distance in typical structure space and TE space are shown in Figure 3.2. Notice that the variance in structure data is very small in comparison to our TE data. The plot also indicates that energetic jumps of up to about 20,000 (cal/mol) are still commonplace.

This deviation from typical structure-based rules is important to recognize. Current methods in structure alignment make many assumptions about protein structure to decrease the running time of the algorithm. Because of the features of this data set, we have to investigate every assumption for validity with our data. In fact, many changes were implemented from current methods to detect commonalities in thermodynamically defined protein structures. These changes are discussed in Section 6.2.3.

3.2 Thermodynamic Environment (TE) Space

We define the **thermodynamic environment space**, or **TE space** as the subspace spanned by energetic dimensions $[\Delta G]$, $[\Delta H_{apol}]$, $[\Delta H_{pol}]$, $[\Delta S]$ of the COREX output. TE space is simply the space taken up by the collection of points that are output from COREX. An example of the TE space used in this project is shown in Figure 3.3. Over 17,000 points in four dimensions are plotted; this is raw output from the COREX algorithm run over a database of proteins. (The actual database used in this study is discussed in Section 6.1 on page 87.) The original energetic

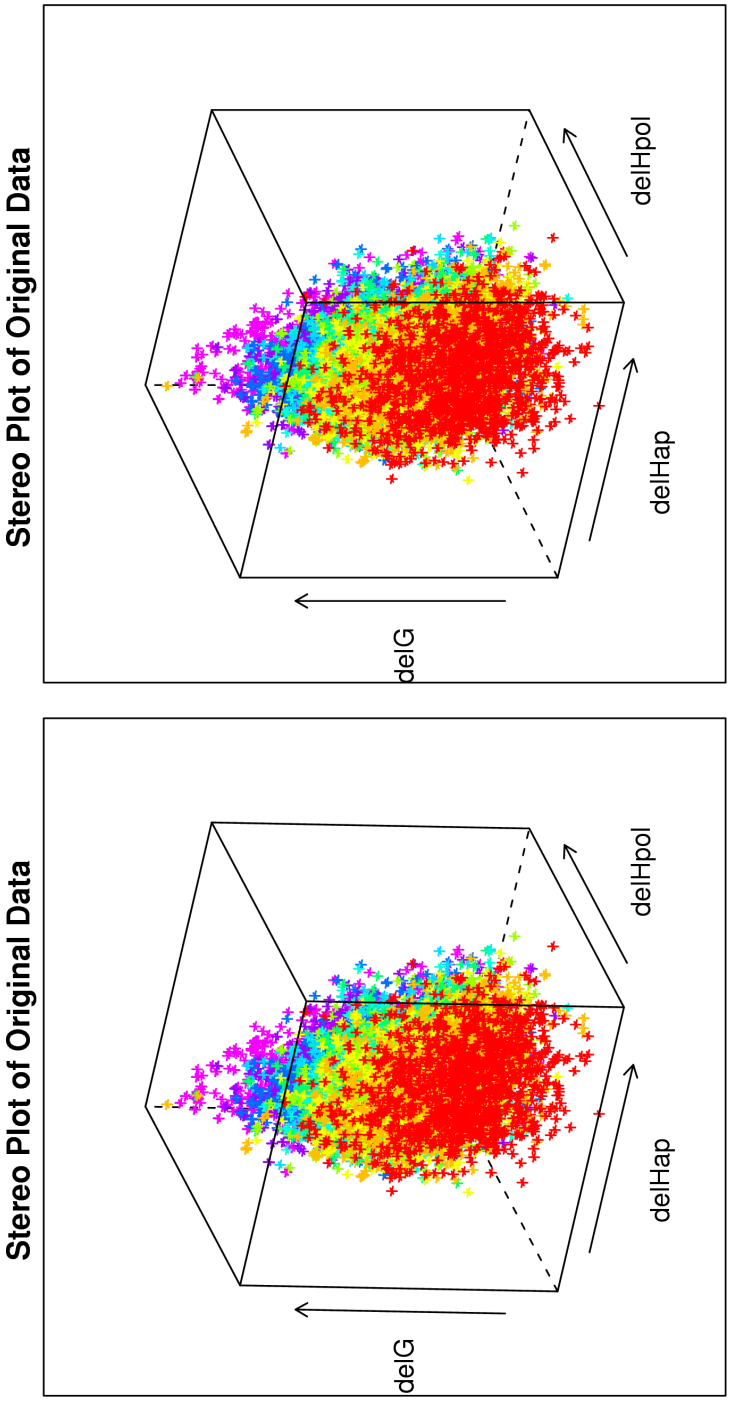


Figure 3.3: The original data is plotted on the original thermodynamic axes. The entropy dimension is plotted as color.

axes are labeled. The entropy dimension, $[\Delta S]$, is presented as the color of the points.

Inspecting Figure 3.3, we can see that the TE space spans all of the four thermodynamic dimensions (again, $[\Delta S]$ is represented as color). Next, the data assume an arrowhead shape in the space. These boundaries on the shape of the space imply some biophysical limitation to the TE space.

3.3 Properties of the Thermodynamic Environment Space

A major underpinning of this project is a biophysical characterization of this TE space. Characterizing the structure of this space will allow us to answer questions that are still unknown about COREX and proteins. For example, why are two different residues in two different secondary structures from different proteins of different lengths positioned near each other in TE space? What can the position in TE space tell us about residue stability? Can we apply this knowledge? The answers to these questions hinge on the interpretation of the TE space, and are answered by this project.

The methods that I carried out to characterize the TE space are presented in the next two chapters. There are two main portions to the analysis. The first set of methods we attempted were statistical and cluster based; these are discussed in Chapter 4. They did not reveal the fine level of detail needed to elucidate the biophysical mechanisms behind the TE space. Yet they are included because they do

provide some information about the TE space. The second set of methods discussed in Chapter 5 revolves around a linear subspace decomposition of the original data. We then linked this lower dimensional space back to our original data, and our original data back to the microscopic property, ΔASA. This allowed us to accurately elucidate the biophysical mechanism behind the organization of the energetic space.

Because the thermodynamic values given to the residues are ensemble-based, we must change our way of thinking about residues in TE space. All residues now become 'faceless' in that they are just place holders; the TE space describes the environment the residue is in, not the residue itself. This is the result of the methods: if we are combinatorially unfolding all possible windows of residues and calculating $[\circ] = \langle \circ \rangle_{\mathcal{F}} - \langle \circ \rangle_{\mathcal{U}}$ for all energetic properties, then the energetics come from the difference of weighted subensembles of very different states where only one window is fixed. Thus, the energetics cannot be the result of only one window or residue. To be clear, each residue does contribute to its environment but does not preponderate the results. This is also the reason why the results from the statical methods for TE space characterization were not enough to make definite statements.

Chapter 4

Statistical Methods for Elucidating the Organization of the Thermodynamic Environment Space

4.1 Statistical Data Mining Tools: Clustering

Introduction With the advent of large, high-dimensional data sets and the desire to discover patterns or structure in the data, statistical data mining tools, like clustering(Braak, 1986; Hand, 1997, 1999), linear and nonlinear multidimensional scaling(Guess and Wilson, 2002; Hill, 1974; Tenenbaum et al., 2000), are timely solutions(Holm and Sander, 1996). These data mining tools are convenient for determining unbiased (with relation to the investigator) relationships in data.

In this project we use single (`HCLUST`) and double agglomerative hierarchi-

cal (HEATMAP) clustering, and partitioning around medoids (PAM; or CLARA for very large data sets). The input to all data mining tools is either the raw data or a distance matrix representing interpoint distances. For very high dimensional data or nonlinear data, the distance matrix can sometimes converted by a topological decomposition technique such as multidimensional scaling(Tenenbaum et al., 2000) or locally linear embedding(Roweis and Saul, 2000). The output of these clustering methods is either a dendrogram representing a phylogeny based upon the similarities of the data, or a label for each point that dictates what family that point belongs to.

It is interesting to note that the location of a data point in a high-dimensional space represented by a distance matrix is determined by that point's similarity to its spatial neighbors and also its dissimilarity with remote neighbors. Consider an example of three proteins, A, B and C. Let A and B have great structural overlap, and C be unrelated to the A and B. Then, A and B will be close together in space because of (1) their high similarity to each other and (2) their high dissimilarity to C. If B and C were somewhat similar, then that change make the distance between A and B larger.

Clustering Methods There are two types of hierarchical clustering, agglomerative and divisive. Agglomerative builds clusters in a bottom-up fashion. It first considers each point as its own cluster and then tries to build successively larger clusters depending on where each point is in space and the algorithm. Divisive clustering is a top-down method. It considers all the points in one large cluster then tries to form smaller and smaller clusters until each point is then it its own cluster.

We use Ward's method of hierarchical clustering(Ward, 1963). Ward's method finds clusters such that their members have the highest amount of similarity. The exact R command we used is shown in Listing4.1.

Source Code 4.1: R Source Code for Clustering Using Ward's Method

```
# Assuming the data is a vector of observations,
output <- hclust( dist(data), method="ward" );

# Or, assuming the data is an NxN matrix of similarities,
output <- hclust( as.dist(data), method="ward" );
```

Ward's method was used to find fold families in our database of thermodynamically defined proteins, as discussed in Section 6.3.

Double hierarchical clustering(Eisen et al., 1998) is carried out using the `heatmap` function in R. The goal of double hierarchical clustering is similar to other clustering methods, but gives both a row and column dendrogram. Thus, we can determine the organization the dimensions that the data are in, as well as the data themselves. The `heatmap` algorithm was used to find amino acid and secondary structure correlations with the structure of our thermodynamic environment space, described below in this chapter.

`PAM`, or partition around medoids, is the last cluster method we used(Kaufman and Rousseeuw, 1990). PAM begins by finding "medoids" or centrally located points in the data. It then assigns each point to in the data set to the closest medoid. An objective function that measures the amount of dissimilarity is calculated. If that objective function can be reduced, by reassigning a point to another cluster, it does so. PAM stops when it's objective function can no longer be minimized. `PAM`[1]

[1]Actually, because the database had nearly 80,000 pieces of data `CLARA` was used. `CLARA` is a

was used to assign a cluster number to each point in our human protein database.

4.2 TE Characterization I: Statistical and Cluster-based TE Subspace Decomposition

4.2.1 Secondary Structure and Amino Acid Type do not Define the TE Space

It was demonstrated that the amino acid sequence, and thus protein fold, can be discriminated from decoys by training a position specific scoring matrix (PSSM) using purely thermodynamic information. Each residue in the thermodynamic database used to train the PSSM, now by definition, lies in our thermodynamic environment space. Furthermore, it was shown that this TE space is not organized by secondary structure rules (Larson and Hilser, 2004). This implies that there is some fundamental biophysical meaning to the organization and location of the data in the TE space.

For example, consider the three points listed in Table 4.1. The table shows one random residue selected from the database and its two closest neighbors. It also shows the amino acid type, the secondary structure, and the ensemble averaged energetics associated with the residues. These three residues were used to help find their fold among decoys. So, there is some information associated with the location in TE space with each residue. The randomly selected amino acid (entry number

relaxed version of PAM that is used for large applications.

Entry No.	PDBID	AA	SS	$[\Delta G]$	$[\Delta H_{\text{apol}}]$	$[\Delta H_{\text{pol}}]$	$[\Delta S]$
4280	1FP5	PRO-381	Coil	-3504	4999	-7391	-3794
4559	1FW1	GLN-121	Helix	-3722	4960	-7412	-4103
16106	1ZXQ	ALA-140	Turn	-3647	4842	-7296	-3470

Table 4.1: Table showing the structural and energetic properties for a random residue (row 1) and its two closest neighbors in TE space. The residue in row 1, was randomly chosen from the database. Notice that the secondary structural (SS) in each of the three residues is different: coil, helix, turn for rows 1–3, respectively. Also notice that the amino acids (AA) are different at each location. All energetic units are in (cal/mol). To illustrate the proximity of the two points to the first, their distances from the first are 222 and 232 (cal/mol), respectively; and, the average distance for all other points from the first point is 8571 (cal/mol). These points are relatively very close. The reason why is explicated later in this Section.

4280) was PRO-381 from the Ige Heavy Chain Epsilon-1 (PDBID: 1FP5) protein. It is in a coil. The closest residue in TE space to PRO-381 is GLN-121 (entry number 4559) from the Glutathione Transferase Zeta (PDBID: 1FW1), shown in row 2. It is in a helix. The last residue is ALA-140 (entry number 16106) from the ICAM-2 protein (PDBID: 1ZXQ). The alanine is in a hydrogen-bonded turn. Methods of typical structural comparison would have no reason to place these three residues nearby each other. So, what biophysical reason is there for these residues to be close together in TE space? What does it mean for a residue to be positioned where it is? Why are other residues far away in TE space?

4.2.2 Cluster-based Decomposition of the TE Space

Clustering is a statistical data mining tool used to assign a label to each point in the data set. Clustering finds families of data in space and attempts to give them

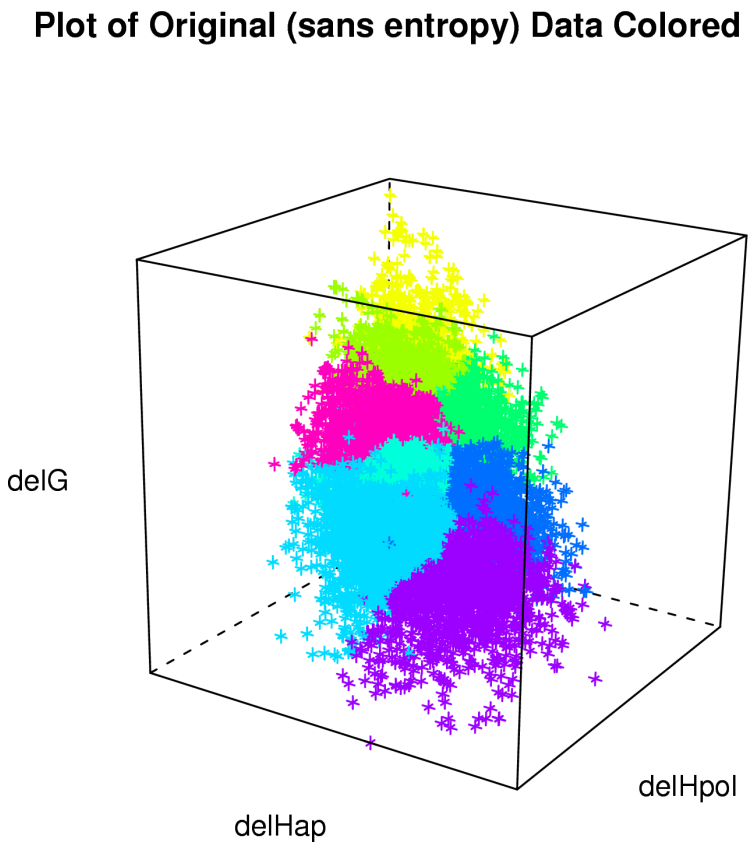


Figure 4.1: Image showing the CLARA clustering of TE space.

the same label. The algorithm is purely mathematical and so any bias from an investigator is removed. There are very many clustering algorithms; Jain is a good review(Jain et al., 1999). Using a method similar to Larson, I have clustered the entire TE space into eight mutually exclusive regions. The entire data set was clustered using the CLARA (Kaufman and Rousseeuw, 1990) algorithm in R. While amino acid type and secondary structure do not define the space, I will show that there is some evidence of non-random propensities for amino acids and secondary

structures to cluster in TE space. The parameters to `CLARA` were: "metric," "manhattan"; "samples," 50; "samp.size," 5000; "medoids.x," `TRUE`; "keep.data," `TRUE`, and "rngR," `FALSE`. Figure 4.1, on page 37, shows the data in the TE space, colored by their assigned environment number, TE1–TE8. The Figure shows TE's 1 and 6 at the extremes of the long axis of the data, while TEs 5 and 8 are the most extreme on the second axis. This decomposition turns each vector into a number 1–8. Thus, we can now treat the energetic protein structure as a sequence. This is exactly what Larson did to discriminate protein folds from decoys (Larson and Hilser, 2004). Given the cluster definitions in TE space, we can gather statistical evidence of the organization of TE space, for example looking for locations in TE space with high densities of secondary structure or particular amino acids.

Amino Acid Propensities in TE Space

Larson et al demonstrated the log-odds probability of an amino acid to be in a certain TE (Larson and Hilser, 2004). That concept is recapitulated here, with my clustering. The Log-Odds probability is the log of a ratio, usually the propensity for some object to be in some environment over the propensity for that environment in relation to all other environments. It's a population-normalized probability. The Equation for the Log-Odds propensity for an amino acid to be in a certain TE is:

$$\text{Log-Odds AA in TE} = \ln\left(\frac{\frac{AA_i \in TE_j}{\sum_{i=1}^{20} TE_i}}{\frac{\sum_{j=1}^{\text{Num. TE}} TE_j}{\sum AA}}\right) .$$

And, the R Source code to calculate such values from our data is shown in Listing 4.2. The algorithm is generalized to calculate the log-odds of each observation (row element) to be in the column element. So, this same code can be used in the following section, just with the appropriate data.

Source Code 4.2: R Source Code for Calculating the Log-Odds from an R Project's data.frame

```
##
## logOdds -- find the logOdds of each observation (row element)
## ======= to be in the column element.
##
require(boot)

logOdds <- function( data, fix.data=FALSE )
{
    totalRes <- sum(data);
    rVal <- matrix( nrow=nrow(data), ncol=ncol(data), dimnames=
        dimnames(data), 0 );

    for ( i in 1:nrow(data))
    {
        for ( j in 1:ncol(data))
        {
            rVal[i,j] <- (data[i,j]/rowSums(data)[i]) / (colSums(
                data)[j]/totalRes);
            if ( fix.data == TRUE )
            {
                if ( rVal[i,j] == 0 )
                {
                    rVal[i,j] <- 1/100;
                }
                else if ( rVal[i,j] == Inf )
                {
                    rVal[i,j] <- 100;
                }
            }
        }
    }

    return(log(rVal));
}
```

Figure 4.2 shows histograms of the propensity for each amino acid to be in each TE. This figure hints at specific pairings of residues in TE space. For example, the small residues PRO and GLY tend to pair together in all TEs. Furthermore, the larger hydrophobic amino acids PHE, TYR and TRP tend to cluster together, and with opposite tendencies to the small residues. TE3 doesn't appear to have much propensity at all (except for CYS). TE8 appears to favor charged residues, CYS, ASN, ASP, SER, GLU and disfavor PHE, TYR, and TRP. The following general rules can be created from the data:

- FYW, the aromatics, do not highly populate the low stability environments TEs1, 2 and 8,
- PG share the same propensities for TEs,
- the aromatics and VIL pair together except where the aromatics don't show any propensity for or against populating TE7,
- CND pair together and are highly populated in TEs 2, 4 and 8, and are rarely found in TE5,
- C diverges from ND in that it is highly populated in TE6, but not TE7, where ND don't share that propensity. This may be due to the dual nature of cysteine.

Secondary Structural Elements in TE Space

Next, we investigated the propensity for secondary structures in TE space. Figure 4.3 shows the Log-Odds propensity for secondary structural unit to be in certain

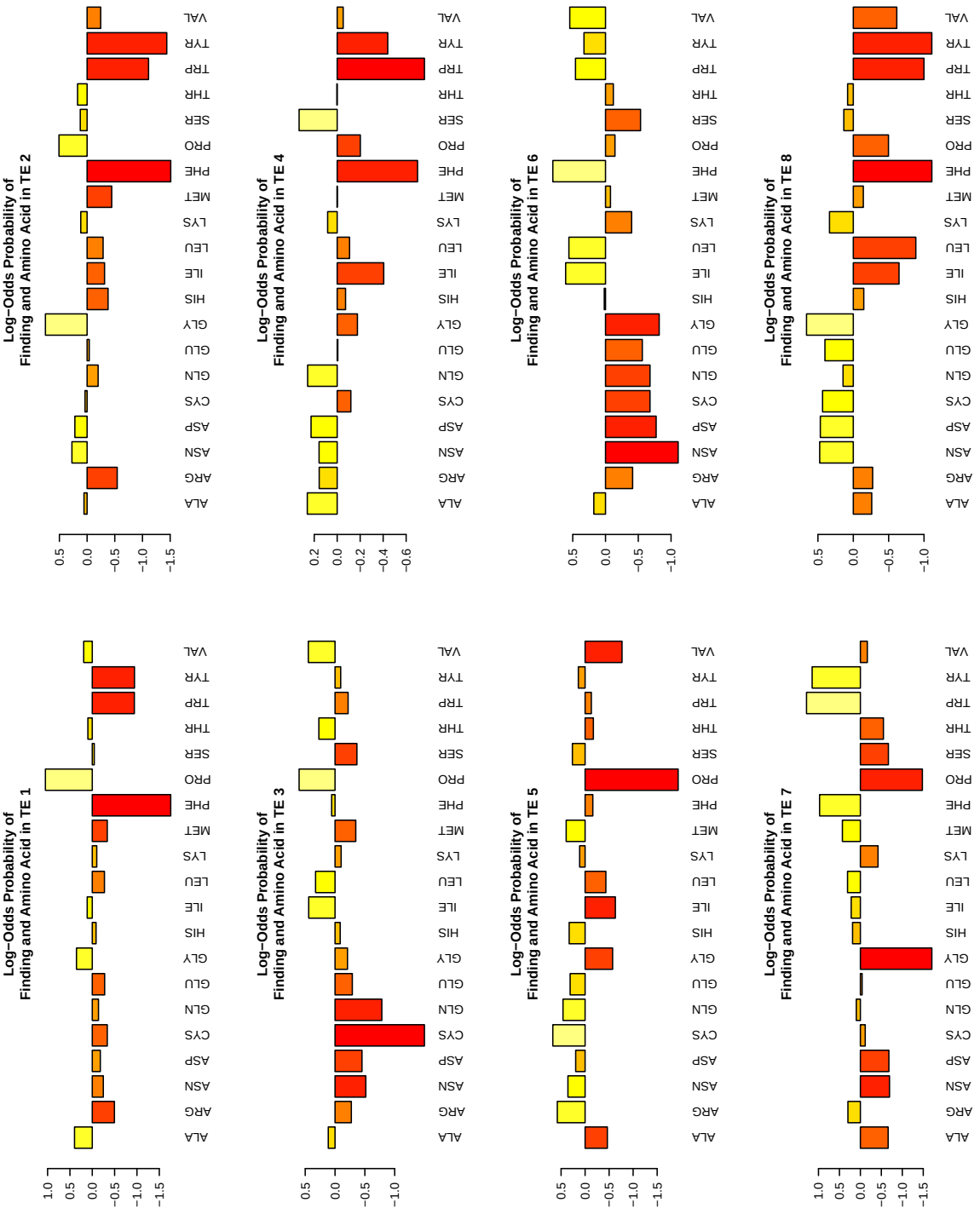


Figure 4.2: Log-Odds probability of finding each amino acid in TEs 1–8.

TEs. Coils and Turns show a moderate or high propensity for TEs 1–3 and 8, and a low propensity for TEs 5–7. The helical regions interestingly do not always show the same propensities in the same environments. That is, the 310 Helix and the Alpha Helix, in TEs 3–8 show differing signs in magnitude of log-odds probability. Beta sheet shows a positive propensity for TEs 3, 6 and 7.

In general, these magnitudes of the values of the log-odds plots in incapable of providing definite statements about propensities. However these results help give support to the final characterization of TE space.

Amino Acid Type and Secondary Structure Combined in TE Space

Continuing the search for the biophysical meaning of the TE space, we then considered the log-odds joint probability of an amino acid and a secondary structural element to be in a TE. Deciphering this will indicate how the energetic space is organized when we consider each amino acid in each secondary structure type. Because the data are large (the matrix is 127×8), they are summarized through hierarchical clustering. Figure 4.4 shows the dendrogram result from hierarchically clustering the joint-probability log-odds ratio of finding an amino acid–SS pair in a given TE. The major branches are labeled with their majority preference. This shows us how secondary structures pair with amino acids in TE space.

These results can be applied to fine-grained mutation studies. We have statistical relationships bewtwen amino acids, TEs and secondary structures. We also know about the joint probability of amino acids and secondary structurs to cluster in certain TEs. This information can be combined to more accurately make residue

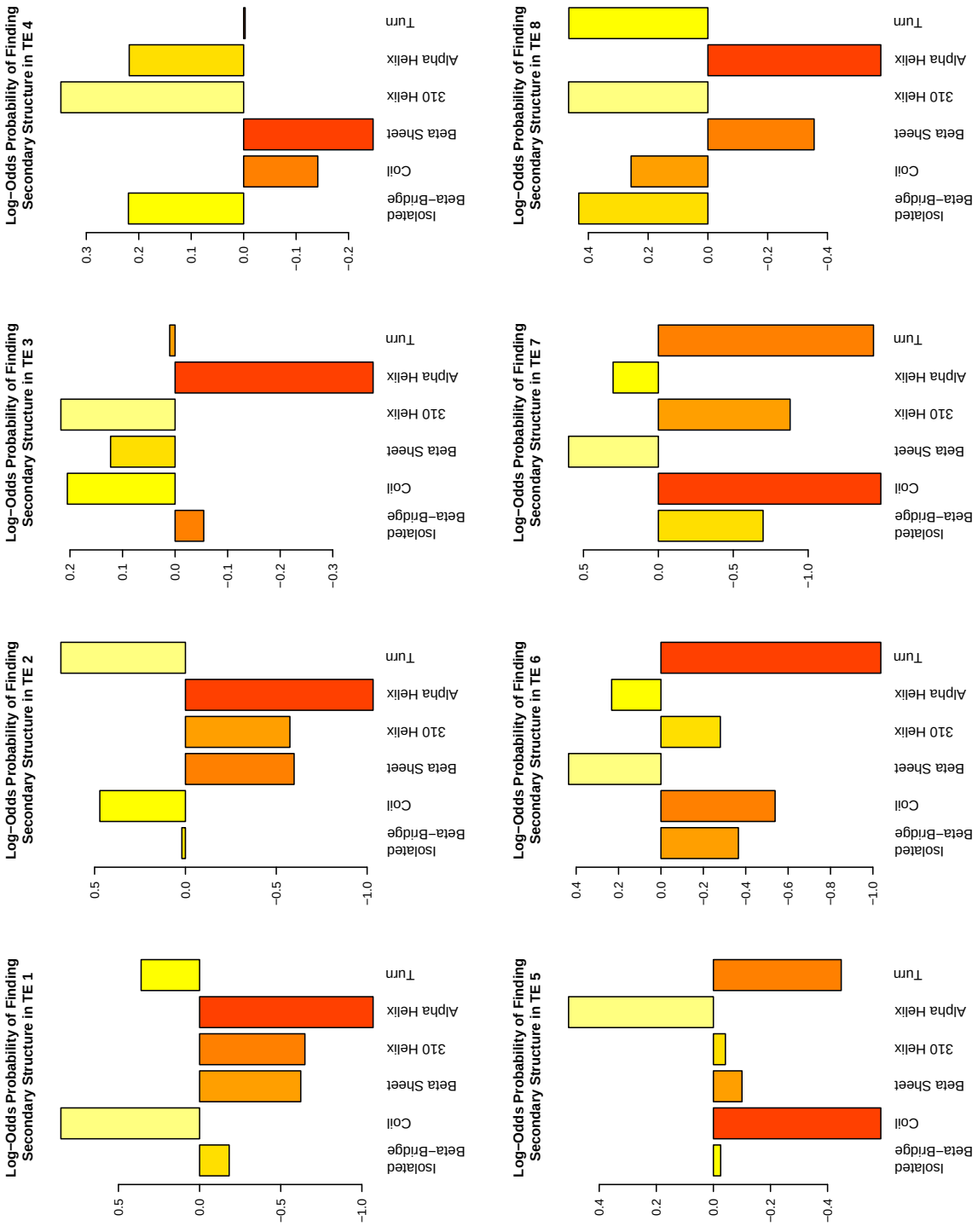


Figure 4.3: Log-Odds probability of finding each secondary structural element in TEs 1–8.

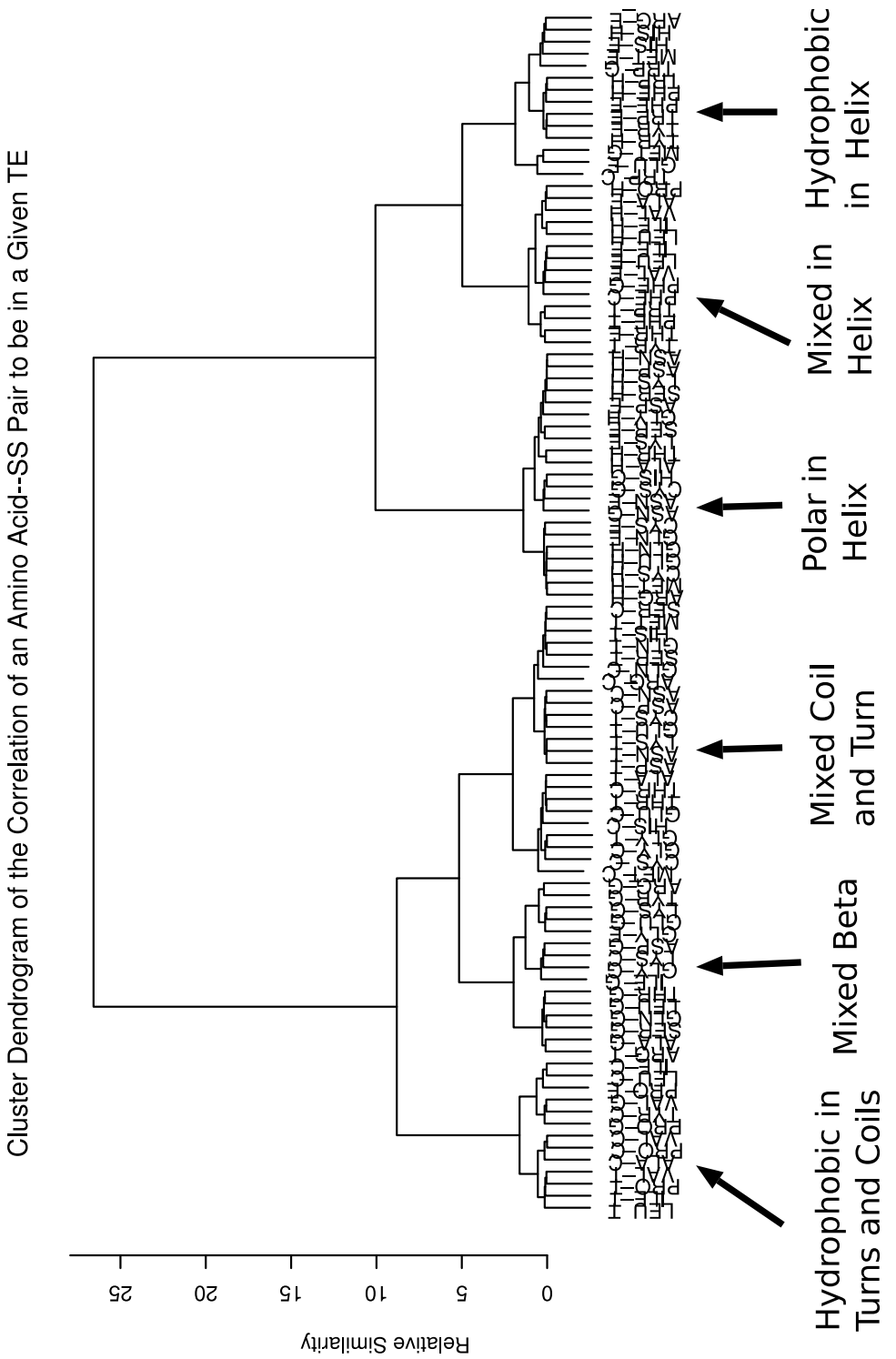


Figure 4.4: Figure showing the clustering of the log-odds joint-probability of finding an amino acid–SS pair to be in a TE.

mutation predictions.

Chapter 5

Elucidating the Organization of the Thermodynamic Environment Space through Linear Algebraic Methods

5.1 TE Characterization II: Principal Component Analysis (PCA) of the TE Space Reveals Biophysical Organization

Preliminaries Analyzing the clustering of the TE space provided us with a good feel for its organization. There are regions of low and high stability, low and high $\Delta ASA_{\text{apol}}/\Delta ASA_{\text{pol}}$, and high and low entropy. We even know of the existence of non-random propensities for amino acid types to cluster in certain TEs, but we still

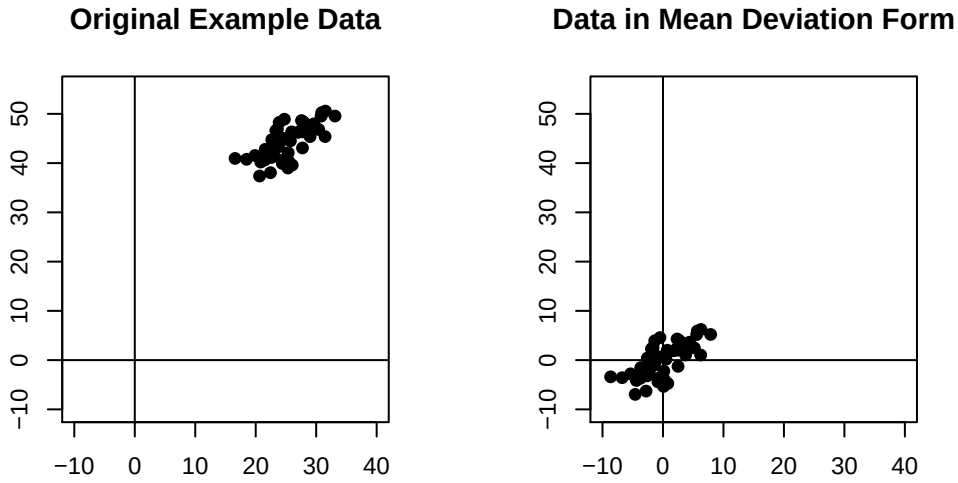


Figure 5.1: Figure illustrating the MDF procedure.

don't know why. PCA of the TE space provides these answers. But, before discussing PCA, it may help to define the mean deviation form and sample covariance matrix of a data set. All other terms used below are explained above or are common to the general field of linear algebra and may be reviewed elsewhere(Lay, 2003).

Definition 1 (Mean Deviation Form (MDF))**.** *The* ***mean deviation form*** *of a vector of observations* X *is*

$$\hat{X} = X - \frac{1}{N}\sum X \, .$$

$\hat{X}$ *is then the vector set of values representing the the original points' deviation from the mean, hence the name. The MDF procedure is illustrated by the change of location of the data in Figure 5.1.*

Definition 2 (Sample Covariance Matrix)**.** *The* ***sample covariance matrix*** *of a data*

set of N observations in MDF is,

$$S = \frac{1}{N-1}\hat{X}\hat{X}^{\perp} .$$

Sample covariance matrices are square, symmetric, positive semi-definite. This means they can be orthogonally diagonalized, as discussed below. The entries on the diagonal are the variance for that given dimension. If an off diagonal element, $S_{i,j}$ is 0 the two dimensions i and j are "uncorrelated."

5.1.1 PCA

PCA is a linear method used for subspace decomposition. It takes an $p \times n$ matrix of observations in mean deviation form and finds a special $p \times p$ change of basis matrix. The particular change of basis matrix PCA finds, orthogonally diagonalizes the sample covariance matrix of the data. Given a matrix A of n observations in p dimensions, already in MDF, associate with it another matrix S its $p \times p$ sample covariance matrix. PCA finds the $p \times p$ change of basis matrix P,

$$A = PB ,$$

such that the dimensions of B are uncorrelated and of decreasing variance. This orthogonally diagonalizes S. S is symmetric, positive semi-definite, so can be orthogonally diagonalized by its eigenvectors, as $S = PDP^{\perp}$. Let D be the diagonal matrix with entries $D_{ii} = \lambda_i$ the eigenvalues of S ordered so $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$. Then, let P be the matrix of unit eigenvectors of S in order of their correspond-

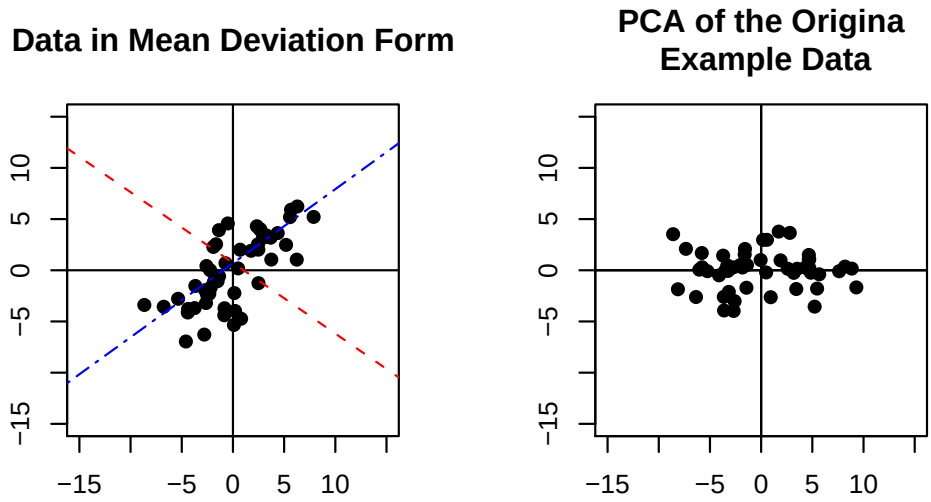


Figure 5.2: This example shows the result of PCA on an example data set. The image at the left shows the original MDF data. The dash-dot blue line corresponds to the axis of greatest variance for the example data set. The dashed red line indicates the axis of second most variance. The image at the right shows the data projected onto these axes.

ing eigenvalues. Then, $D = P^{\perp}SP$. The unit eigenvectors of S are called the "principal components." The first principal component corresponds to the largest eigenvalue, the second principal component to the second largest eigenvalue, and so on. Each eigenvalue corresponds to the variance of the given dimension of data in B.

PCA is useful for a few reasons. First, it can reduce the dimensionality of the data. Use only the first $d < p$ eigenvectors and this makes B an n by d matrix. This reduces the data by cutting off the projected dimension(s) with lowest variance. An example data set was created for illustrative purposes and subjected to PCA. This example data set is shown in Figure 5.2 on page 49.

Nonlinear methods for subspace decomposition were also tested. IsoMap (Tenenbaum et al., 2000) and Local Linear Embedding (LLE) (Roweis and Saul, 2000) showed no benefit over PCA. In fact with linear data, IsoMap and LLE decompose into PCA.

5.1.2 PCA of the TE Data

PCA was performed on the ensemble-averaged data. First, the data set was put into MDF. Next, the covariance matrix for the data was calculated. Then, the eigenvector decomposition for the covariance matrix was calculated. Finally, the data were projected onto the new orthonormal basis by multiplying the original data by the eigenvectors of S. All of this is neatly wrapped into the R function `prcomp`. The exact call to `prcomp` is shown in Listing 5.1.

	PC1	PC2	PC3	PC4
$[\Delta G]$	-0.55	0.15	0.59	-0.57
$[\Delta H_{\text{ap}}]$	0.65	0.69	0.22	-0.23
$[\Delta H_{\text{pol}}]$	-0.51	0.70	-0.23	0.44
$T\,[\Delta S_{\text{conf}}]$	-0.09	0.11	-0.74	-0.66

Table 5.1: Principal components of the original thermodynamic environment data. Thus, the axis with most variance is $PC1 = -0.47\,[\Delta G] + 0.65\,[\Delta H_{\text{apol}}] + -0.51\,[\Delta H_{\text{pol}}] - 0.09\,[\Delta S]$.

Source Code 5.1: R Command for Calculating the Principal Component Analysis of the TE Data

```
all.pr <- prcomp( allData[c("delG", "delHap", "delHpol", "dS"),],
    scale=F, center=T, retx=T);
```

5.1.3 Results

Eigenvalues

The first three principal components explained 99.92% of the variance. There is a sharp decrease in the magnitude of the eigenvalues corresponding to the eigenvectors; this indicates a nonrandom signal in the data, and further supports the use of PCA as a valid method of dimensionality reduction in our application. The eigenvalues are plotted in Figure 5.3. The proportion of the variance explained by the eigenvalues is: 75.24%, 22.03%, 2.65% and 0.08%, for PC1–4, respectively. Thus PC1 alone explains the majority of the variance in the data set.

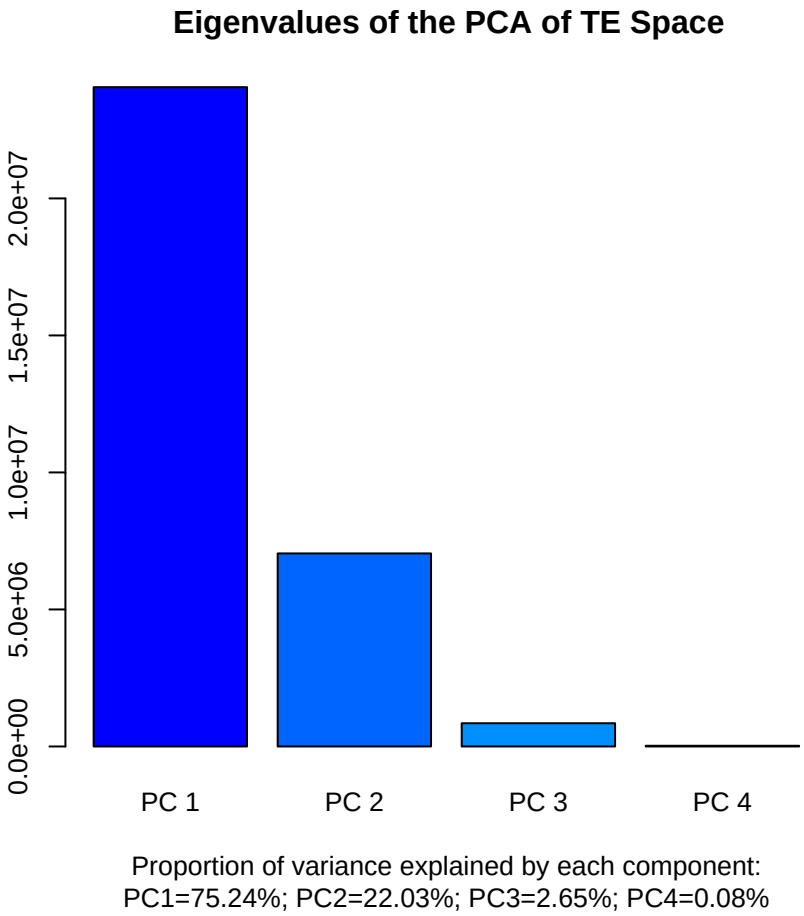


Figure 5.3: Magnitude of eigenvalues from the PCA of the TE space. This Figure is included to support our use of PCA as a valid linear method of subspace decomposition. Random and nonlinear data matrices tend to have eigenvalues of nearly equal value, hence our data is not random and can be explained well by PCA.

Eigenvectors

The rotation matrix returned from the PCA calculation indicates the linear combination of energetic terms that define the principal axes. The rotation matrix is shown in Table 5.1. This provides us with the information on how the natural dispersion of points in TE space is related to the energetic parameters, $[\Delta G]$, $[\Delta H_{\text{apol}}]$, $[\Delta H_{\text{pol}}]$, and $[\Delta S]$.

Table 5.1 requires a few comments. First, by considering the first PC (column 1) we can calculate the change in the four energetic parameters required for change of equal distance in PC1. For example, a change in 1000 units along the axis defined by PC1 corresponds to traveling through the original energetic space by

$$\begin{aligned} 1000 \cdot PC1 &= 1000\,(-0.55\,[\Delta G] + 0.65\,[\Delta H_{apol}] - 0.51\,[\Delta H_{pol}] - 0.09\,[\Delta S]) \\ &= -550\,[\Delta G] + 650\,[\Delta H_{apol}] - 510\,[\Delta H_{pol}] - 90\,[\Delta S]\ . \end{aligned}$$

Therefore, the energetic difference explained by a change of 1000 units exactly incident with PC1 is $-550\ (cal/mol)$ along $[\Delta G]$, $650\ (cal/mol)$ along $[\Delta H_{apol}]$, $-510\ (cal/mol)$ along $[\Delta H_{pol}]$, and $-90\ (cal/mol)$ along $[\Delta S]$. This is a general rule for analyzing changes in location around TE space: it maps the principal axes to their original energetic coordinates. Notice that a one unit change along PC2 requires a very small change along the stability axis ($[\Delta G] \approx 0.15(cal/mol)$).

Relation of Cluster Centers to Principal Components Analysis It has been shown recently (Zha et al., 2002) that the cluster center locations determined

by the popular K-means algorithm with relaxed constraints are given by the eigen-decomposition of the covariance matrix and thus are directly related to the principal components. This explains the organization of the TE cluster centers within our TE space and their relation to the principal components. As mentioned previously, the clustered data were used to discriminate folds (Larson and Hilser, 2004); now we know why the cluster centers are organized as they are.

Equating the Change in Accessible Surface Area to the Principal Component Axes

Using our rotation matrix from the PCA of the TE data, we can calculate what ensemble-average energetic changes are required to mirror a concomitant change along a given PC. From that, we can then calculate the average change in ΔASA from the unfolding event for that given residue. More specificly, using Equations 2.18 and 2.19 on page 16, and given the values from the principal components, shown in Table 5.1, we can calculate the change in apolar/polar surface area with a given change in stability on each axis. The data for this conversion, and the ratio of apolar to polar ensemble-averaged enthalpy are shown in Table 5.2.

This is a valid transformation because the phenomenological effect of surface area exposure relative to energetics is additive (Freire and Murphy, 1991; Xie and Freire, 1994).

	PC1	PC2	PC3	PC4
$[\Delta ASA_{\text{apolar}}]$ (Å^2)	-0.027	-0.028	-0.009	-0.009
$[\Delta ASA_{\text{polar}}]$ (Å^2)	-0.013	+0.017	-0.006	-0.011
Ratio $\frac{[\Delta ASA_{\text{apolar}}]}{[\Delta ASA_{\text{polar}}]}$	2.15:1	-1.64:1	1.66:1	0.86:1

Table 5.2: Table showing the correspondence between principal component axes and the change in accessible surface area. Looking down each column we can calculate the average ΔASA required for a one unit change along the principal component. For example, a one unit change along PC1 requires a change of 0.027 Å^2 apolar and 0.013 Å^2 polar accessible surface area in the $\mathcal{F} \rightarrow \mathcal{U}$ transition.

5.2 Biophysical Interpretation of the Characterization of the Thermodynamic Environment Space

The above analysis in this chapter details the steps we underwent to mathematically characterize the structure of the thermodynamic space. Here, I shall provide interpretation of the results and provide examples supporting the analysis.

5.2.1 Biophysical Interpretation of PC1

Recall that Table 5.2 shows the change, and type of change in ASA required for an accompanying change in energetics. Because a positive change incident with the first principal component requires a decrease in the average ASA of unfolding of 0.027 Å^2 of apolar surface area and 0.013 Å^2 of polar surface area and a negative change incident with the same axis will just change the signs of the ASA changes, we can state that the biophysical interpretation of PC1 is the simultaneous increase

or simultaneous decrease in the amount of ASA presented to solvent upon unfolding. Furthermore, because we can look at these ASA changes as ensemble-averaged residue-level changes for the $\mathcal{F} \rightarrow \mathcal{U}$ transition, negative values indicate a larger change in surface area for the $\mathcal{U}$ subensemble than for $\mathcal{F}$. Stated simply, residues with higher values on PC1 are more stable because their unfolded subensembles have a very low probability due to the exposure of large amounts of surface area at a ratio of 2:1 apolar to polar.

We can also make statements about stability from this as well because the ASA changes are related to the $[\Delta G]$. Looking at the table, a protein can be stabilized (negative ΔG) by conservatively exposing both types of accessible surface area at the approximate 2 to 1 ratio. This includes the area of direct unfolding and the complementary surface area as well.

To corroborate and illustrate these findings, Figures 5.4 and 5.5 plot the data spanned across PC1. The first figure shows the data colored by the total surface area exposed ($\Delta ASA_{\text{apol}} + \Delta ASA_{\text{pol}}$) against PC1. The second plot shows the data sorted by rank order along PC1, then colored according to the total amount of surface area change.

Principal Axis Variance Defines Thermodynamic Structural Features

Because the first principal component has the largest variance associated with it we will see the largest changes incident with PC1. Thus, the difference vectors associated with each contiguous residue pair will have, generally, the smallest angles away from PC1. On average, PC2 will have difference vectors second most incident to itself. And this will make it rare to find contiguous residue differences that

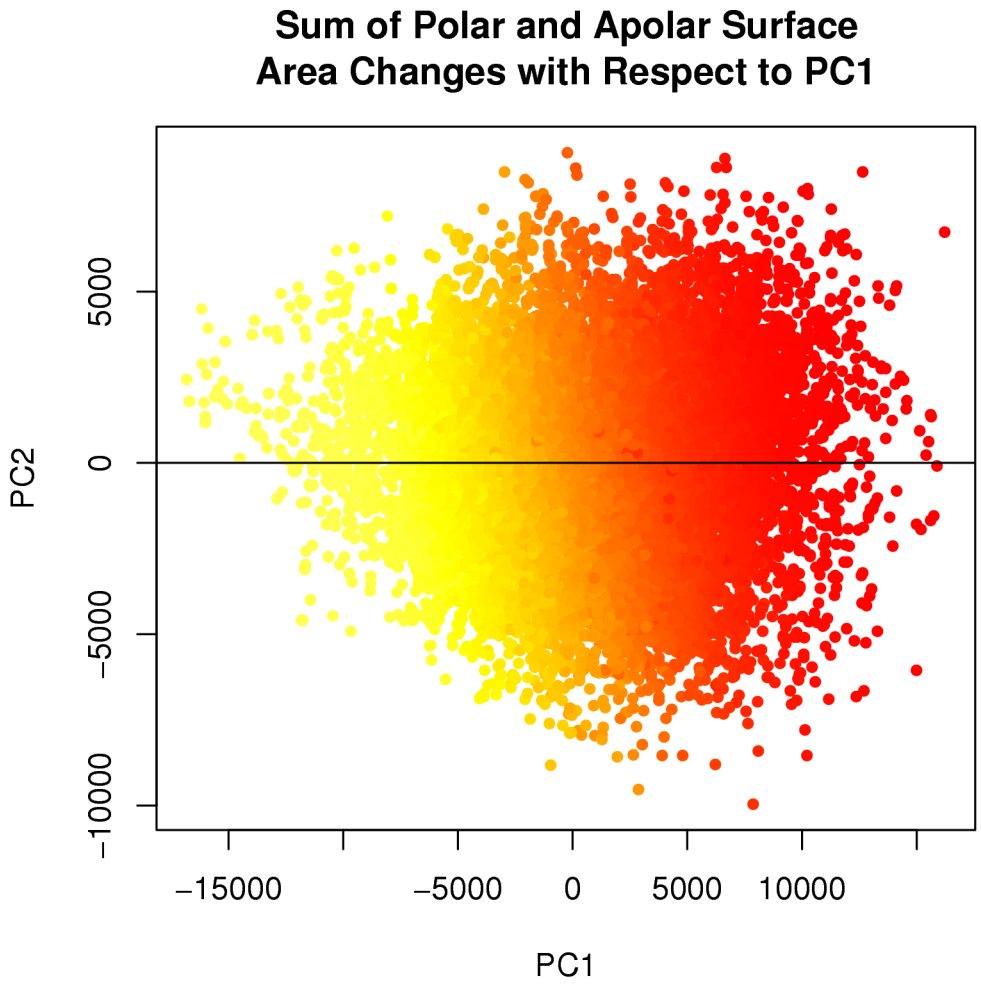


Figure 5.4: Figure Showing the Total Change in Accessible Surface Area Along PC1. The data are plotted onto PC1 and PC2 and colored by their total ΔASA value.

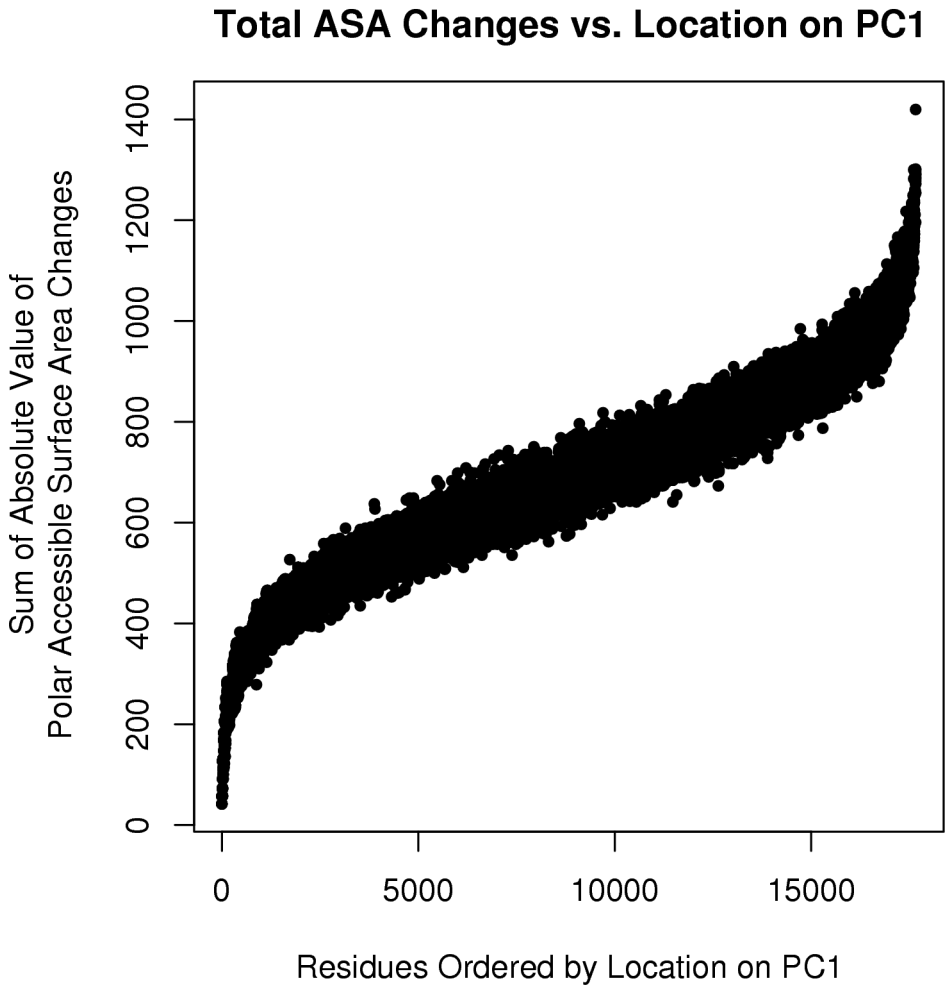


Figure 5.5: Total Change in Accessible Surface Area with Respect to PC1. This image shows the data rank ordered along PC1. The data were then colored by their total change in accessible surface area. This may be thought of as a side view of the previous image.

are nearly incident with PC3.

Summarizing PC1: changes in a positive direction exactly parallel with PC1 expose more surface area, while negative changes unfold less at the familiar 2 to 1, apolar to polar ratio. This 2 to 1 ratio further indicates that the most dominant (because it is along PC1) unfolding event unfolds nearly twice as much apolar surface area that polar surface area.

5.2.2 Biophysical Interpretation of PC2

Going back to Table 5.2 we can investigate the biophysical meaning behind PC2 in the same vein. The table shows that for a one unit change along PC2 the $\mathcal{F} \rightarrow \mathcal{U}$ ΔASAs are -0.028 Å^2 of apolar surface area and 0.017 Å^2 of polar surface area. This result shows that proteins are slightly de/stabilized ($\Delta\Delta G \approx 0.15\ (cal/mol)$) through a one unit change in PC2. Also, interestingly, a positive change incident with PC2 requires the exposure of less apolar surface area while exposing more polar surface area. A negative change along PC2 is then just the reflection of those results.

Because the stability coefficient of PC2 is relatively small, changes along PC2 have a smaller effect on stability than do changes along PC1. This can be seen as a mechanistic way for a protein to change polarity of a region (or set of microstates) without adversely affecting stability. Thus, PC2 is then more directly related to the type of surface area being exposed—not necessarily the quantity.

PC2 combined with PC1 gives all possible combinations of exposing more or less of polar or apolar surface area.

5.2.3 Biophysical Interpretation of PC3

The table for ASAs shows a change of -0.009 Å^2 apol and -0.006Å^2 pol for a one unit change along PC3. These values are much smaller that those required for ASA changes along PCs 1 and 2. This indicates the major energetic component is not due to solvation enthalpy changes due to unfolding, but something else. Looking back at Table 5.1, we see that in column 3, that the entropy change for PC3 is three to five times larger than it is for PCs 1 and 2. So, a very small change in accessible surface area but a large change in entropy and moderate change in stability is what defines PC3. The entropy is 55% of the magnitude of PC3; the entropy and stability change comprise 90% of the magnitude of PC3. This explains why when one looks at an image of the TE space colored by $[\Delta S]$ the axis is very close to parallel with entropy as evinced by the change of color.

5.2.4 Biophysical Interpretation of PC4

The amount of variance of the data that is explained by PC4 is so small, that we consider it insignificant. This is the purpose of SVD/PCA analysis on multidimensional data—to remove the insignificant dimensions such that the amount of variance (information) explained still remains high. And, as we've seen, with more than 99.92% of the variance explained by the first three axes, this remains a valid procedure.

Hence, the fourth principal component can be seen as rank one noise. Rank one noise, or noise in general from PCA/SVD is discussed at length in Methods in Enzymology(DeSa and Matheson, 2004; Henry and Hofrichter, 1992) and else-

where (Lay, 2003).

5.2.5 Center of Mass

We carried out the PCA such that the data were put into MDF. This provides the views of maximum variance around the center of mass, not the origin. The residue-level ensemble-average center of mass was determined to be: $[\Delta G] = -8137.75\,(cal/mol)$, $[\Delta H_{\text{apol}}] = 9535.74\,(cal/mol)$, $[\Delta H_{\text{pol}}] = -11724.89\,(cal/mol)$, and $[\Delta S] = -4562.23\,(cal/mol)$. In our analysis, this point represents the average unfolding event around which all other events vary.

5.3 Analysis of Vastly Different Residues Along the Principal Components

Introduction The next two sections are used to provide examples to illustrate our findings. Here, we will locate residues very far apart on each of the PCs. We will then contrast their structural and energetic properties. If our analysis of the biophysical meaning of the PCs is correct, then we should see that reflected in the proteins.

To keep the ideas clear it is important to introduce the methodology. First, the ensemble average values, $[\Delta G]$ for example, is already a difference vector: $[\Delta G] = \langle \Delta G \rangle_{\mathcal{F}} - \langle \Delta G \rangle_{\mathcal{U}}$. Next, calculating the vector from vector $\hat{a}$ to $\hat{b}$ is $\hat{\delta} = \hat{b} - \hat{a}$. Let us take the example of calculating the difference in energetics changes in going from an unstable region to a stable region. Assume that

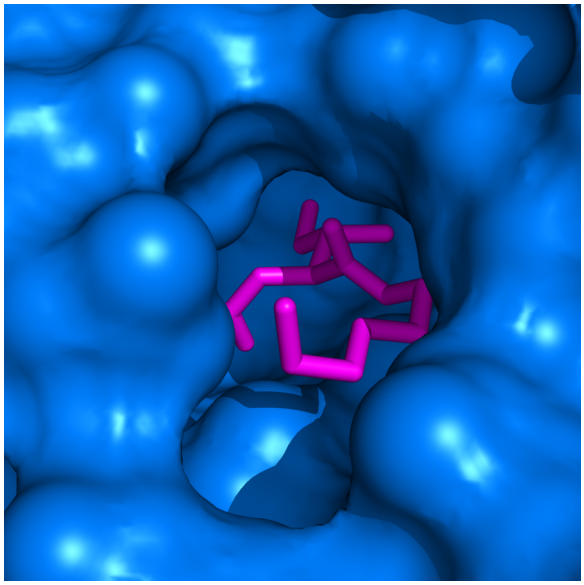
$[\Delta G]_{\text{stable}} = -20,000\ (cal/mol)$ is the ensemble average stability for the stable residue. Likewise, let $[\Delta G]_{\text{unstable}} = -1,000\ (cal/mol)$ for the relatively unstable residue. Then, the vector from unstable to stable—the vector that explains what we need to do to traverse to the stable environment—is, $[\Delta G]_{\text{stable}} - [\Delta G]_{\text{unstable}} = -20,000 - -1,000 = -19,000$. Therefore, the less stable residue needs to lower its $[\Delta G]$ by $19,000\ (cal/mol)$ to achieve the higher stability. Confusion may arise because (1) high stability is negative $[\Delta G]$ and (2) we are discussing the differences of difference vectors.

Principal Component 1

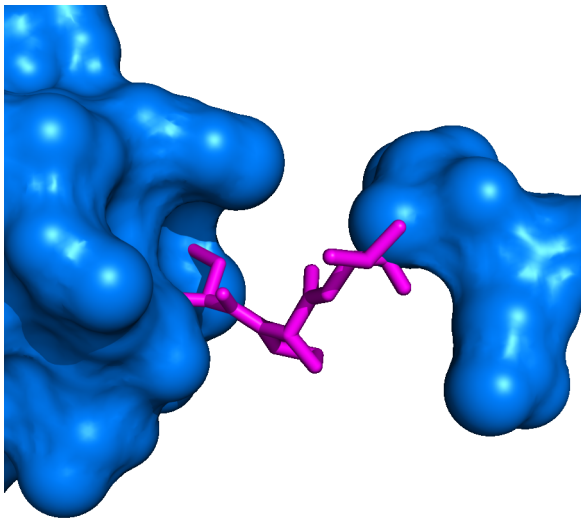
Starting with PC1 we chose two residues, one from near the maximum positive end of PC1 and one near the maximum negative end of PC1. The two residues chosen are depicted in the following table, and shown in Figure 5.6 in their respective structures.

Index	PROT	AA	RESI	SS	TE	$[\Delta G]$	$[\Delta H_{\text{ap}}]$	$[\Delta H_{\text{pol}}]$	$[\Delta S]$
9585	1JHJ	ILE	156	E	7	-15210	25097	-15500	-7153
7925	1I71	PRO	79	C	1	1346	172	-1413	-2350

Analyzing the accessible surface areas from the differences in the apolar and polar enthalpies between the two residues, we see that in going from the `PRO` to the `ILE`, there is a increased stability change of $16.5\ (kcal/mol)$, which comes from a difference of exposing 1030 Å^2 and 348 Å^2 more of apolar and polar surface area, respectively. Biophysically, it is unfavorable to expose such large quantities of surface area to solvent upon unfolding. Thus, the probabilities of the unfolded



(a)



(b)

Figure 5.6: Image Showing the Different Residues Chosen to Maximize Their Distance Along PCA1. The image at top (a) shows the ILE-156 and its unfolding neighbors (pink colored sticks); the residue at maximum PC1. Notice that upon removal of these residues there would be a very large amount of change in solvent-accessible surface area. Water could now fill the entire cavity left by the removal. The bottom image (b) shows the residue and its unfolding neighbors with lowest PC1 value. Notice that there is no cavity left to fill, upon removal.

subensemble are very low for ILE-156. Therefore, the energetics from the folded subensemble dominate and the residue is seen as "stable". Likewise, the unfolded subensemble of PRO-79 unfolds much less surface area which is more favorable. Therefore, the unfolded subensemble is more stable which means the residue is more likely to be in an unfolded state—unstable—than the ILE-156.

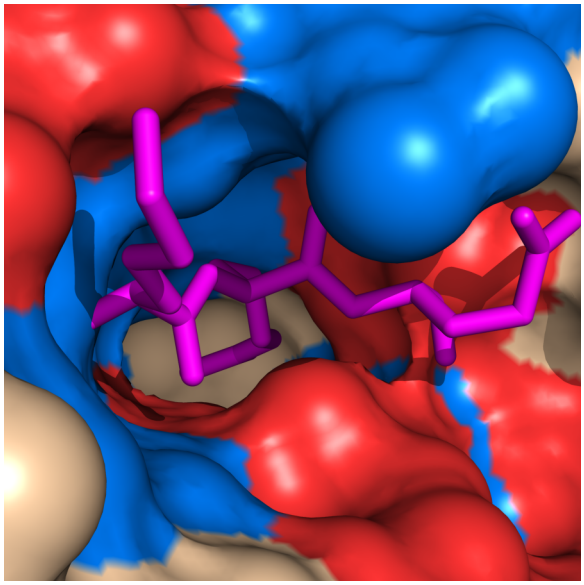
Principal Component 2

I selected the two residues shown in the table below. They were both far apart with respect to PC2. Notice the large differences in $[\Delta H_{\text{apol}}]$ and $[\Delta H_{\text{pol}}]$. This indicates a large difference in the type of surface area being exposed. This seems reasonable considering the differences in the environments as shown in Figure 5.7. The difference in apolar surface area observed from the transition from the ARG-471 to LEU-180 is 321 Å^2 more exposed. The polar surface area change is -433 Å^2. Thus, as we expect, the unfolding environments are vastly different with regard to type of surface area exposure when taking large differences along PC2.

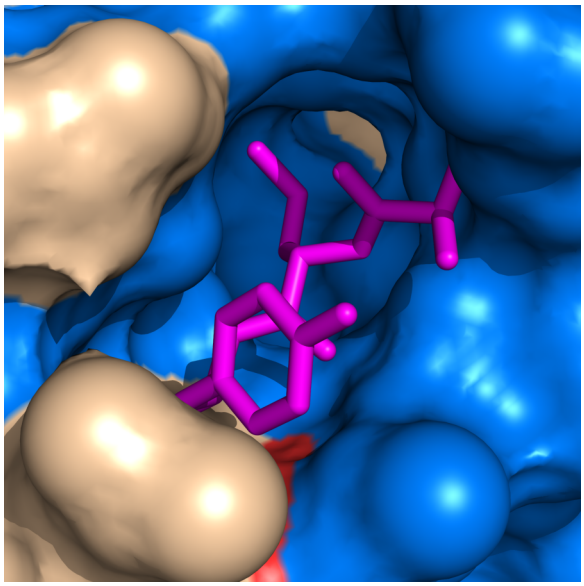
Index	PROT	AA	RESI	SS	TE	$[\Delta G]$	$[\Delta H_{\text{ap}}]$	$[\Delta H_{\text{pol}}]$	$[\Delta S]$
8506	1IFR	ARG	471	E	5	-14090	7782	-22628	-6513
6151	1GSM	LEU	180	E	3	-6805	15559	-5081	-4115

Principal Component 3

We have chosen two points very far apart on PC3. The points are not necessarily incident with PC3. This is because it is more difficult to find distal residues directly parallel with PC3 as the variance of PC3 is much smaller than for PC1 or PC2. Their



(a)



(b)

Figure 5.7: Example residues chosen based on their (relatively large) distance from each other on PCA2. The top image (a) shows the ARG-471 and its unfolding neighbors and pink colored sticks. Notice the relatively large amount of polar surface area (colored red) nearby the ARG-471 residue. The bottom image (b) shows the much more negative PC2 residue LEU-180 and its unfolding neighbors. Notice the surrounding surface area is almost completely apolar (colored blue).

data are summarized in the following table. Notice that the entropy and stability values vary inversely as indicated in the table of PC axes.

Index	PROT	AA	RESI	SS	TE	$[\Delta G]$	$[\Delta H_{\text{ap}}]$	$[\Delta H_{\text{pol}}]$	$[\Delta S]$
45	1A17	ILE	63	H	3	-11096	7773	-10180	-374
16971	2ILK	ILE	147	H	5	-6263	12835	-14159	-9555

5.4 Employing a "Thermodynamic Cycle" to Further Support the Biophysical Characterization of TE Space

Introduction The point of this analysis is to further support our assertions of the characterization of TE space. We will do so by making a cycle around the thermodynamic space incident with an embedded plane in the span of PC1 and PC2 and investigating other possible underlying mechanisms that go on during the transitions.

We have defined four centroids in TE space. Each centroid is plus or minus 8700 units along PC1, and plus or minus 2800 units off of PC2, from the origin. These points are far enough from each other to produce significant observable biophysical differences while not being too far away as to have no close neighbors.

Symbolically,

$$\begin{aligned}
\text{Centroid 1} &= +PC1 \times 8700 \text{ units} + PC2 \times 2800 \text{ units} \\
\text{Centroid 2} &= +PC1 \times 8700 \text{ units} - PC2 \times 2800 \text{ units} \\
\text{Centroid 3} &= -PC1 \times 8700 \text{ units} - PC2 \times 2800 \text{ units} \\
\text{Centroid 4} &= -PC1 \times 8700 \text{ units} + PC2 \times 2800 \text{ units} .
\end{aligned}$$

This creates a plane embedded in the span of PC1 and PC2. Because our eigenvector basis from PCA is clearly a subspace of $\mathbb{R}^3$, and due to the orthogonal decomposition theorem(Lay, 2003), which states that any vector in W a subspace of $\mathbb{R}^n$ can be decomposed into the sum of its orthogonal parts, we can write any change in this plane as the sum of the change along PC1 and the sum of the change along PC2—because PCA calculates these as orthogonal vectors. In fact, we can represent any vector in our PCA space as a combination of PC1, PC2 and PC3 which allows us to separate the biophysical contributions across changes associated with each axis.

The goal is now to calculate the difference along each axis and quantify the change in energetics, relate that to ΔASA and ensure that our assertions from the previous section are sound.

Figure 5.8 shows the plane embedded in the span of PC1 and PC2. The information for the four residues closest to the centroids are shown in Table 5.3. As this diagram shows,

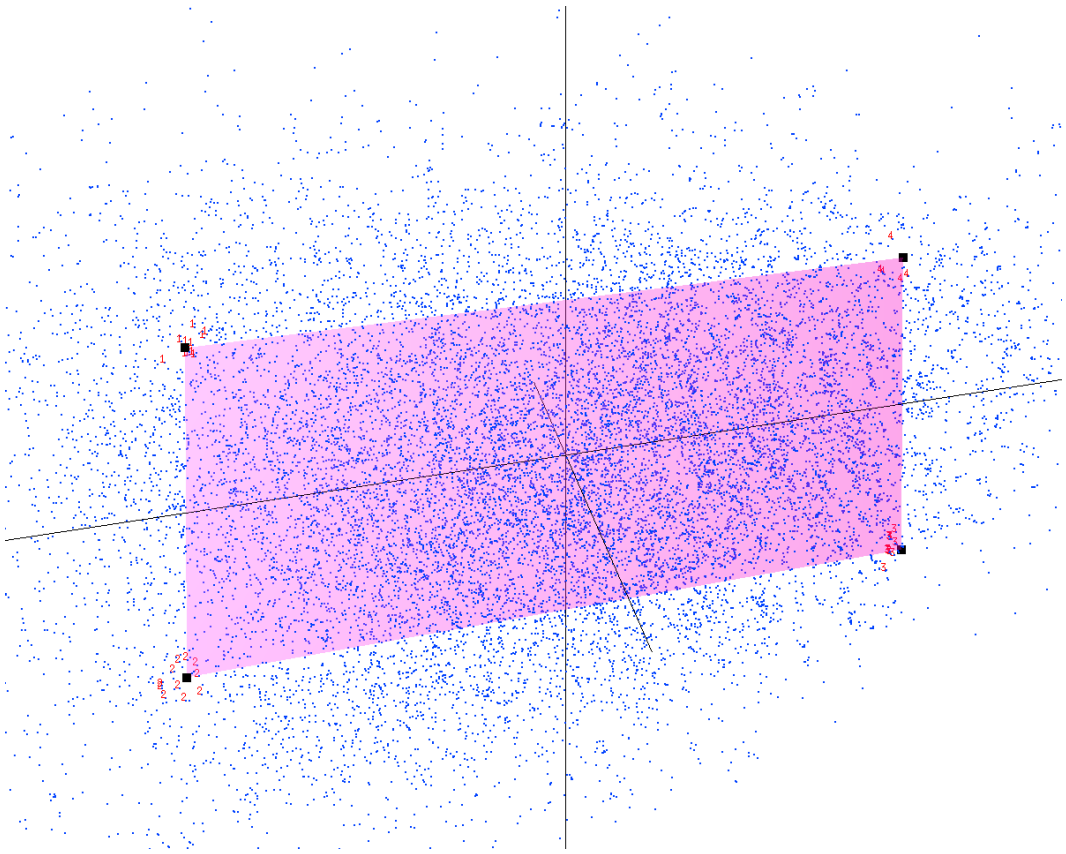


Figure 5.8: Figure of the plane incident with PC1 and PC2. PC1 is the horizontal axis; PC2 the vertical. The data are plotted in PCA-TE space as small blue points. The black squares are the limits of the plane embedded in the TE space. The coordinates of the four corners define our centroids for analysis, detailed in Table 5.3. The upper right is centroid 1; lower right, centroid 2; lower left, centroid 3; upper right centroid 4. The thermodynamic cycle will visit the 11 closest points (the closest point to the centroid, and its 10 closest neighbors) to each corner of the plane. The 11 closest points to the centroids are indicated in small red text, labeled by their respective centroid. At each corner we will dissect the thermodynamic subensembles and their energetic properties. We will determine the direct unfolding ΔASA as well as the complementary surface area exposed upon unfolding.

Index	PROT	AA	RESI	SS	TE	$[\Delta G]$	$[\Delta H_{ap}]$	$[\Delta H_{pol}]$	$[\Delta S]$
11865	1LCL	TRP	127	E	7	-12673	17158	-14213	-5036
8677	1IHK	SER	149	E	7	-13182	13178	-18042	-6047
16868	2ILK	ASP	44	C	2	-3597	2048	-9273	-3846
4647	1FW1	ASP	209	T	1	-2843	5838	-5424	-3382

Table 5.3: The four residues closest to the corners of the plane embedded in the span of PC1 and PC2. The 10 closest residues and their properties to each of these four points, will be investigated in this "thermodynamic cycle". It should be noted that the four centroids do not necessarily have to coincide with the coordinates from a point in our database. Thus we chose to investigate the nearest point to each centroid and its 10 closest neighbors.

$$\begin{array}{ccc} \texttt{TRP}\text{-}127_{Res_{11865}} & \rightleftharpoons & \texttt{ASP}\text{-}209_{Res_{4647}} \\ \downuparrows & & \downuparrows \\ \texttt{SER}\text{-}149_{Res_{8677}} & \rightleftharpoons & \texttt{ASP}\text{-}44_{Res_{16868}} \end{array}$$

we will start at residue TRP-127 in 1LCL, traverse to SER-149 in 1IHK along -PC2. We then traverse from SER-149 in 1IHK to ASP-44 in 2ILK along -PC1. Then, the ASP-44 in 2ILK to ASP-209 in 1FW1 transition will be examined, which is incident with +PC2. Finally, we examine the last transition, ASP-209 in 1FW1 to TRP-127 in 1LCL, which coincides with +PC1.

The Four Centroids and Their Neighbors

Centroid One The first centroid is located in TE space that is stable as it has $[\Delta G] \approx -12,673\ (cal/mol)$. The selected residues around centroid one are shown in Table 5.4 on page 73. Using the closest residue to the centroid (row 1) as a representative point for this area of TE space, we can calculate its change in surface

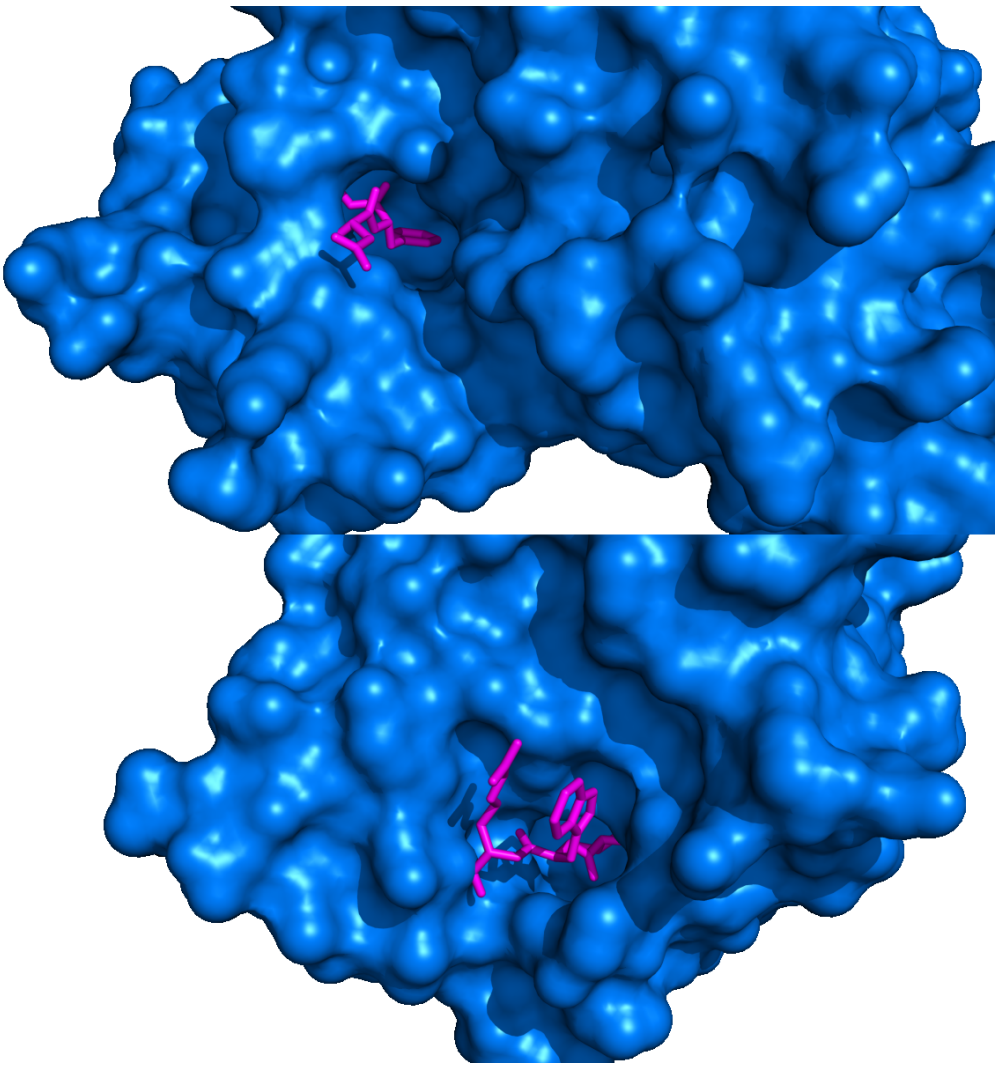


Figure 5.9: Two images showing points taken from Table 5.4 to illustrate the microenvironment near centroid one. Notice that upon removal of the magenta colored residues that a large amount of surface area would be exposed. Centroid one has the property that upon the $\mathcal{F} \rightarrow \mathcal{U}$ transition it unfolds 700 Å^2 more apolar surface area, and 350 Å^2 more polar surface area. This is the reason why residues around centroid one are relatively stable: the probability of microstates unfolding that much surface area is nearly 0.

area using Table 5.2. From the first row of Table 5.4 and Equations 2.18 and 2.19 we calculate the following:

$$\Delta ASA_{apol,\mathcal{F}} = 0.04 \text{ Å}^2 \quad (5.1)$$

$$\Delta ASA_{apol,\mathcal{U}} = 709.34 \text{ Å}^2 \quad (5.2)$$

$$\Delta ASA_{pol,\mathcal{F}} = 0.02 \text{ Å}^2 \quad (5.3)$$

$$\Delta ASA_{pol,\mathcal{U}} = 350.98 \text{ Å}^2 . \quad (5.4)$$

Equations 5.1–5.4 show that the $\mathcal{U}$ subensemble for residues near this point unfold a large amount of apolar and polar surface area. This makes the $\mathcal{U}$ probabilities nearly 0. And so these residues stable as we can see from the $[\Delta G]$. Also, the apolar to polar surface area change ratio is about 2 to 1. This characterizes the microenvironment around these points.

The data for the 10 points nearest the central point are found in Table 5.4. Figure 5.9 is an image showing two residues chosen randomly from the set of points near centroid one. When we separate out the surface area changes for the direct unfolding, versus the complementary surface area we see the results in Figure 5.10.

Centroid Two Using analysis similar to centroid one, we shall consider centroid two. The representative point is SER-149 from 1IHK. Thermodynamic data for the 11 points nearest the centroid are shown in Table 5.5. The general microenvironment has a high stability, $[\Delta G] \approx -13,500\ (cal/mol)$. The results for the changes

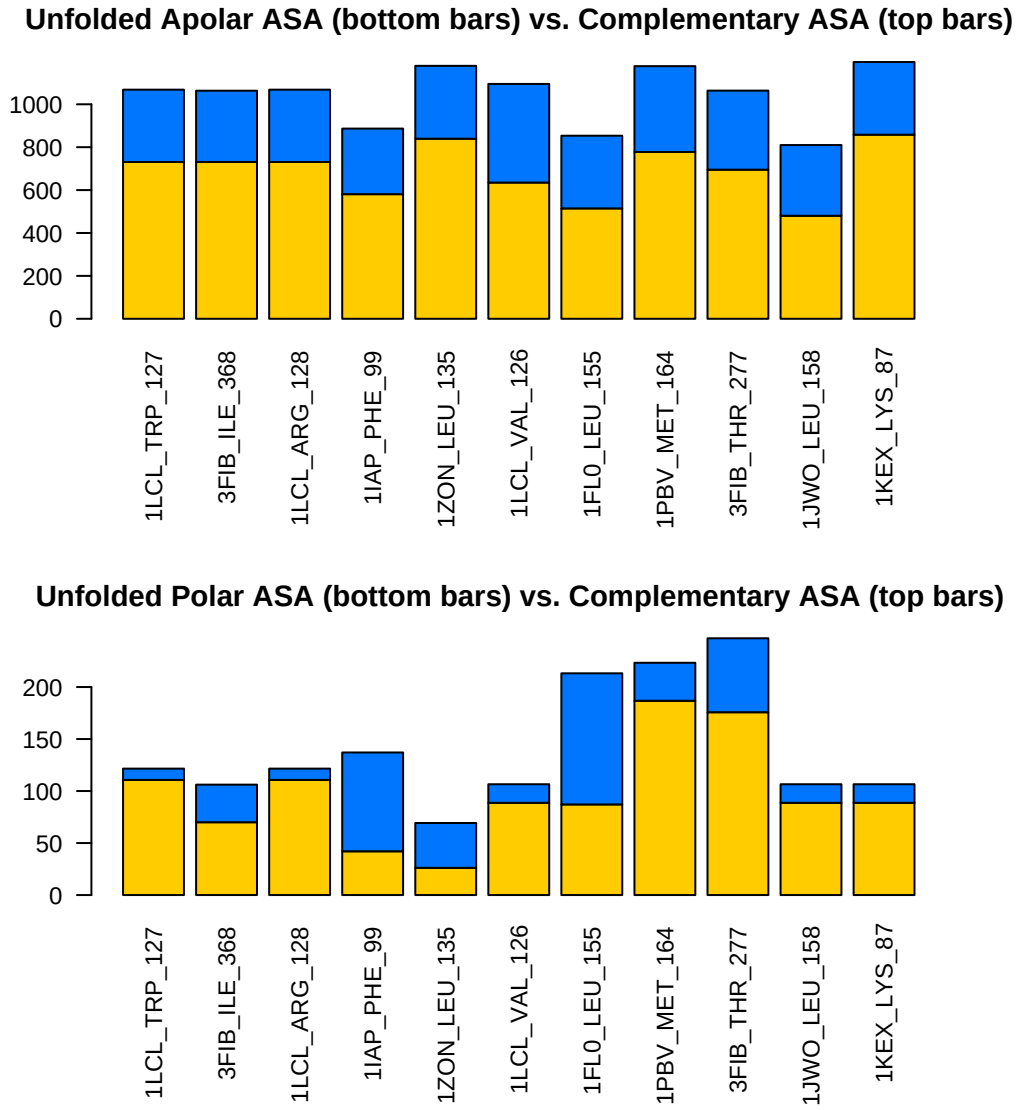


Figure 5.10: Illustration of the Apolar and Polar Changes in Directly Unfolded and Complementary Surface Areas for Residues Near Centroid One. The centroid is labeled, "1LCL_TRP_127". The total apolar surface area severely outweighs the polar surface area. Notice how on average the direct unfolded apolar surface area contribution is larger than the complementary surface area.

Index	Prot	AA	Res No	$[\Delta G]$	$[\Delta H_{ap}]$	$[\Delta H_{pol}]$	$T\,[\Delta S_{conf}]$	$\Delta H_{ap,f}$	$\Delta H_{ap,nf}$	$\Delta H_{pol,f}$	$\Delta H_{pol,nf}$	$\Delta S_{conf,f}$	$\Delta S_{conf,nf}$
11865	1LCL	TRP	127	-12673	17158	-14213	-5036	-1	-17159	1	14215	0.00	16.89
17418	3FIB	ILE	368	-12593	17264	-14244	-4778	-36	-17301	39	14284	0.07	16.09
11866	1LCL	ARG	128	-12391	16972	-14292	-5047	-1	-16973	1	14293	0.00	16.93
8332	1IAP	PHE	99	-12671	17376	-14348	-4913	-20	-17397	24	14373	0.04	16.52
15793	1ZON	LEU	135	-12480	17269	-14545	-5355	-3	-17273	4	14549	0.00	17.96
11864	1LCL	VAL	126	-12767	17334	-13889	-4960	-1	-17335	1	13890	0.00	16.63
3813	1FL0	LEU	155	-12479	17552	-14197	-5566	-57	-17609	79	14277	0.14	18.81
14003	1PBV	MET	164	-12637	16723	-14012	-5034	-132	-16856	391	14404	0.71	17.59
17328	3FIB	THR	277	-12376	17452	-13901	-5208	-36	-17489	39	13940	0.07	17.53
10171	1JWO	LEU	158	-12456	17085	-14640	-5117	-12	-17097	24	14664	0.03	17.20
11073	1KEX	LYS	87	-12138	16887	-14136	-5410	-831	-17719	707	14843	2.21	20.35

Table 5.4: Ten nearest neighbors to the points closest to the first point in the "thermodynamic cycle." This represents the microenvironment around the first spot.

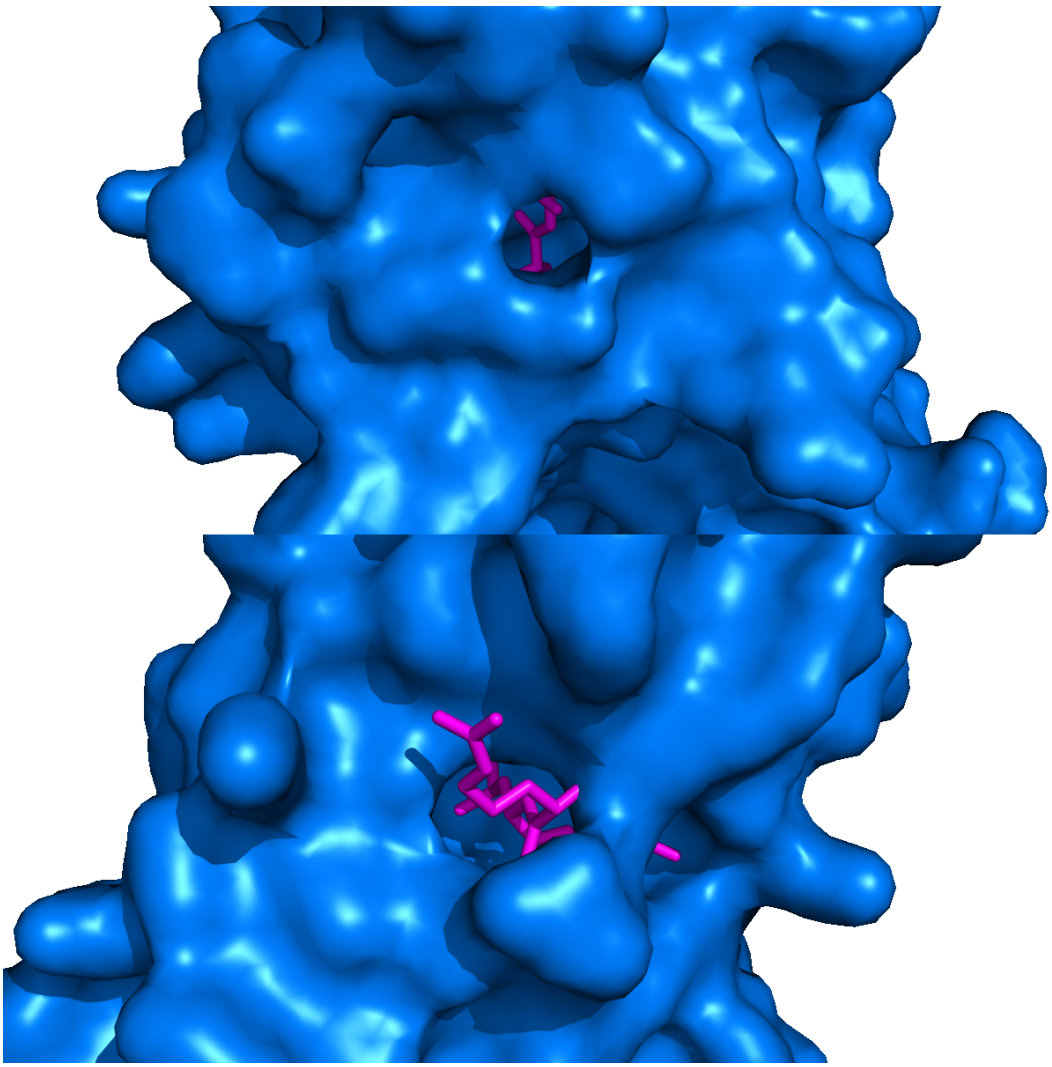


Figure 5.11: Figure Showing the Nearest Neighbors for Centroid 2. The surface of the protein is colored blue. The pink colored sticks are the residue and its neighbors of unfolding for points near the second centroid. Notice that because we have not traversed PC1 to go from centroid 1 to centroid 2, we do not see a large change in the magnitude of surface area exposure change for points near centroids 1 and 2. That is, the average apolar surface area change is about 1000 Å^2 and the average polar surface area change is about 175 Å^2 for both centroids.

in polar and apolar ASA for both the $\mathcal{F}$ and $\mathcal{U}$ subensembles are,

$$\begin{aligned}
\Delta ASA_{apol,\mathcal{F}} &= 10.17 \text{ Å}^2 \\
\Delta ASA_{apol,\mathcal{U}} &= 554.98 \text{ Å}^2 \\
\Delta ASA_{pol,\mathcal{F}} &= 8.94 \text{ Å}^2 \\
\Delta ASA_{pol,\mathcal{U}} &= 454.44 \text{ Å}^2 .
\end{aligned}$$

This microenvironment is characterized by a very slight change in the folded subensemble from the native state (low $\Delta ASA_{apol,\mathcal{F}}$ and $\Delta ASA_{pol,\mathcal{F}}$) and a large—but less apolar—change in the unfolded state (high $\Delta ASA_{apol,\mathcal{U}}$ and $\Delta ASA_{pol,\mathcal{U}}$, but low apolar to polar ratio of 1.22 to 1).

The barplot showing the direct and complementary surface area for both apolar and polar surface area changes is shown in Figure 5.12.

Centroid Three The data for centroid three are in Table 5.6. The region around centroid 3 is unstable, $[\Delta G] \approx -4,700$. The microenvironment around centroid 3 is characterized by a slight deviation from the native state for the folded subensemble and, with respect to centroids 1 and 2, a much smaller deviation from native state for the unfolded subensemble. Also, this is the first environment to unfold more polar surface area than apolar at about 2.26 to 1.00. The representative point is `ASP`-44 from 2ILK. Because the values in Equations 5.6 and 5.8 are lower than for centroids 1 and 2, we should see this reflected in the protein structure. Figure 5.14 illustrates this nicely.

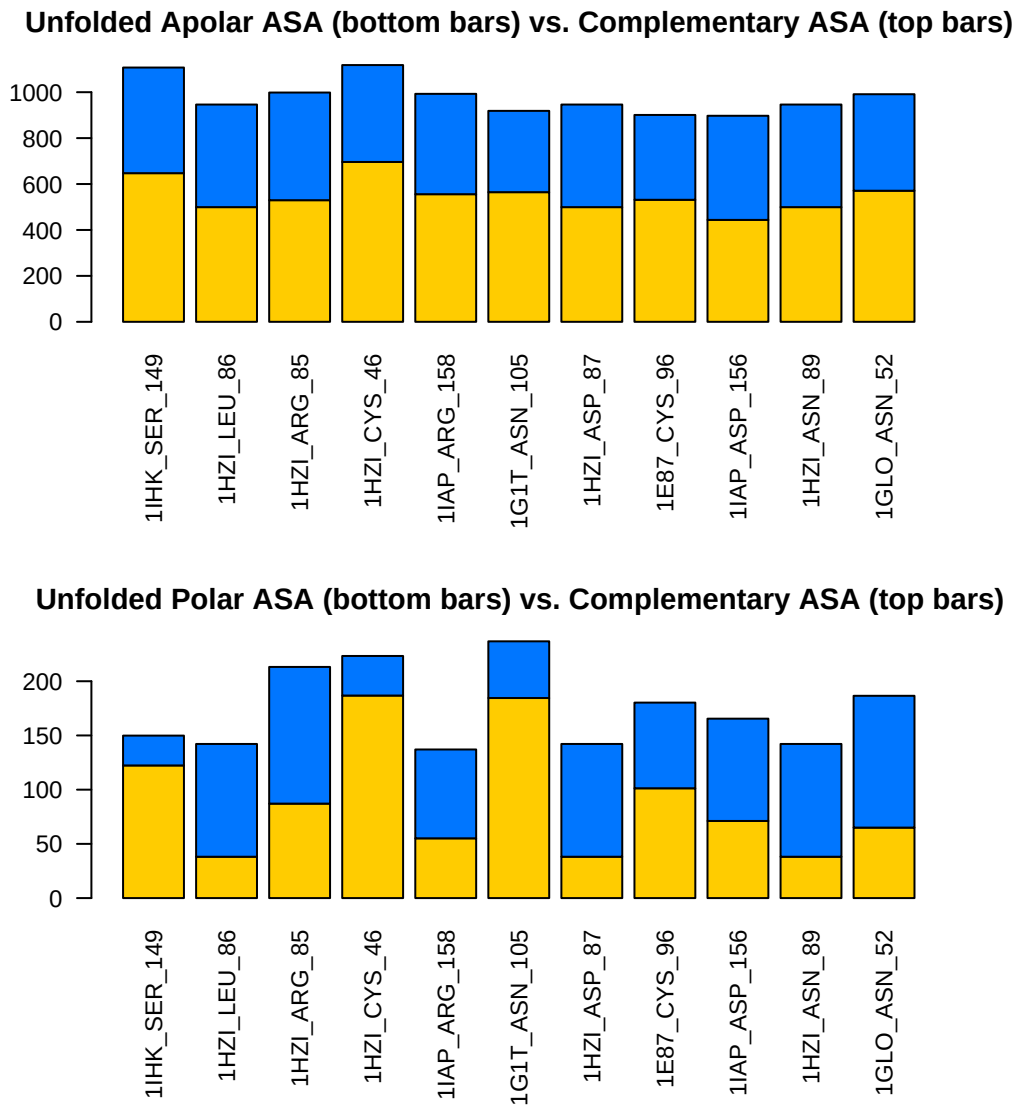


Figure 5.12: Explicit calculation of accessible surface area changes around centroid two.

Index	Prot	AA	Res No	$[\Delta G]$	$[\Delta H_{ap}]$	$[\Delta H_{pol}]$	$T\,[\Delta S_{conf}]$	$\Delta H_{ap,f}$	$\Delta H_{ap,nf}$	$\Delta H_{pol,f}$	$\Delta H_{pol,nf}$	$\Delta S_{conf,f}$	$\Delta S_{conf,nf}$
8677	1IHK	SER	149	-13182	13178	-18042	-6047	-246	-13425	362	18405	0.70	20.98
7336	1HZI	LEU	86	-13358	13140	-17933	-5245	-6	-13146	12	17945	0.01	17.61
7335	1HZI	ARG	85	-13468	13057	-18160	-5348	-6	-13063	12	18173	0.01	17.95
7296	1HZI	CYS	46	-13309	13544	-18045	-5631	-6	-13550	12	18057	0.01	18.90
8391	1IAP	ARG	158	-13542	13506	-18214	-5522	-20	-13526	24	18239	0.04	18.56
4755	1G1T	ASN	105	-13597	13088	-18195	-5595	-25	-13113	32	18228	0.05	18.82
7337	1HZI	ASP	87	-13370	12937	-17977	-5220	-6	-12943	12	17989	0.01	17.52
3287	1E87	CYS	96	-13426	13332	-18489	-6129	-44	-13377	80	18569	0.15	20.71
8389	1IAP	ASP	156	-13513	13669	-18054	-5455	-20	-13690	24	18079	0.04	18.34
7339	1HZI	ASN	89	-13156	13150	-17758	-5234	-6	-13157	12	17771	0.01	17.57
5412	1GLO	ASN	52	-12975	13396	-17876	-5528	-24	-13421	37	17914	0.06	18.61

Table 5.5: Ten nearest neighbors to the second centroid in the "thermodynamic cycle." This represents the microenvironment around the second spot.

$$\Delta ASA_{apol,\mathcal{F}} = 22.82 \text{ Å}^2 \quad (5.5)$$

$$\Delta ASA_{apol,\mathcal{U}} = 107.52 \text{ Å}^2 \quad (5.6)$$

$$\Delta ASA_{pol,\mathcal{F}} = 13.70 \text{ Å}^2 \quad (5.7)$$

$$\Delta ASA_{pol,\mathcal{U}} = 242.67 \text{ Å}^2 . \quad (5.8)$$

Centroid Four The microenvironment near centroid four is characterized by, again, a small deviation from native state for the folded subensemble, but now with a moderate magnitude of unfolding with a higher apolar to polar surface area change ratio. This centroid, like centroid 3, unfolds less total area than centroids 1 and 2. However, unlike centroid 3, it unfolds more apolar surface area. Figure 5.15 shows the residues in question and Table 5.7 shows the changes and types of surface area.

$$\Delta ASA_{apol,\mathcal{F}} = 3.30 \text{ Å}^2$$

$$\Delta ASA_{apol,\mathcal{U}} = 244.65 \text{ Å}^2$$

$$\Delta ASA_{pol,\mathcal{F}} = 3.03 \text{ Å}^2$$

$$\Delta ASA_{pol,\mathcal{U}} = 136.96 \text{ Å}^2 .$$

Index	Prot	AA	Res No	$[\Delta G]$	$[\Delta H_{ap}]$	$[\Delta H_{pol}]$	$T\,[\Delta S_{conf}]$	$\Delta H_{ap,f}$	$\Delta H_{ap,nf}$	$\Delta H_{pol,f}$	$\Delta H_{pol,nf}$	$\Delta S_{conf,f}$	$\Delta S_{conf,nf}$
16868	2ILK	ASP	44	-3597	2048	-9273	-3846	-552	-2601	555	9828	1.38	14.28
16869	2ILK	ASN	45	-3579	2178	-9215	-3836	-552	-2730	554	9769	1.38	14.25
17514	5PNT	THR	5	-3440	1776	-9030	-4186	-15	-1791	45	9075	0.06	14.10
6683	1HDR	GLY	165	-3310	2213	-9405	-4697	-38	-2251	49	9454	0.09	15.84
16867	2ILK	LEU	43	-3729	1517	-9603	-3876	-554	-2071	558	10161	1.39	14.39
15363	1RLW	ASN	95	-3905	2452	-9585	-4234	-25	-2478	38	9623	0.06	14.26
10165	1JWO	HIS	152	-3695	2212	-8654	-3701	-7	-2220	7	8661	0.01	12.42
10164	1JWO	ARG	151	-3695	2212	-8654	-3701	-7	-2220	7	8661	0.01	12.42
10166	1JWO	PRO	153	-3698	2187	-8643	-3696	-7	-2195	7	8650	0.01	12.41
6686	1HDR	SER	168	-3282	2425	-9406	-4731	-37	-2462	47	9454	0.08	15.95
6687	1HDR	GLY	169	-3316	2475	-9338	-4757	-37	-2512	49	9388	0.09	16.04

Table 5.6: Ten nearest neighbors to the points closest to the third point in the "thermodynamic cycle." This represents the microenvironment around the third spot.

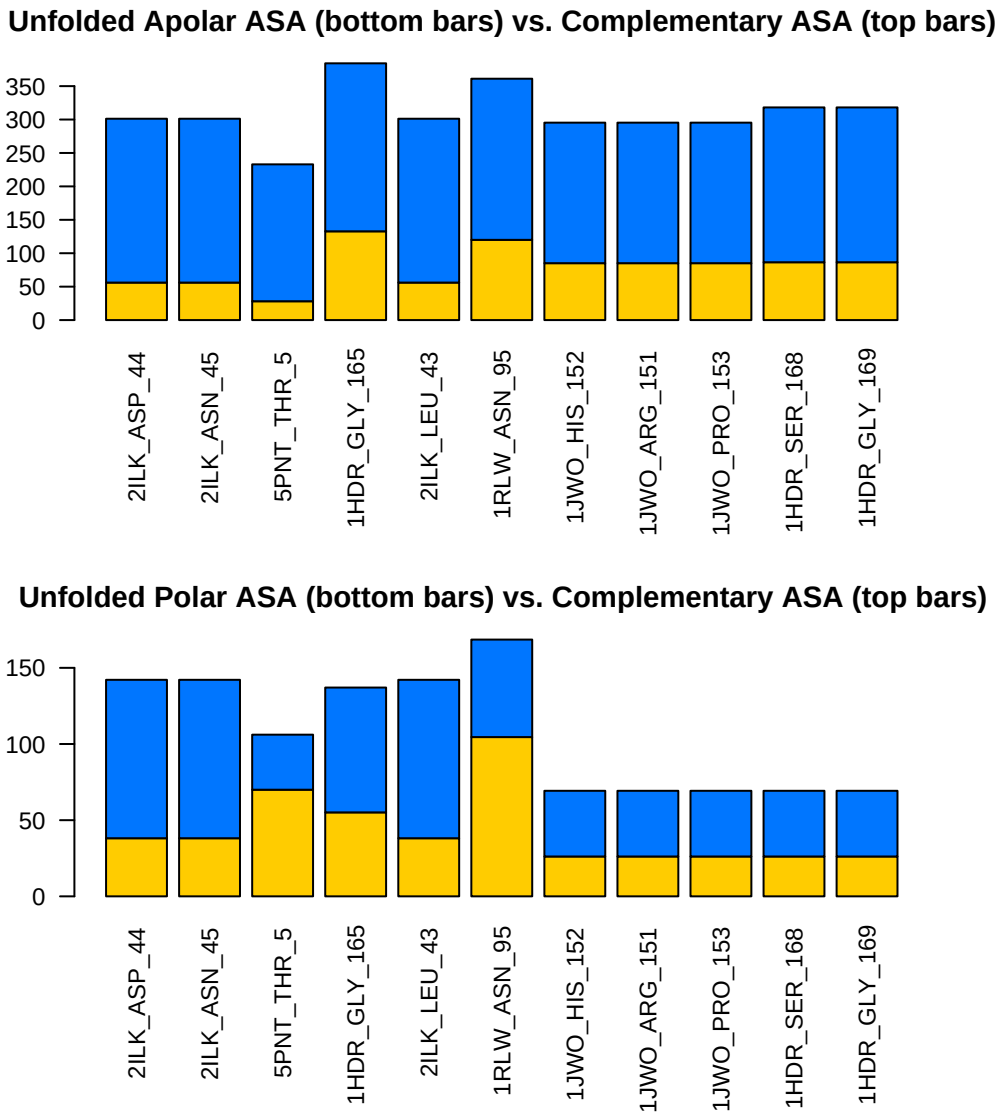


Figure 5.13: The Accessible Surface Areas in the Neighborhood for Centroid 3.

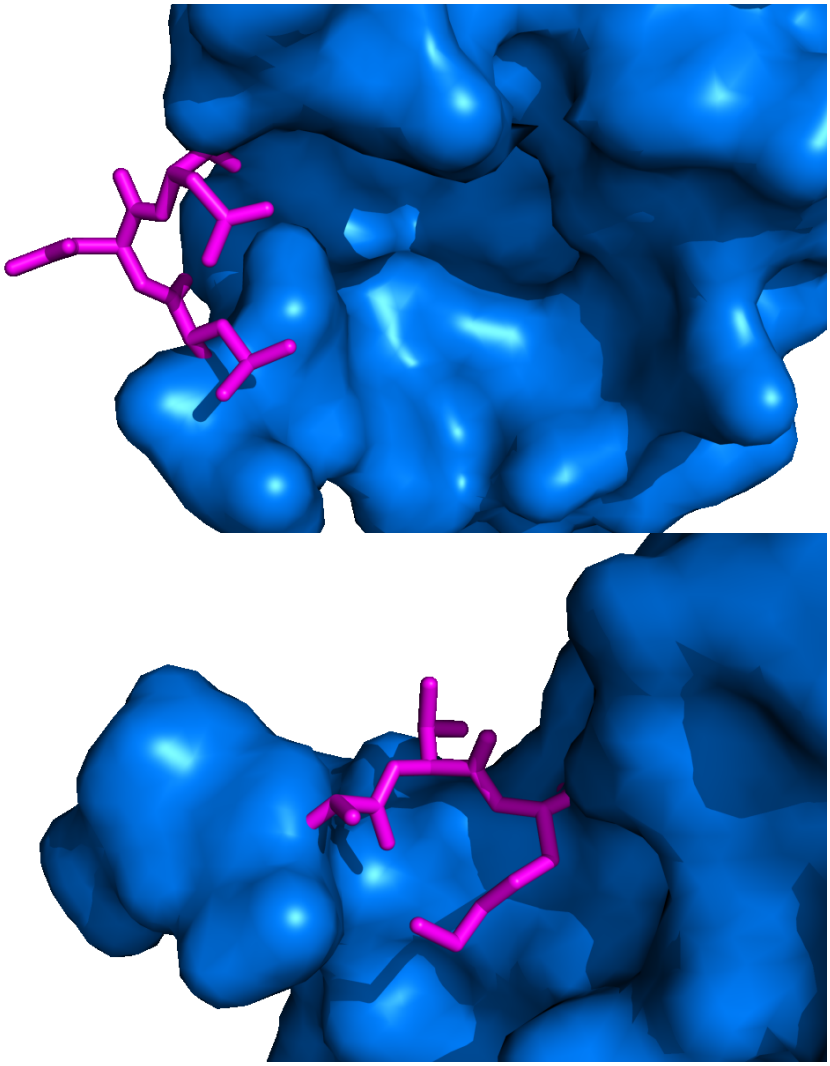


Figure 5.14: Example Residues from the Neighborhood for Centroid 3. Notice that in accordance with the unfolding surface areas we are expecting to see much less change in accessible surface area upon unfolding as we would for centroids 1 and 2. Centroids 1 and 2 upon unfolding would leave large solvent accessible cavities, centroid 3 does not.

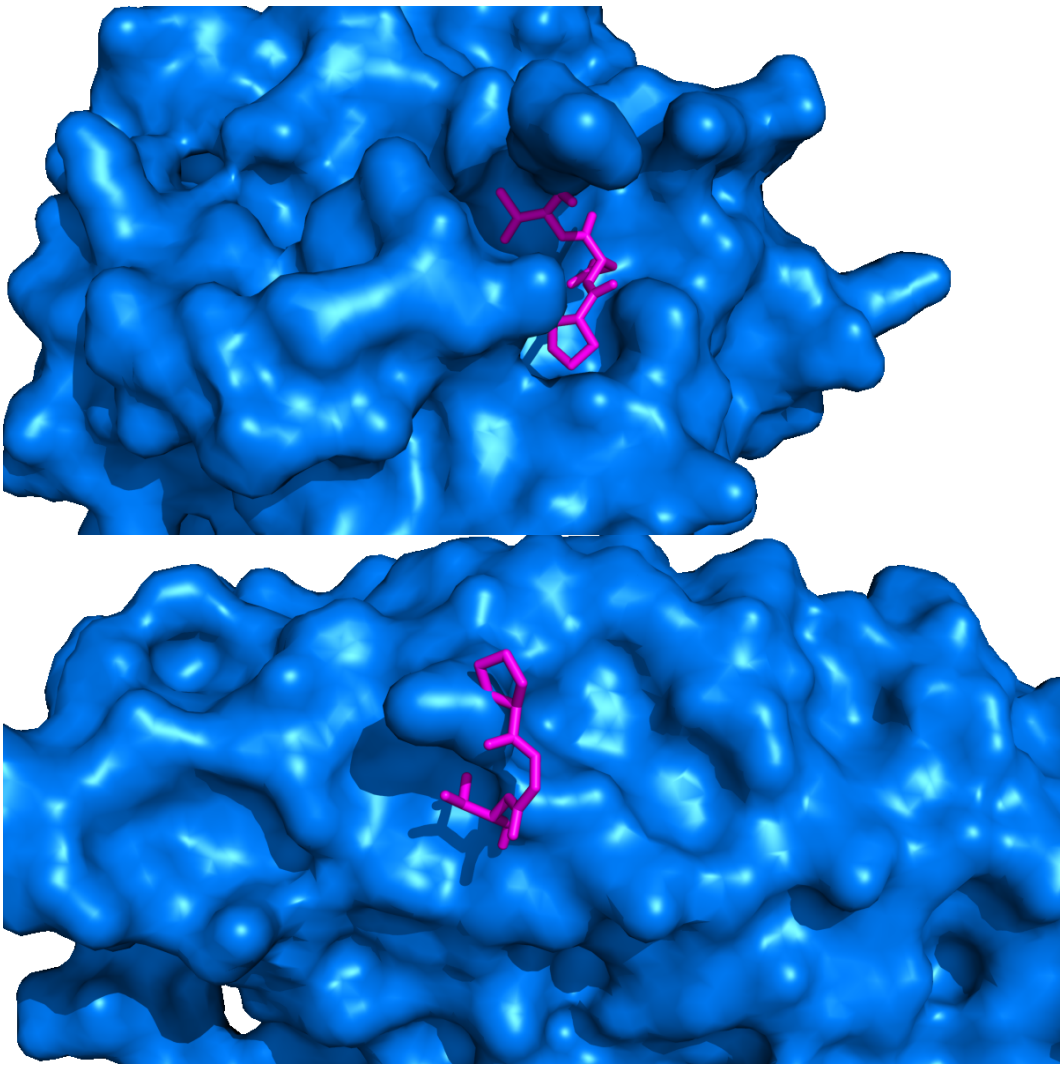


Figure 5.15: Example Residues for the Fourth Neighborhood.

Index	Prot	AA	Res No	$[\Delta G]$	$[\Delta H_{ap}]$	$[\Delta H_{pol}]$	$T\,[\Delta S_{conf}]$	$\Delta H_{ap,f}$	$\Delta H_{ap,nf}$	$\Delta H_{pol,f}$	$\Delta H_{pol,nf}$	$\Delta S_{conf,f}$	$\Delta S_{conf,nf}$
4647	1FW1	ASP	209	-2843	5838	-5424	-3382	-80	-5918	123	5547	0.24	11.58
4648	1FW1	THR	210	-2843	5838	-5424	-3382	-80	-5918	123	5547	0.24	11.58
4649	1FW1	PRO	211	-2843	5838	-5424	-3382	-80	-5918	123	5547	0.24	11.58
4650	1FW1	THR	212	-2843	5838	-5424	-3382	-80	-5918	123	5547	0.24	11.58
10466	1K1B	GLY	223	-2730	5812	-5265	-3414	-136	-5949	649	5915	1.28	12.73
10467	1K1B	THR	224	-2737	5819	-5219	-3408	-137	-5956	650	5870	1.00	12.71
10468	1K1B	LEU	225	-2975	6868	-4551	-3440	-149	-7017	671	5223	1.32	12.86
10465	1K1B	PRO	222	-2730	5810	-5265	-3414	-136	-5947	649	5914	1.28	12.73
8883	1IJT	VAL	121	-2604	5884	-5206	-3433	-32	-5916	51	52	0.08	11.60
4907	1G96	THR	109	-3221	5773	-5468	-3110	-895	-6668	2194	7663	5.63	16.06
8884	1IJT	GLU	122	-2597	5912	-5227	-3434	-30	-5943	50	5277	0.08	11.60

Table 5.7: Ten nearest neighbors to the points closest to the fourth point in the "thermodynamic cycle." This represents the microenvironment around the fourth spot.

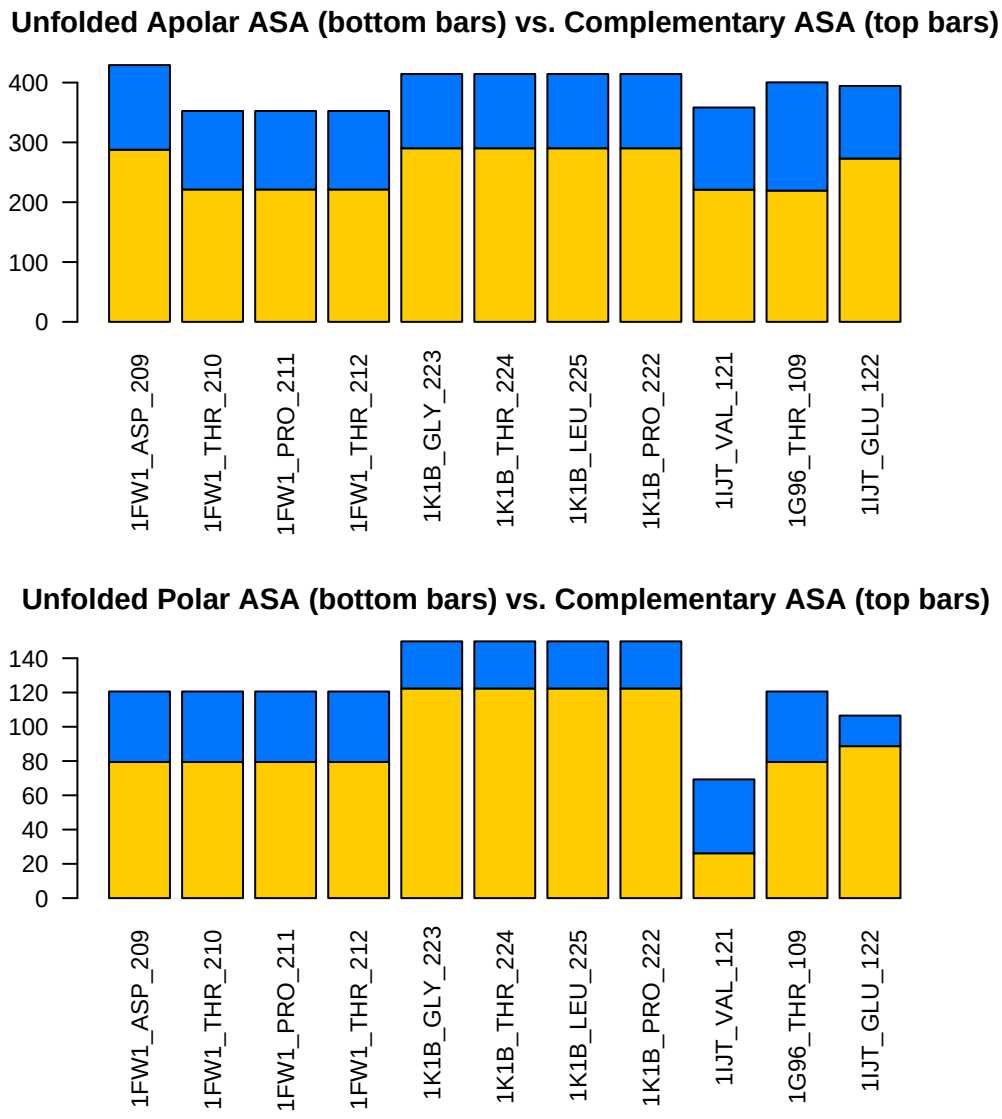


Figure 5.16: Accessible Surface Area Changes for Residues in the Fourth Neighborhood.

Summary This tour around the TE space gives us a more in-depth view of the energetics changes associated with various parts of the space. Recapping, centroids 1 and 2 expose very large amounts of total surface area with respect to centroids 3 and 4. This is due to the large distance of centroids 1 and 2 from centroids 3 and 4 with regard to PC1. The apolar to polar surface area change ratios for centroids 1 and 4 are much higher, about 2 to 1, with regard to centroids 2 and 3.

As all four tables of accessible surface area changes show, the microstate energetics of specific microenvironments are all following similar biophysical phenomena. We can make this statement because the energetics of the folded and unfolded subensembles in each microenvironment are similar. If one considers that the $[\Delta H_{\text{apol}}]$ is made up from the difference of two subensembles then there could be an infinite number of ways of achieving the same $[\Delta H_{\text{apol}}]$. (For example, $[\Delta H_{\text{apol}}] = 10$ could be from $\langle \Delta H_{\text{apol}} \rangle_{\mathcal{F}} = 20$ and $\langle \Delta H_{\text{apol}} \rangle_{\mathcal{U}} = 10$, or it could be from $\langle \Delta H_{\text{apol}} \rangle_{\mathcal{F}} = -1000$ and $\langle \Delta H_{\text{apol}} \rangle_{\mathcal{U}} = -1010$. We observe that residues nearby each other in TE space have similar subensemble energetics: microenvironments are defined by the same, or similar, biophysical process.)

5.5 Discussion of the Interpretation of the Biophysical Characterization of TE Space

Determination of the Distance Between two Residues in TE Space We can now entertain some relevant questions with our new biophysical view of the TE space. The first posited was: why are two residues close together or far apart in TE

space, even with amino acid, sequence, secondary structure and other differences? Two residues are close together in TE space if their differences in the folded and unfolded subensembles, expose similar amounts and types of surface area.

But, this can be done in more than just one or two ways. The structure and most stable states could expose a large amount of polar surface area, only to have its significance (on PC2) wiped reduced by a distal patch of residues unfolding a large amount of apolar surface area (long-range communication: cooperativity). Inherent in all COREX data are structure, sequence, solvent accessibility relationships, and all possible combinations of partial unfolding.

The Arrowhead Shape Given our new interpretation of the shape of TE space, we can answer this question as well. These boundaries are the limits of the differences of apolar and polar surface area that a protein can unfolded in all possible combinations. Towards the point of the arrowhead, the $\Delta ASA_{\text{apolar}}$ and $\Delta ASA_{\text{polar}}$ are coming together because their subensembles are unfolding smaller and smaller amounts of surface area.

Can we apply this knowledge? Yes. The future implications are discussed in Chapter 7. But, let me state here that engineering principles for residue mutations, stability mutations—even fine tuned for secondary structures—can be devised given our interpretation of the TE space.

Chapter 6

The Thermodynamic Definition of Protein Folds

6.1 Database of *Homo sapiens* Proteins

In order to the study the thermodynamic representation of proteins, we collected a sample of *Homo sapiens* proteins from the PDB (Berman et al., 2000). Each protein had to have the following properties: (1) be of reasonable length, 50–250 amino acids; (2) be a complete structure (no missing atoms); (3) be results from an X-ray crystallography study of no more than 2.5 Å RMSD. All redundant atomic coordinates for "alternate locations" were removed.

Relevant Collected Data We employed the Stride (Frishman and Argos, 1995) algorithm for secondary structure determination for each residue in our *Homo sapiens* database. Larson donated the list of codons for each amino acid in the database.

Other information that was collected includes: the molecular weight of each amino acid; the surface area for each amino acid; the length of each protein in the database; and the SCOP and CATH protein family information. Table 6.1 lists the proteins we used in this study.

Table 6.1: Table showing the list of proteins used in this study.

PDBID	Protein Name
1a17	Serine/Threonine Protein Phosphatase 5
1a3k	Galectin-3
1ad6	Retinoblastoma Tumor Suppressor
1aly	Cd40 Ligand
1b56	Fatty Acid Binding Protein
1b9o	Alpha-Lactalbumin
1bd8	P19ink4d Cdk4/6 Inhibitor
1bik	Bikunin
1bkf	Fk506 Binding Protein
1bkr	Spectrin Beta Chain
1br9	Metalloproteinase-2 Inhibitor
1buo	Promyelocytic Leukemia Zinc Finger Protein Plzf
1by2	Mac-2 Binding Protein
1byq	Heat Shock Protein 90
1cbs	Cellular Retinoic-Acid-Binding Protein Type Ii
1cdy	T-Cell Surface Glycoprotein Cd4

1cll	Calmodulin
1ctq	Transforming Protein P21/H-Ras-1
1cy5	Apoptotic Protease Activating Factor 1
1czt	Coagulation Factor V
1d7p	Coagulation Factor Viii Precursor
1dv8	Asialoglycoprotein Receptor 1
1e21	Ribonuclease 1
1e87	Early Activation Antigen Cd69
1eaz	Tandem Ph Domain Containing Protein-1
1esr	Monocyte Chemotactic Protein 2
1fao	Dual Adaptor Of Phosphotyrosine
1fil	Profilin
1fl0	Endothelial-Monocyte Activating Polypeptide Ii
1fna	Fibronectin Cell-Adhesion Module Type Iii-10
1fnl	Low Affinity Immunoglobulin Gamma Fc Region
1fp5	Ige Heavy Chain Epsilon-1
1fw1	Glutathione Transferase Zeta
1g1t	E-Selectin
1g96	Cystatin C
1gen	Gelatinase A
1ggz	Calmodulin-Related Protein Nb-1
1gh2	Thioredoxin-Like Protein
1glo	Cathepsin S

1gnu	Gabarap
1gp0	Cation-Independent Mannose-6-Phosphate Receptor
1gqv	Eosinophil-Derived Neurotoxin
1gsm	Mucosal Addressin Cell Adhesion Molecule-1
1h6h	Neutrophil Cytosol Factor 4
1hdo	Biliverdin Ix Beta Reductase
1hdr	Dihydropteridine Reductase
1hmt	Human Muscle Fatty Acid Binding Protein
1hna	Glutathione S-Transferase (Human, Class Mu) (Gstm2-2)
1hup	Mannose-Binding Protein
1hzi	Interleukin-4
1i1n	Protein-L-Isoaspartate O-Methyltransferase
1i27	Transcription Factor Iif
1i2t	Hyd Protein
1i4m	Major Prion Protein
1i71	Apolipoprotein(A)
1i76	Neutrophil Collagenase
1iam	Intercellular Adhesion Molecule-1
1iap	Guanine Nucleotide Exchange Factor P115rhogef
1ifr	Lamin A/C
1ihk	Glia-Activating Factor
1ijr	Proto-Oncogene Tyrosine-Protein Kinase Lck
1ijt	Fibroblast Growth Factor 4

1ikt	Estradiol 17 Beta-Dehydrogenase 4
1imj	Ccg1-Interacting Factor B
1j74	Mms2
1jhj	Apc10
1jk3	Macrophage Metalloelastase
1jsf	Lysozyme
1jsg	Oncogene Product P14tcl1
1jwf	Adp-Ribosylation Factor Binding Protein Gga1
1jwo	Csk Homologous Kinase
1k04	Focal Adhesion Kinase 1
1k1b	B-Cell Lymphoma 3-Encoded Protein
1k59	Angiogenin
1kao	Rap2a
1kcq	Gelsolin
1kex	Neuropilin-1
1kpf	Protein Kinase C Interacting Protein
1kth	Collagen Alpha 3(Vi) Chain
1l8j	Endothelial Protein C Receptor
1l9l	Granulysin
1lcl	Lysophospholipase
1lds	Beta-2-Microglobulin
1ln1	Phosphatidylcholine Transfer Protein
1lpj	Retinol-Binding Protein Iv, Cellular

1lsl	Thrombospondin 1
1m7b	Rnd3/Rhoe Small Gtp-Binding Protein
1m9z	Tgf-Beta Receptor Type Ii
1mfm	Copper, Zinc Superoxide Dismutase
1mh1	Rac1
1mh9	Deoxyribonucleotidase
1mj4	Sulfite Oxidase
1mwp	Amyloid A4 Protein
1n6h	Ras-Related Protein Rab-5a
1nkr	P58-Cl42 Kir
1pbk	Fkbp25
1pbv	Arno
1pht	Phosphatidylinositol 3-Kinase P85-Alpha Subunit
1pod	Phospholipase A2 (E.C.3.1.1.4)
1qb0	M-Phase Inducer Phosphatase 2 (Cdc25b)
1qdd	Lithostathine
1qkt	Estradiol Receptor
1quu	Human Skeletal Muscle Alpha-Actinin 2
1rbp	Retinol Binding Protein
1rlw	Phospholipase A2
1sra	Sparc
1ten	Tenascin (Third Fibronectin Type Iii Repeat)
1tn3	Tetranectin

1zon	Leukocyte Adhesion Glycoprotein
1zxq	ICAM-2
2abl	Abl Tyrosine Kinase
2cpl	Cyclophilin A
2fcb	Fc Gamma Riib
2fha	Ferritin
2ilk	Interleukin-10
2psr	Psoriasin
2tgii	Transforming Growth Factor-Beta Two (Tgf-B2)
3fib	Fibrinogen Gamma Chain Residues
3il8	Interleukin 8
5pnt	Low Molecular Weight Phosphotyrosyl Phosphatase

Executing COREX over the Database of *Homo sapiens* Proteins We executed the COREX algorithm on each protein in the database. The parameters used were the same as those from Larson (Larson and Hilser, 2004). The "window size" was 5, the "minimum window size" was 4. For shorter proteins the entire protein ensemble was enumerated. For longer proteins, a Monte Carlo (MC) (Metropolis and Ulam, 1949) sampled ensemble was created (Larson and Hilser, 2004), with the "number of samples kept" at 250,000. Larson had determined that for proteins with less than 250 amino acids, collecting 250,000 states was sufficient to accurately represent the information in the entirely enumerated ensemble. The "entropy

scaling factor" was set at 0.5, favoring the native state. The experiments were carried out at the theoretical temperature of 25° C.

6.2 Structure Alignment

6.2.1 Introduction and Statement of the Problem

Structure Representation A protein structure is the set of atomic coordinates in $\mathbb{R}^3$ collected from a high-resolution structure determination experiment. Define the vector $\mathbf{a}_i$ to be the (x, y, z) coordinates of the ith atom in the set of atomic coordinates. Then for the problem of structure alignment we augment all the vectors $\mathbf{a}_i$ into a matrix,

$$\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_i \\ \vdots \\ \mathbf{a}_N \end{bmatrix} = A \in \mathbb{R}^{N\times 3}\,, \tag{6.1}$$

where N is the number of atoms in the protein. For the problem of protein structure alignment—and in this project—only the atomic coordinates of the α-carbons are used. This has been shown to be useful because it reduces the number of coordinates needed for comparison and it retains the biologically relevant structural details (Holm and Sander, 1993; Shindyalov and Bourne, 1998). Thus, for a protein with n residues its matrix representation would be $A \in \mathbb{R}^{n\times 3}$, and its transpose

$A^{\perp} \in \mathbb{R}^{3\times n}$.

Statement of the Problem The problem of protein structure alignment is defined as follows. Given two proteins, evolutionarily related or not, determine largest subsets of equal number of residues from each protein, such that the sum squared deviation (RMSD) between the optimally superposed subsets is minimized. As mentioned in the Section 2.2.2, this requires two steps. The first is selecting which residues in the first protein will be matched with which residues in the second protein. The second is computing the optimal translation and rotation to determine the RMSD of the optimal alignment. The details of each step are now considered.

6.2.2 The First Step: Detecting the Near-Optimal Subsets

Introduction There are various methods to determine the near-optimal subsets of residues from the two proteins to be aligned. The term "near-optimal" is used because the problem of finding the provably optimal subsets is classified as NPC. The methods vary greatly from the difference of distance matrices (DALI) (Holm and Sander, 1993), secondary structure matching combined with the branch-and-bound method (Holm and Sander, 1996), combinatorial extension of the optimal alignment path (Shindyalov and Bourne, 1998), genetic algorithms (Szustakowski and Weng, 2000), deterministic annealing (Chen et al., 2005), iterated double dynamic programming (Taylor, 1999) and many more.

In an interesting aside, Godzik compared the alignments from the literature of some proteins and determined that in many cases the optimal alignments are

not unique, but exist in a family of optimal alignments (Godzik, 1996). The level of agreement of various methods often diverged proportional to the divergence of sequence identity. The two most salient methods in use today are DALI and CE. Both were implemented for this project. CE was chosen for its greater efficiency and longer alignments. CE also makes no assumptions about secondary structure, as DALI does (in its second revision (Holm and Sander, 1996)).

6.2.3 The CE Method

Introduction CE was created by Drs. Shindyalov and Bourne (Shindyalov and Bourne, 1998); improved with the addition of the "CE Score" (Jia et al., 2004); and extended for multiple structures using Monte Carlo methods (Guda et al., 2004); and augmenting sequence information (Ponomarenko et al., 2005). CE is similar to other methods in that it calculates a global alignment considering the difference of distance matrices.

Mathematical Necessities In order to discuss the CE method, I first introduce the necessary mathematical tools, the distance matrix and the similarity matrix.

Definition 3 (Distance Matrix)**.** *Let V be a given vector set with m entries in $\mathbb{R}^n$, like in Equation 6.1. We can write V as a matrix:* $V = \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,n} \\ v_{2,1} & v_{2,2} & v_{2,n} \\ & \vdots & \\ v_{m,1} & v_{m,2} & v_{m,n} \end{bmatrix}$. *Then, a **distance matrix** for the vector set V is the matrix D such that each element $d_{i,j} \in D$ represents the distance, usually Euclidean, from vector v_i to vector v_j.*

Symbolically,

$$D_{i,j} = \sum_{i=1}^{m} \sum_{j=1}^{m} \sqrt{\sum_{k=1}^{n} (V_{i,k} - V_{j,k})^2} \tag{6.2}$$

An example distance matrix for the P14TCL1 protein (PDBID: 1JSF) is shown in Figure 6.1. Note the symmetry about the main diagonal. Because the distance from point v_i to point v_j in some object is the same as the distance from point v_j to point v_i, all distance matrices are symmetric. (This brings about many interesting possibilities in linear algebra. First, the matrix is orthogonally diagonalizable, the matrix is positive semi-definite—which indicates real valued eigenvalues—an orthonormal eigenspace for the columns of the matrix can be created, and more. This is discussed further in Section 6.2.5.)

Definition 4 (Similarity Matrix)**.** *Let $a \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^{p \times n}$ be two vector sets, with m and p entries in $\mathbb{R}^n$, respectively. Furthermore, let $A \in \mathbb{R}^{m \times m}$ and $B \in \mathbb{R}^{p \times p}$ be their respective distance matrices created from Equation 6.2. The similarity matrix, S for the two structures is defined as:*

$$S_{i,j} = \frac{1}{w} \sum_{i=1}^{m-w} \sum_{j=1}^{p-w} \sum_{k=1}^{w} A_{i,i+k} - B_{j,j+k}, \tag{6.3}$$

where $w \in Z^+$ is a fixed window size, here $w = 8$.

The similarity matrix measures the similarity between two substructures, one anchored at residue i to $i + k \in a$ and the other anchored at j to $j + k \in b$. High values denote dissimilar regions, low values similar regions, and 0 is an exact match. Because the distance from points $(a_i, a_j) \in A$ is not necessarily the same

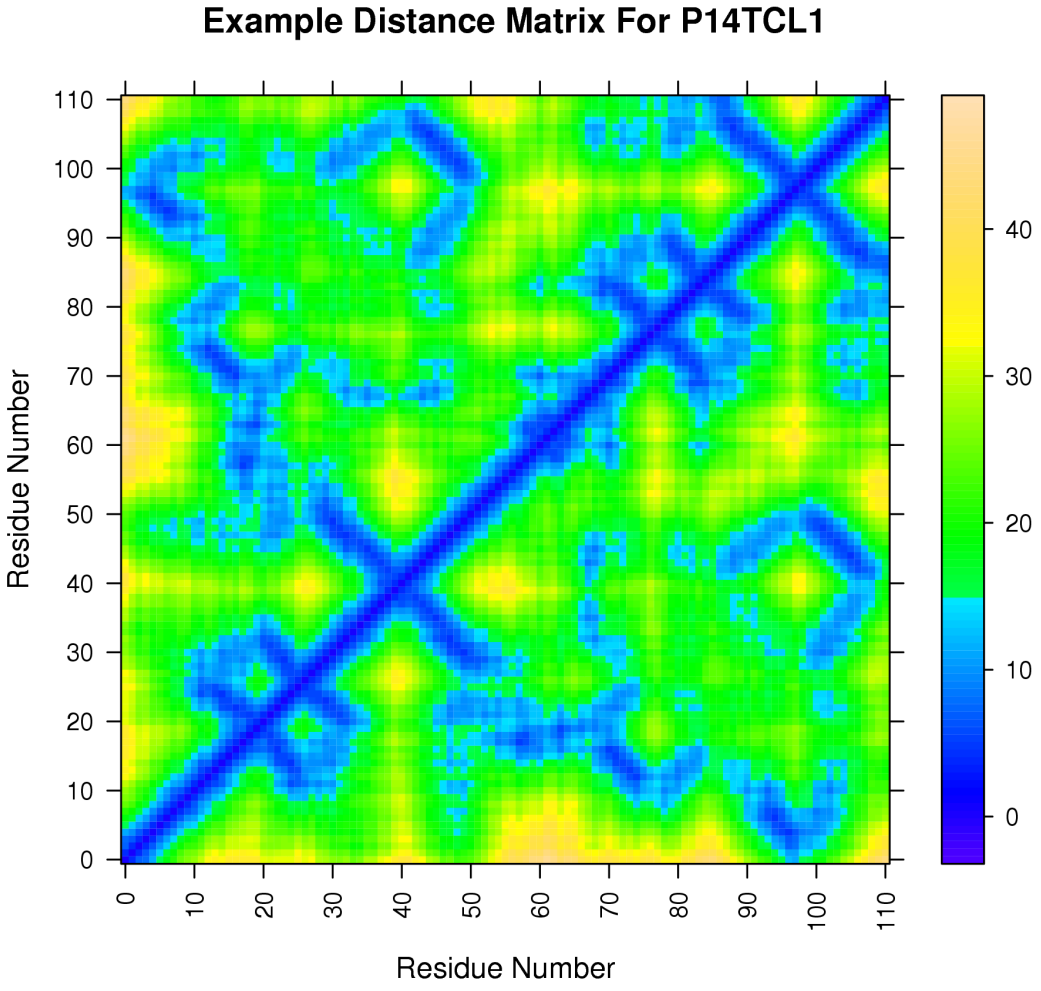


Figure 6.1: Example Distance Matrix for the P14TCL1 protein. Units of distance in Å.

as the distance from point $(b_i, b_j) \in B$, not all distance matrices are symmetric. In fact, they're only symmetric when $a \equiv b$. Figure 6.2 shows an example similarity matrix for a protein against itself.

Also, any row, say i, in S describes the similarity of the substructure anchored at residue i to $i + w \in a$ to every contiguous span of w residues in b. Analogously, any column in S measures the similarity of the substructure anchored at some residue in b to all those substructures throughout a. So, the signal in the rows can be thought of as coming from a and the signal from the columns in b. This become an important concept for analysis in Section 6.2.5.

Near-optimal paths can be quickly read off of a similarity matrix. For example, to find the best path through the similarity matrix in Figure 6.3 just start in the lower left corner and trace to the upper right corner along the lowest scoring diagonal path (dark green). Every (i, j) element in the similarity matrix deemed a good match then pairs together residue i in protein A to residue j in protein B. Gaps are allowed: move one or more positions to the up or left, in the similarity matrix. This figure shows an example similarity matrix for a protein against another with high structural overlap. To contrast, consider the similarity matrix in Figure 6.4; this shows an example similarity matrix for a protein and a non-homologous target. Notice how the optimal path in the latter is not readily apparent. The source code for creating a similarity matrix is shown in Listing 6.1.

Source Code 6.1: Source Code for Making a Similarity Matrix

```
1 //=============================================================
2 // Calculates the Similarity matrix, S.
3 //=============================================================
```

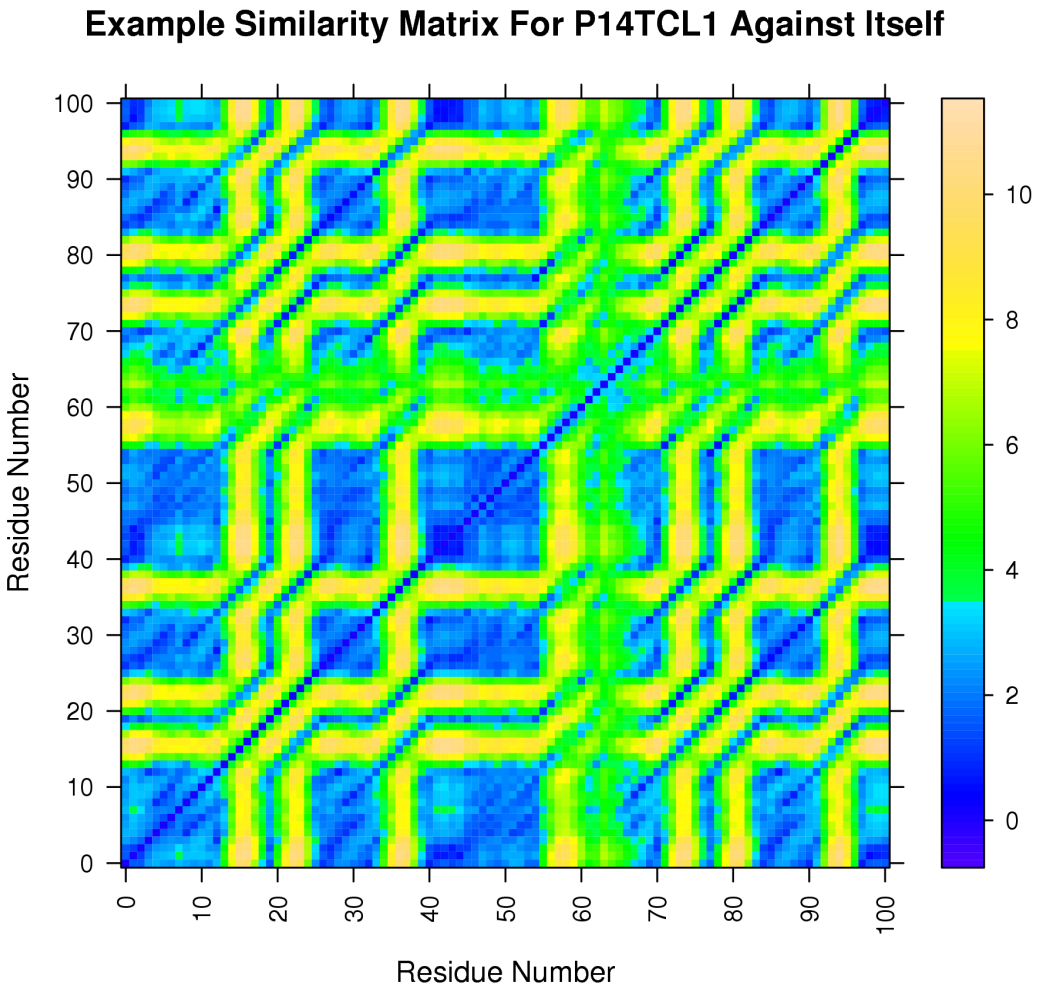


Figure 6.2: Similarity Matrix for Protein P14TCL1 Against Itself. Notice that the matrix is indeed symmetric. Finding the optimal alignment is equivalent to finding the longest path of lowest scores (dark green) from $(1, 1)$ in the lower left corner to $(120, 120)$ in the upper right hand corner. Repeated substructure is easy to find: look for rectangular dark colored blocks. Unique substructure is easy to find as well, look for high scoring regions that run through the entire protein (columns 37-40, 46-49). These usually correspond to loop/turn regions in proteins. Also, notice that off-diagonal regions of low scores indicates similar matching substructure at sequentially distant residues.

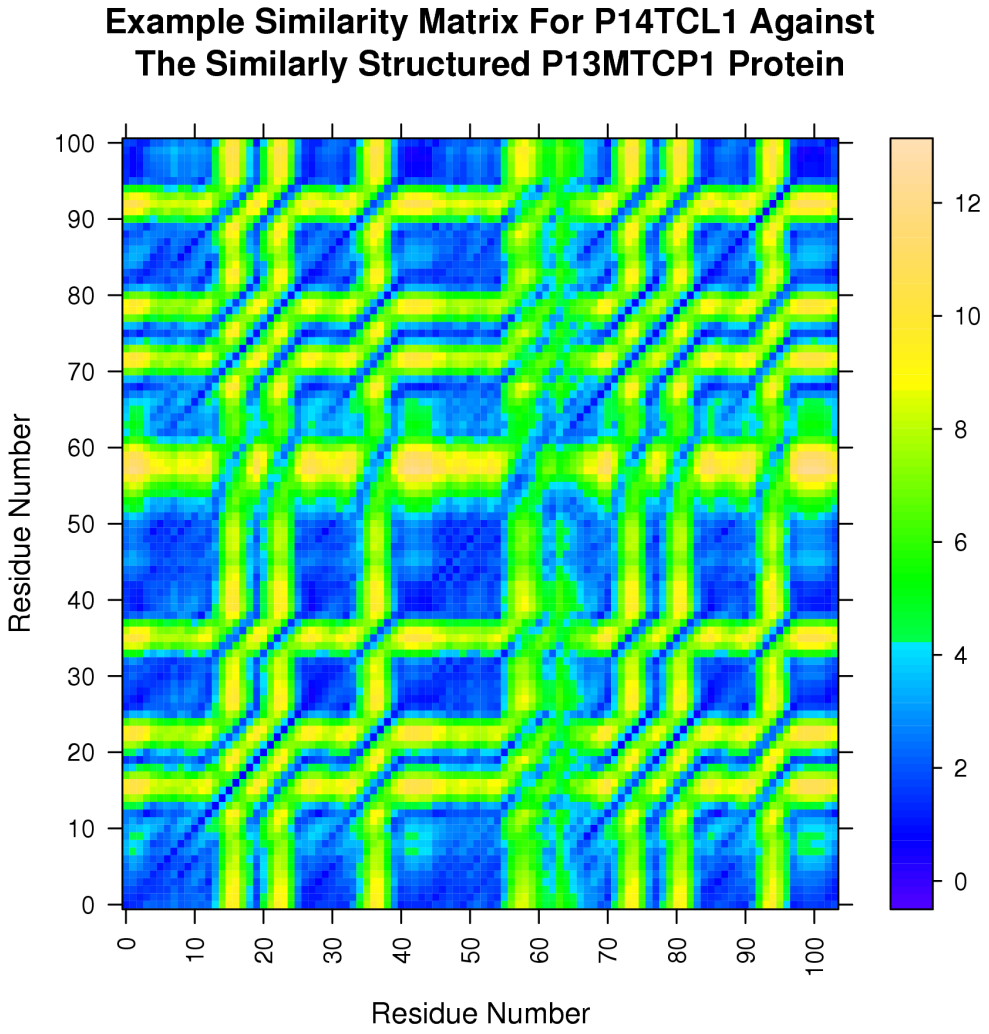


Figure 6.3: In this figure, notice that while it does look symmetric, it is not. The width of the regions around residues 55–60 are not the same.

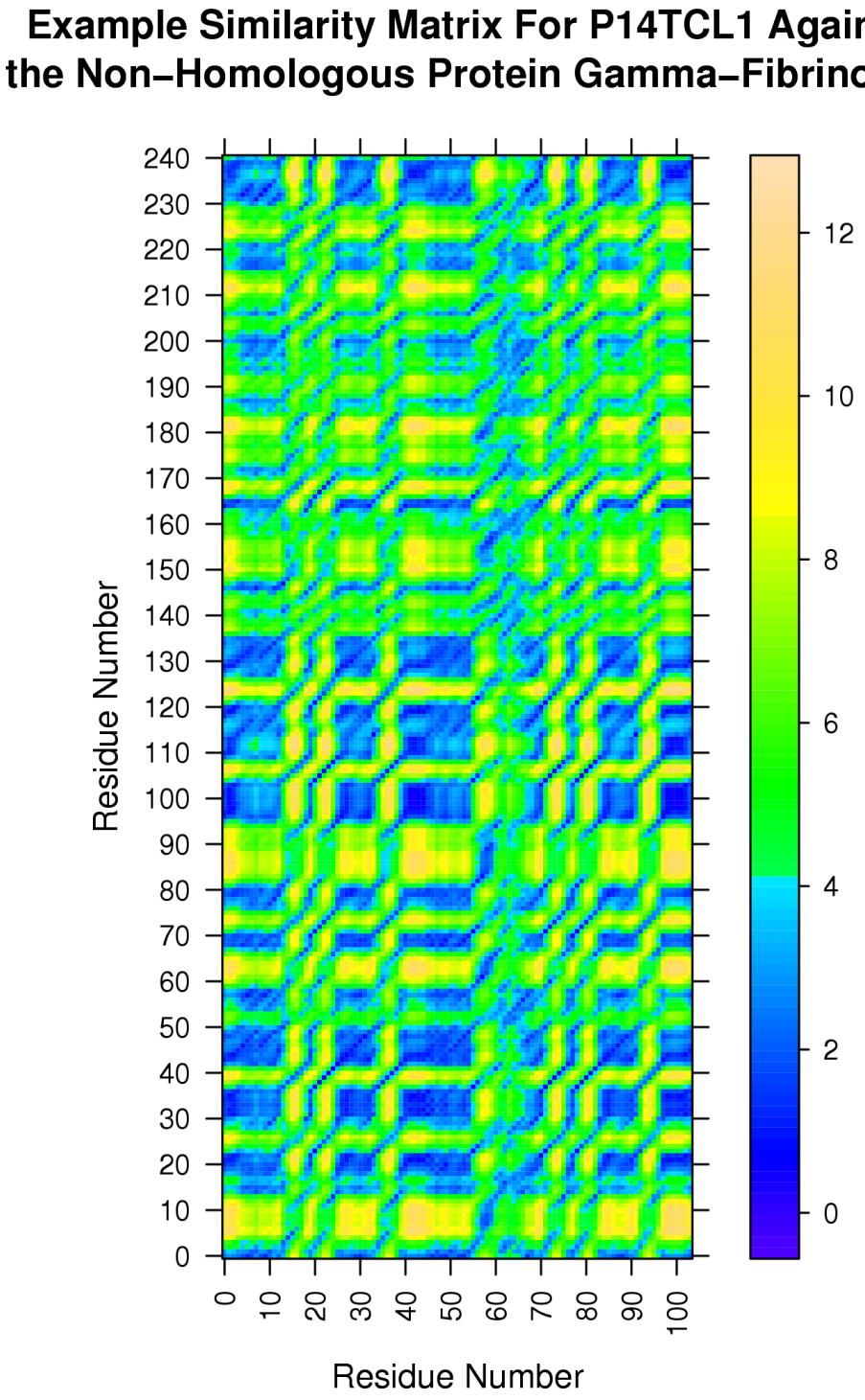


Figure 6.4: P14TCL1 was paired with a much longer protein, Gamma Fibrinogen (PDBID: 3FIB) which is structurally not similar. The dark green boxes do correspond to good structural overlaps, but they're localized (of limited length) due to the repetitive nature of secondary structure.

```
4  double** calcS(double** d1, double** d2, int lenA, int lenB,
       double winSize)
5  {
6    int i;
7
8    // initialize the 2D similarity matrix
9    double** S = malloc(sizeof(double*)*lenA);
10   for ( i = 0; i < lenA; i++ )
11       S[i] = (double*) malloc( sizeof(double)*lenB);
12
13   double sumSize = (winSize-1.0)*(winSize-2.0) / 2.0;
14
15   int iA, iB, row, col;
16   for ( iA = 0; iA < lenA; iA++ ) {
17       for ( iB = 0; iB < lenB; iB++ ) {
18           S[iA][iB] = -1.0;
19
20           if ( iA > lenA - winSize || iB > lenB - winSize )
21               continue;
22
23           double score = 0.0;
24           for ( row = 0; row < (int) winSize - 2; row++ ) {
25               for ( col = row + 2; col < (int) winSize; col++ ) {
26                   score += fabs( d1[iA+row][iA+col] - d2[iB+row][
                         iB+col] );
27               }
28           }
29
30           S[iA][iB] = score / sumSize;
31       }
32   }
33   return S;
34 }
```

The CE Method CE finds the longest path of lowest score through the similarity matrix for two given protein structures. It does this by starting anywhere in the similarity matrix (although $(1, 1)$ is the best choice, as it ensures consideration of the longest possible path first) and comparing that starting entry, let's call it $S_{i,j}$ with a predefined cutoff measure of 3.0 Å. If $S_{i,j} > 3.0$ Å then the substructures

anchored at i and j in proteins A and B, respectively, are not a good match and thus ignored. If this is the case, $S_{i+1,j}$ is considered. Once i is exhausted, or we have gapped 30 times, we reset i and increment j. If $S_{i,j} < 3.0$ Å then that residue pair (i, j) is stored and $(i + 8, j + 8)$ is considered. The algorithm is repeated until the similarity matrix is exhaustively searched. At this point, we have the longest possible, best scoring subsets of residues from the two structures. We can then move on to the next step, optimal alignment.

I have made my variant of CE available for public use. It is on the PyMOL-Wiki (http://www.pymolwiki.org/index.php/Cealign).

To cut down on the running time of this algorithm CE partitions the protein sequence into windows of eight residues, ignores the distances between a residue and its two neighbors. There are other algorithmic shortcuts discussed below. CE ignores the distance between a residue and its two neighboring residues because it adds no information. For example, the distance between an alpha carbon and it's immediate neighbors' alpha carbon is in the range of 3.4–3.8 Å, almost exclusively. See Figure 3.2. Also, any deviation outside that range is hardly biophysically possible or of significant magnitude to make an impact on the calculations, and thus ignored.

Necessary Deviations from the CE Algorithm As noted earlier, the structure of a thermodynamically defined protein does not follow typical structure-based rules. This necessitates investigation of every structure-based assumption that the original authors of CE decided upon. We enumerate the assumptions and necessary changes, here.

First, because sequence neighbors can have large jumps in TE space, as shown in Figure 3.2, during the structure matching step of structure alignment we must consider neighbor-neighbor distances whereas the original CE algorithm does not. Our variant of CE has this feature built back in. Listing 6.2 highlights the change. The indices for counting are changed to include neighbors.

Source Code 6.2: Source Code for Including Neighbor-Neighbor Distance Calculations

```
for ( row = 0; row < (int) winSize; row++ ) {
    for ( col = row; col < (int) winSize; col++ ) {
        score += fabs( d1[iA+row][iA+col] - d2[iB+row][iB+col] );
    }
}
```

Next, due to the extremely large energetic jumps, some normalization of the data is necessary. If not, then the extremely large jumps would dominate structure similarity measures. Our variant of CE was built such that when the distance matrices are being compared to build the similarity matrix, the distance matrices are locally scaled such that the largest distance in each submatrix is 1.0. This allows window-based structure scaling. It increases the running time of the algorithm, but because it is more tolerant of vagaries of scaling, produces more accurate thermodynamic structure alignments. The source code for this change is shown in Listing 6.3 (the original code is in Listing 6.1 on Page 99).

Source Code 6.3: Source Code for Making a Locally Scaled Similarity Matrix

```
// find the max in this submatrix;
double localMaxA = 1.0;
double localMaxB = 1.0;
for ( unsigned int row = 0; row < winSize; row++ ) {
```

```
5     for ( unsigned int col = 0; col < winSize; col++ ) {
6       localMaxA = (d1[iA+row][iA+col] > localMaxA) ? d1[iA+row][iA+
            col] : localMaxA;
7       localMaxB = (d2[iB+row][iB+col] > localMaxB) ? d2[iB+row][iB+
            col] : localMaxB;
8     }
9   }
10
11  // N^2 all against all in the submatrix
12  for ( unsigned int row = 0; row < winSize; row++ ) {
13    for ( unsigned int col = 0; col < winSize; col++ ) {
14      score += fabs( (d1[iA+row][iA+col]/localMaxA) - (d2[iB+row][iB
            +col]/localMaxB) );
15    }
16  }
17
18  S[iA][iB] = score / sumSize;
```

The next place we deviate from CE is during the refinement step. CE calls for a sequence based refinement step. Our project is based solely upon the structure of the thermodynamically defined proteins so we chose not implement the refinement step. The results of CE without the refinement step are still quite relevant (Shindyalov and Bourne, 1998).

The last set of deviations deal with the empirically defined parameters enforced by CE. There is a window size (eight residues was the best determined window size), a maximum gap of 30, a minimum cutoff of 3.0 Å, and a minimum path (overall) cutoff of 4.0 Å. These values were determined from fine-tuning CE against know structure databases. We do not have the luxury of known homology in thermodynamically defined protein structure. We calculated these tunable parameters by iteratively comparing three known structure families and clustering the results. The optimal distance cutoffs discovered were $0.04 * w$, where w is the window size. We set both cutoffs to this value. To find the longest continual substructure, we set

the maximum gap to 0, but increased the window size iteratively from six to twenty six. This allows us to find smaller regions as well as larger.

6.2.4 The Second Step: Determining the Optimal Translation and Rotation to Minimize RMSD

Statement of the Problem Now that we have the two vectors sets of atomic coordinates representing the longest near-optimal alignment of the two proteins, we endeavor to find their optimal structural overlap. Given two N-dimensional vector sets in MDF, $\mathbf{x}_m$ and $\mathbf{y}_m$, of length m, find the best rotation matrix, Q such that

$$\|\mathbf{x}_m - Q\mathbf{y}_m\| \,, \tag{6.4}$$

is minimized. There are many ways to solve this problem exactly. In this project, I use the SVD to solve the linear least squares problem. As shown above, let X be the augmented matrix for $\mathbf{x}_m$ and Y be the augmented matrix for $\mathbf{y}_m$. To solve the problem, first construct the correlation matrix, R of the two matrices X and Y:

$$R = Y^{\perp} X \,. \tag{6.5}$$

R is directly calculated from the two vector sets. The initial error is,

$$\epsilon_0 = \sum_{i=1}^{m} X_i \times X_i + Y_i \times Y_i \,. \tag{6.6}$$

I now decompose the correlation matrix using the Singular Value Decomposition (SVD) theorem. Briefly, the SVD decomposes an $m \times n$ matrix into three special matrices, U, Σ, V. U is the matrix of "left singular vectors," and represents an orthonormal basis for the Columns of R. Σ is the upper diagonal matrix of singular values for R. Finally, V the matrix of "right singular vectors," is an orthonormal basis for the Rows of the correlation matrix. Mathematically,

$$R = U\Sigma V^{\perp} . \tag{6.7}$$

The necessary rotation matrix to superpose the two vector sets is calculated by,

$$Q = UV^{\perp} . \tag{6.8}$$

Because R is symmetric, positive semi-definite, all its eigenvalues are real and its eigenvectors can be made into an orthonormal basis for the its columns. Because V and U are both orthonormal, their product is also orthonormal. This matrix is the orthonormal projection of the Row space onto the Col space of the matrix R, which solves the least squares problem. The Python source code to solve this problem is shown in Listing 6.4.

Source Code 6.4: This is variant of the Kabsch algorithm. This code has been made available on the PyMOLWiki for public use, see http://www.pymolwiki.org/index.php/Kabsch.

```
1 #!python
2 from array import *
3 import numpy
```

```
4
5  def simpAlign( mat1, mat2, name1, name2, mol1=None,\
6          mol2=None, align=0, L=0 ):
7      """
8      optAlign performs the Kabsch alignment algorithm
9        on the two vector sets, mat1 and mat2.
10     """
11
12     # check for consistency
13     assert(len(mat1) == len(mat2))
14
15     # must alway center the two proteins to avoid
16     # affine transformations.  Center the two proteins
17     # to their selections.
18     COM1 = numpy.sum(mat1,axis=0) / float(L)
19     COM2 = numpy.sum(mat2,axis=0) / float(L)
20     mat1 = mat1 - COM1
21     mat2 = mat2 - COM2
22
23     # Initial error.
24     E0 = numpy.sum( numpy.sum(mat1 * mat1,axis=0),axis=0)
25       + numpy.sum( numpy.sum(mat2 * mat2,axis=0),axis=0)
26
27     # Perform the SVD
28     V, S, Wt = numpy.linalg.svd( numpy.dot(numpy.transpose(mat2),
           mat1))
29
30     # Detect reflections
31     reflect = float(str(float(numpy.linalg.det(V) * numpy.linalg.
           det(Wt))))
32     if reflect == -1.0:
33         S[-1] = -S[-1]
34         V[:,-1] = -V[:,-1]
35
36     RMSD = E0 - (2.0 * sum(S))
37     RMSD = numpy.sqrt(abs(RMSD / L))
38
39     if ( align == 0 ):
40         return RMSD;
41
42     #U is simply V*Wt
43     U = numpy.dot(V, Wt)
44
45     # rotate and translate the molecule
46     mat2 = numpy.dot((mat2 - COM2), U) + COM1
47     stored.sel2 = mat2.tolist()
```

This algorithm is general: it works in any number of dimensions, not just 3. In fact, in this project we will be using this for 3D and 4D alignments.

One Interesting Caveat: Reflections One interesting caveat with this method of optimal alignment is the existence of reflections. That is, there may be two (or more) solutions of the problem due to chirality. The simple method to check for a reflection is to recall that the determinate ($\det$) of a matrix is the area of the parallel piped spanned by its vectors. This implies, that area spanned by an orthonormal matrix is always ± 1. Thus, if the linear algebraic system resulted in a reflection, the determinate of either U or V might be -1. Therefore, we check that $\det(U) \times \det(V) \neq -1$. If it is, we have a reflection. To correct for the reflection we just mirror the least significant axis to optimally properly solve the problem.

6.2.5 The SVD of a Similarity Matrix

The SVD of a Similarity Matrix A similarity matrix is a topological decomposition of a four-dimensional metric space down to a two-dimensional metric space. What latent information is encoded in the similarity matrix aside from the optimal path for alignment? Can we find a better—more obvious path? Because path-finding is NPC, this may help us more quickly determine optimal paths.

Let's take a look at the SVD and concomitant $UV^{\perp}$ calculation and meaning. This analysis will follow directly from the above definition, while I look for "meaning" along the way. Let us have two vector sets representing the structure of two arbitrary proteins be $a \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^{p \times n}$. Let $A \in \mathbb{R}^{m \times m}$ and $B \in \mathbb{R}^{p \times p}$ be the

distance matrices for a and b, respectively. Let $S \in \mathbb{R}^{m-w,p-w}$ be the similarity matrix created from A and B. Then, let's calculate the singular value decomposition of S:

$$S = U\Sigma V^{\perp} \tag{6.9}$$

The Singular Values of S First, let's get the singular values and orthonormal basis for the columns of S. Consider the matrix, $S^{\perp}S$, it is symmetric, positive semi-definite, as we know. We also know that we can find an orthonormal eigenbasis for the columns of S using $S^{\perp}S$. So,

$$\begin{aligned} \|Sv_i\|^2 &= (Sv_i)(Sv_i) \\ &= (Sv_i)^{\perp}(Sv_i) \\ &= v_i^{\perp}S^{\perp}Sv_i \\ &= v_i^{\perp}\lambda_i v_i \\ &= \lambda_i \ . \end{aligned} \tag{6.10}$$

What we see is that our singular values can be calculated by multiplying S by the eigenvectors of $S^{\perp}S$, v_i. Because it's projecting into a orthonormal basis, $v_i^{\perp}v_i = \mathcal{I}$ the projection is equal to the amount by which the eigenvectors are scaled. Furthermore, the eigenvector subspace spanned by $\{v_1, v_2, \ldots, v_n\}$ is orthonormal. If we take the square root of Equation 6.10, $Sv_i = \sigma_i u_i$, we see that they are in the span of the columns of S, because it is a linear combination of the columns of S. So, the singular values (σ_i) are the lengths of the projections of the orthonormal eigenvectors v_i through S.

V, the Orthonormal Basis of Col$S$ Let, V (from the previous paragraph) be, $\begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}$. This is clearly an orthonormal matrix. Furthermore, V spans the Col S.

Σ, the diagonal singular values of S Let's now define Σ, the diagonal matrix of singular values of S. So, define $\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$.

U, the orthonormal projection space We now have the similarity matrix S; the orthonormal basis of Col$S$, V; the diagonal singular value matrix Σ; and we need to find U. We start by defining U, then take a look at the properties of U. Let

$$\begin{aligned} u_i &= \frac{1}{\|Sv_i\|} Sv_i \\ &= \frac{1}{\sigma_i} Sv_i, \end{aligned} \tag{6.11}$$

and let $U = \begin{bmatrix} u_1 & u_2 & \cdots & u_r \end{bmatrix}$, where r is the rank of S. That means, if rank $S < m$ then for $k > r$ the projections Sv_k will be in the null space of S. Extend this set such that we just append zeros from $r + 1 \ldots m$. We know that $\|Sv_i\| = \sigma_i$, but we now want the vector itself, Sv_i. So, first off, I need to prove that given an orthonormal matrix U, that the projection of x onto U is Ux and is also orthonormal. Also, U spans the Row S.

Theorem 1 (The image of a vector projected through an orthogonal matrix is also orthogonal)**.** *Prove that for any orthogonal matrix $U \in \mathbb{R}^{n \times n}$ and any vector $x \in R^n$ that Ux is also orthogonal.*

Proof. Let $U \in \mathbb{R}^{n \times n}$ be an orthogonal matrix (the columns of U are all orthogonal). Let $x \in \mathbb{R}^n$. Write U as a collection of row vectors, u_i:

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}.$$

Then,

$$\begin{aligned} Ux &= U^{\perp} \cdot x \\ &= \begin{bmatrix} u_1^T & u_2^T & \cdots u_n^T \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \\ &= \begin{bmatrix} u_1^T x_1 & u_2^T x_2 & \cdots & u_n^T x_n \end{bmatrix}. \end{aligned}$$

Thus, Ux is just the scalar constants times the columns of U. So, scaling the orthogonal vectors does not affect their orthogonality—just their size. Also, if U is orthonormal, it is a projection matrix and in that case, $\|x\| = \|Ux\|$. □

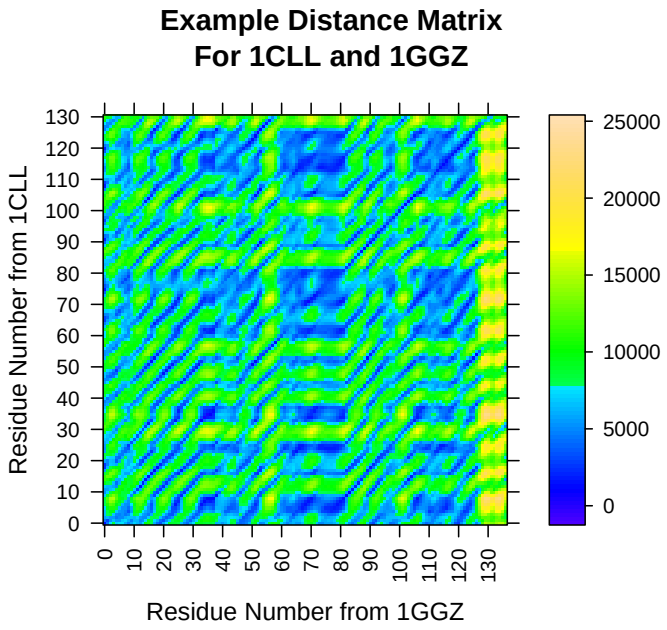
So, u_i are orthonormal (because we scale by $1/\sigma_i$) and as we have shown above all v_i are orthogonal and as just shown above, that indicates Sv_i are also orthogonal and now orthonormal.

$U \times V^{\perp}$, of the SVD of S In a typical least squares solution for a correlation matrix of two equated vector sets the rotation matrix $Q = UV^{\perp}$ is the $n \times n$ rotation matrix that provides the optimal rotation to provide the minimal overlap between the distances of the rows of the vector sets. Here, S is not necessarily square, so what is the meaning of $UV^{\perp}$?

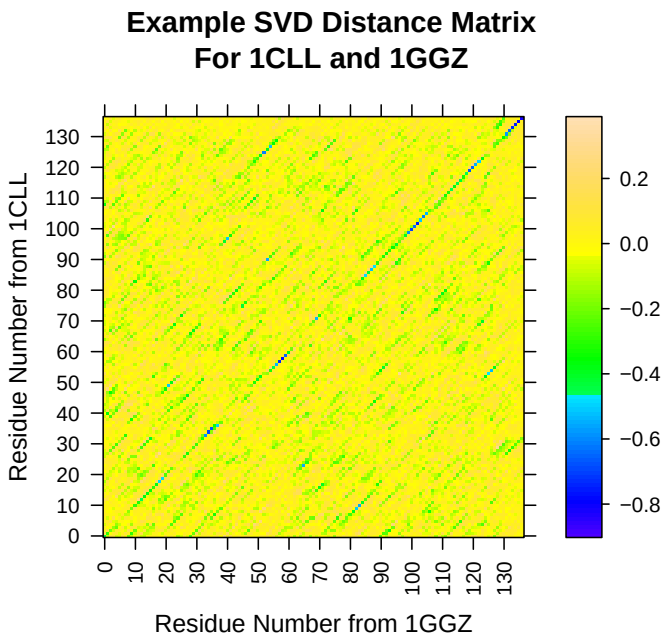
Well, V is the orthonormal (projection matrix) which spans the columns of S. U is the orthonormal (also a projection matrix) that also spans the columns of S.

Because we know that U spans Col A and V spans Row A, thus, $U \cdot V^{\perp}$ is simply the covariance of the two subspaces.

Results This observation is presented here because to my knowledge this has not been performed before and may prove useful for those investigating structure alignment. The result of the multiplication of the U and $V^{\perp}$ is shown in Figure 6.5b and compared to a normal similarity matrix. The similarity matrix shown at the top is from the structural comparison of two homologous proteins (PDBIDs: 1CLL, 1GGZ). There are many possibly good paths through the similarity matrix. At the bottom, is the orthonormal basis for the Row space multiplied by the orthonormal basis for the Column space of the similarity matrix shown at the right. Clearly visible are the elements from the optimal alignment of the two proteins (up the diagonal) as determined by the structure alignment program. The elements of maximal structural overlap are those with the lowest corresponding singular values—which represent the magnitude of covariance between the substructures anchored at those points. This presents the possibility of reducing the number of paths searched before finding the optimal. A variant of CE was produced that used these matrices in



(a)



(b)

Figure 6.5: The Similarity Matrix and the SVD of a Similarity Matrix for Two Structurally Similar Proteins, PDBIDs 1CLL and 1GGZ. The top image shows a similarity matrix calculated for two structurally similar proteins. The optimal alignment runs along the diagonal. This bottom image shows the product of the left and right singular vectors: $UV^{\perp}$ for the top matrix. Notice that the residues with lowest covariance indicate the best substructural alignment, which for this pair is along the diagonal. The diagonal is easier to discern in the bottom image.

place of the typical similarity matrix. The results were promising, but were outside the main scope of this project, so not more actively pursued.

6.3 Determination of Protein Fold Families

Introduction Given the database of protein structures and a method to compare them, we may now consider the organization of the thermodynamic protein fold space. Which proteins are related to which others and why? What families of folds can we identify. Results from this study will provide us with another viewpoint from which to consider the evolutionary descent of proteins.

We used our CE variant in an all-against-all fashion over our *Homo sapiens* database. We then combine this method with agglomerative hierarchical clustering tools to determine the phylogeny. Given the phylogeny, we can make comparisons with the current literature and methods.

The hierarchical clustering techniques we used in this chapter were introduced with all other clustering techniques, in Section 4.1.

6.3.1 Finding the Fold Families

We used our variant of CE, called `steph3`, to compare each thermodynamically defined protein structure to each other in the database of *Homo sapiens* proteins. The BASH script that iterates over the entire database is shown below in Listing 6.5.

Source Code 6.5: BASH Shell Code to Pairwise Compare Each Thermodynamically Defined Protein Structure in our Database.

```
# Use our locally-scaled CE variant, steph3.
S3_COMMAND=~/Projects/steph3/src/steph3

# Loop over all relevant window sizes
for win in `seq 6 26`; do
    # Calculate the cutoff given the windowsize.
    c=`/home/tree/playground/bc.sh 0.040*$win`;
    # Set the options to steph3; all options to steph3 are
    # listed in the Appendix.
    S3_OPTIONS=" --cutoff1=$c --cutoff2=$c --window-size=$win --
        gap-max=0"
    # Define the output file
    OUTFILE=STEPH3.DATA.CBK.w$win;
    # Iterate over all the thermodynamically defined structures
    for x in *.st3; do
        for y in *.st3; do
            # format data to the output file
            echo -n -e `basename $x .st3` >> $OUTFILE
            echo -n -e '\t' >> $OUTFILE
            echo -n -e `basename $y .st3` >> $OUTFILE
            echo -n -e '\t' >> $OUTFILE
            # head -7 => CE_SCORE
            $S3_COMMAND $S3_OPTIONS -1 $x -2 $y | head -7 | tail
                -1 | cut -f2 -d" " >> $OUTFILE
        done;
    done;
done;
```

This provides us with a dissimilarity matrix. Now each protein can be thought of as a point in some high dimensional space. We now desire to divide the space into mutually exclusive regions to define the fold families. Using the `hclust` function with Ward's method in R, we can cluster the dissimilarity matrix.

Because of the nature of TE space, we chose to restrict gapping to 0. If a maximum gap of 30, like with the original CE algorithm, is used the alignments are fragmented: many smaller local alignments across the structures are returned.

We desire the longest continuous thermodynamic structure alignment, so we set the `steph3` parameter "gap_max" to 0. To overcome this and find longer alignment, we then iterate over window size of 6 to 26 searching for optimal alignments. The results of each iteration are turned into a distance matrix for that window size. Then, each distance matrix is averaged into a global distance matrix and that matrix is clustered. Specifically, for each window size we apply the following binary mask to its alignment scores matrix,

$$M_{i,j} = \begin{cases} 0 & (i,j) \leq \text{25th quantile} , \\ 1 & \text{otherwise} . \end{cases} \tag{6.12}$$

If a protein is "close" to another (within the 25th quantile) then its distance is to be set to 0. If a protein is not "close" its distance is set to 1. Therefore, for each window size we are keeping the specific relationship information and disregarding the noise. The 25th quantile was chosen to keep some longer distances and because if only comparing a a few families of proteins the 5th quantile would only return such a small subset of proteins to not be a useful metric. Next, we add the binary-masked matrices into an averaged matrix for distance representation. This average distance matrix was then clustered using `hclust` and Ward's algorithm. The source code, in the R language, is shown in Listing 6.6. I will show that this iterative method and binary mask are valid techniques in the next section.

Source Code 6.6: R Source Code for Creating the Global Distance Matrix for Clustering.

```
# define the helper function, inF.  It reads in a file
# from disk that is known to be a square distance matrix.
# It then labels the rows and columns from via the
# data in the first column.
inF <- function( fName ) {
        t_t <- data.frame( read.table( fName ));
        t_t <- matrix( t_t[,3], nrow=sqrt(nrow(t_t)), ncol=sqrt(
            nrow(t_t)), dimnames=list( levels( t_t[,1]), levels( t
            _t[,1])));
        return(t_t);
}

#
# Begin code for average global matrix
#

# clean up in case not first execution
if ( exists("allData")) { rm("allData"); }

# set default min and max window sizes
if (!exists("minW")) { minW <- 6; }
if (!exists("maxW")) { maxW <- 26; }

for ( win in minW:maxW ) {
    # make the filename to read from disk
    inFileName=paste("STEPH3.DATA.w", win, sep="");
    # read the filename.
    inD <- inF(inFileName);
    # initialize the matrix
    if ( ! exists("allData")) {
        allData <- matrix( nrow=nrow(inD), ncol=ncol(inD),
            dimnames=dimnames(inD), 0);
    }

    inD[ which( inD > quantile( inD, 0.25, na.rm=TRUE )) ] <- NA
    # get only the best scores and make the mask
    inD[ which( ! is.na(inD)) ] <- 0;
    inD[ which(   is.na(inD)) ] <- 1;

    allData <- allData + inD;
    rm(inD);
}

plot( hclust( as.dist(allData), method="ward"), main="HCLUST of
    the Average Distance Matrix");
```

6.3.2 Testing the Method

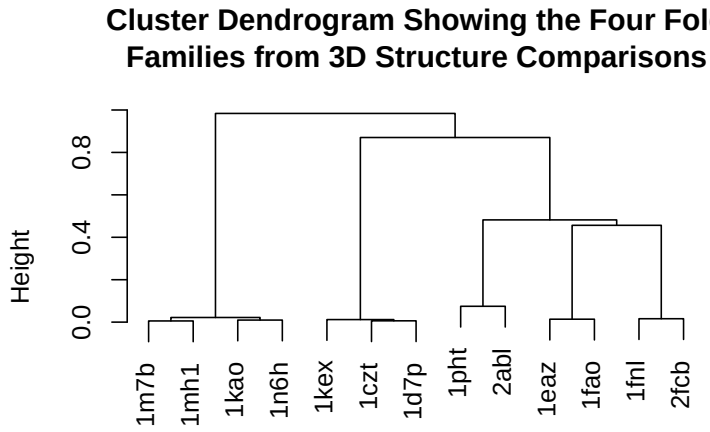
To test the methods, I have applied our iterative CE variant to a data set of five small families of homologous proteins. I tested this method on both the typical 3D protein structures as well as our thermodynamically defined structures. The results from the typical structure-based and thermodynamic structure-based are shown in Figure 6.6.

6.3.3 Applying the Method

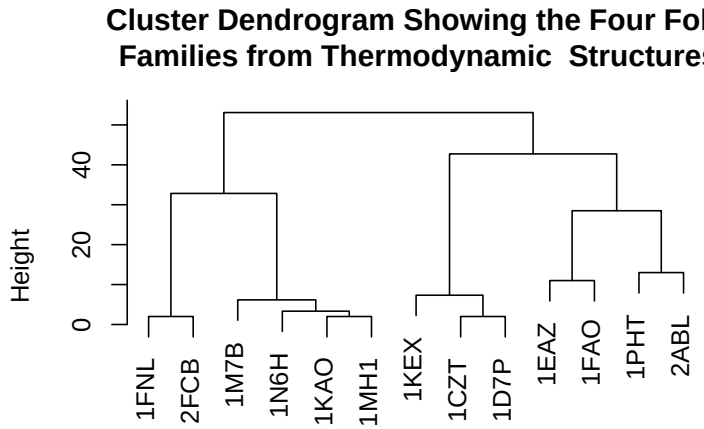
We pairwise compared all the original 3D protein structures in our database. We then iteratively compared the thermodynamically defined protein structures, as well. We provide the results and discussion for a subset of the proteins, and provide the results for the clustering of the entire set of proteins. The results from the clustering of the entire database are promising, but muddied due to the existence some proteins with no structural or thermodynamic neighbors. The smaller subset was of four randomly chosen protein families from the database.

Hidden Dual Nature The major difference between the arrangement of the structure-based hierarchy and the thermodynamically-defined hierarchy is the location of the 1FNL/2FCB branch. In typical structural terms it has been determined to have some propinquity with the folds on the right side of the tree. The results from our clustering show the branch moving to the opposite side of the tree. Why would that happen? Can we explain it?

First, inspection of the optimal alignments and scores, shows that the 1FNL/2FCB



(a) Results of Hierarchically Clustering Four Protein Families Based on 3D Structure.



(b) Results of Hierarchically Clustering Four Protein Families Based on Their Thermodynamically Defined Structure

Figure 6.6: Notice that the four protein families were indeed recovered through the method. Of interesting note is that through our thermodynamic structure alignment, we gained the same fold family classifications. This implies two important results that are discussed in Section 6.3.3.

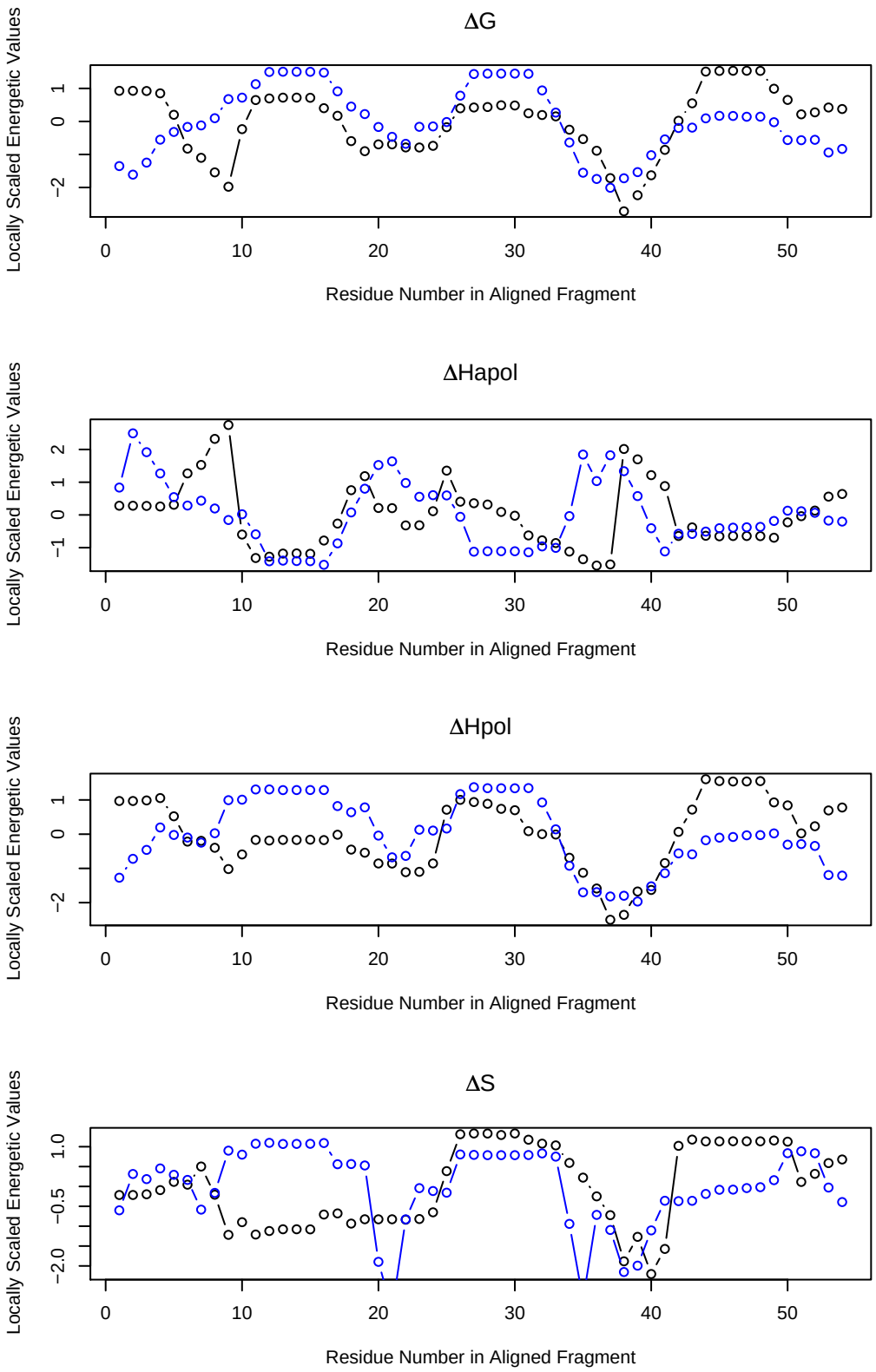


Figure 6.7: Figure Showing Ensemble Averaged Energetic Parameters for Two High Scoring Segments from 1FNL 113:166 and 1M7B 18:71

branch scores well thermodynamically with the 1M7B brach. Figure 6.7 indicates, albeit subtly, why. The overlapping segments are residues 113–166 in 1FNL and 18-71 in 1M7B. In Figures 6.7 and 1.1 these two regions are aligned and their residues renumbered from 1 to more easily discuss the overlapping features. The overlap generally looks acceptable. There are two regions worth focusing on. The first is the region spanning residues 10–18. The second region spans the residues 35–42.

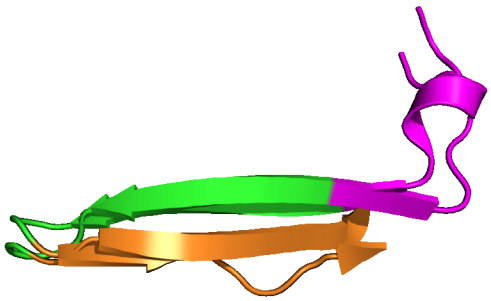
The first region shows high overlap on the apolar enthalpy dimension, with some variation on the polar enthalpy dimension. If one looks at the structures of these two proteins at these locations we can see what their features are. Figure XYZ shows the two aligned fragments and hightlights the residues 10–18. The region from PROTA is consists mainly of apolar residues in a loop fully exposed to solvent. Thus, upon unfolding of this region, the contribution will be a relatively low amount of polar surface area (the residues are mainly apolar) with a relatively low amount of apolar complementary surface area being exposed as well. This effectively balances the apolar to polar enthalpy ratio—low direct polar exposure, low complementary apolar exposure. Thus, the apolar enthalpy dimension for the two proteins should be relatively similar throughout this region, thus the overlap of apolar enthalpy. This also explains why the entropy values for this region are vastly different: consider their locations in structure and secondary structure.

The next region of importance is the span of residues 35–42. This region in both proteins shares a similar polar enthalpy. This turns out to be similar to the reason why the residues 10–18 matched.

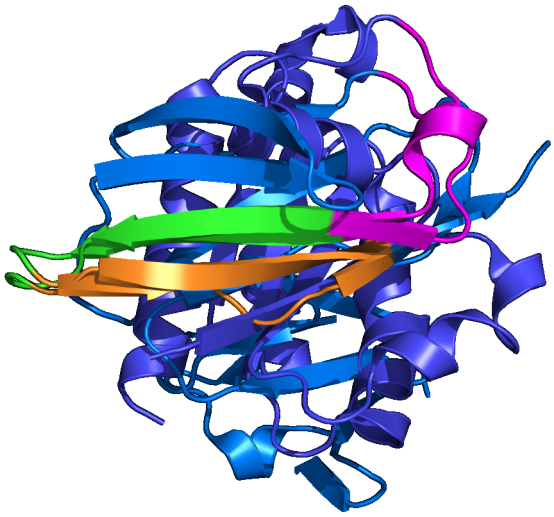
Thus, exploring these two subsequences shows that proteins have at their disposal various combinations of methods by which they can modulate their stability, enthalpy and entropy values even though it would not appear as such when considering the tenets of, "helices and sheets are stable and loops are unstable."

Conserved Structural Features We observe common significant substructural overlap of two proteins with no known homology. For example, consider the 1FNL and 1M7B proteins. They have a significant substructural overlap of 24 residues determined from our thermodynamic data alone. Figure shows the two proteins with their thermodynamically aligned segments highlighted. Notice when superposed they also align very well (3.14 Å RMSD) structurally.

Coil/Loop Anchored Topologies The second observation for the two overlapping structures, is that they have a common feature: they have loops or coils in the same (or nearby) places in sequence. This hints at an underlying principal of TE alignments being anchored at loops/coils with the differing intervening secondary structural elements. If the loop/coils align well then the secondary structural elements in between are there because of their respective topological features not their type of secondary structure. Figure 6.9 shows a colored coded example from the above alignment. This should not be too surprising as we know that coils and loops are typically highly solvent accessible. This implies that loops and coils should energetically similar with respect to other secondary structure types. This is not the case for other secondary structures.



(a)



(b)

Figure 6.8: The top figure (a) shows the structural alignment of the highly similar TE environments from two non-homologous proteins. The alignment is colored magenta, green and orange to set the aligned regions apart. The RMSD for the colored overlapping alignments is 3.14 Angstroms. This alignment occurs many times through our iterative alignment process. The bottom figure (b) is the same alignment but with the entire protein structures shown. Notice that the good TE matched regions actually are structurally similar, but come from structurally nonsimilar proteins.

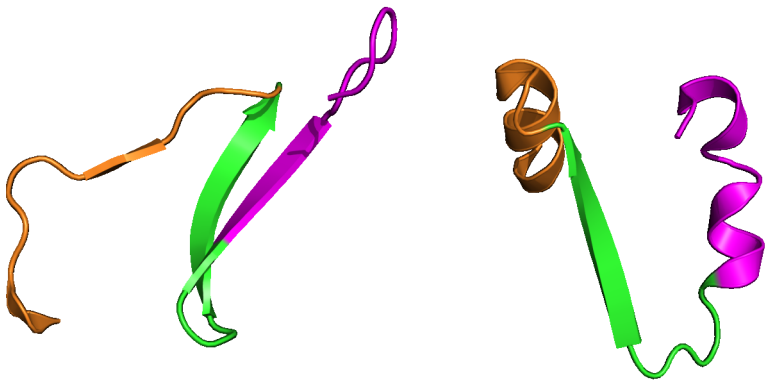


Figure 6.9: Changing Alignments Happens Near Coils/Turns. Notice how the color regions of the alignments change near the loop and coil secondary structure.

Chapter 7

Conclusions and Future Directions

7.1 Thermodynamically Defined Protein Folds: A New Vehicle for Reasoning About Proteins

Characterization of TE Space PCA was used to gain insight into the organization of TE space. We found that certain directions in TE space (PC1) are defined by the magnitude of the unfolding event. We also discovered that PC2 is most directly related to the type of surface area being unfolded. Lastly, PC3 was shown to be most related to a stability and entropy change.

Each point in the TE space is the result of a difference of weighted differences. When combined with the primary, secondary and tertiary structure information of the protein, these values logically define the TE space.

Residues in TE space are stable because their unfolded subensembles expose a large amount of accessible apolar and polar surface area. This large amount of

exposed surface area lowers the probability of these microstates to nearly 0. Once weighted, the majority of the energetic signal of stable residues comes from the dominant, folded subensemble. The unfolded subensemble for unstable residues is weighted more favorably because it exposes less surface area.

Structure Alignment and Fold Family Determination We have shown that we can identify similar thermodynamically defined protein structures. We have also shown that we can hierarchically cluster and partition the fold space, and that this fold space is organized differently that is the structure fold space. These results indicate a method of looking at the protein fold space from a new point of view—an original goal of this project.

Investigation of the results of the TE structure alignments has shown that (1) similar energetic substructures can be found in proteins with no known homology; (2) that in some cases these regions have high typical-structural agreement; and (3) loops and coils may actually be anchor points for residue-residue pairings.

Targeted Residue Mutations One interesting result gained from these theoretical experiments is the correlation between calculable ΔASA and stability. Because of this I can propose a simple method for residue mutations based upon ΔASA. First, calculate the residue-level energetics for a protein. To mutate the residue follow the results from PC1 or PC2 for stability and surface area changes. Also, consider mutating distal residues that are positively coupled. Next, more fine-grained results can be calculated if the secondary structure log-odds probabilities are considered. That is, a PSSM can be created for each different type of secondary

structure to better pinpoint the required changes.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic Local Alignment Search Tool. *J. Mol. Biol.*, 215:403–410.

Bateman, A., Birney, E., Cerruti, L., Durbin, R., Etwiller, L., Eddy, S. R., Griffiths-Jones, S., Howe, K. L., Marshall, M., and Sonnhammer, E. L. L. (2002). The Pfam Protein Families Database. *Nucl. Acids Res.*, 30:276–280.

Baum, J., Dobson, C. M., Evans, P. A., and Hanley, C. (1989). Characterization of a Partly Folded Protein by NMR Methods: Studies on the Molten Globule State of Guinea Pig α-Lactalbumin. *Biochemistry*, 28:7–13.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000). The Protein Data Bank. *Nucl. Acids Res.*, 28:235–242.

Braak, C. J. F. T. (1986). Canonical correspondence analysis: A new eigenvector technique for multivariate direct gradient analysis. *Ecology*, 67(5):1167–1179.

Brenner, S. E., Koehl, P., and Levitt, M. (2000). The astral compendium for protein structure and sequence analysis. *Nucl. Acid. Res.*, 28(1):254–256.

Chen, L., Zhou, T., and Tang, Y. (2005). Protein structure alignment by deterministic annealing. *Bioinformatics*, 21(1):51–62.

Chothia, C. and Levitt, M. (1977). Structure of proteins: Packing of a-helices and pleated sheets. *Proc. Natl. Acad. Sci.*, 74(10):4130–4134.

Coutsias, E. A., Seok, C., and Dill, K. A. (2003). Using Quaternions to Calculate RMSD. *J. Comput. Chem.*, 25(1):56–51.

D'Aquino, J. A., Gomez, J., Hilser, V. J., Lee, K. H., Amzel, L. M., and Freire, E. (1996). The Magnitude of the Backbone Conformational Entropy Change in Protein Folding. *Protein-Struct. Funct. Genet.*, 25:143–156.

Day, R., Beck, D. A. C., Armen, R. S., and Daggett, V. (2003). A consensus view of fold space: Combining SCOP, CATH, and the Dali Domain Dictionary. *Protein Sci.*, 12:2150–2160.

DeLano, W. L. (2002). The PyMOL Molecular Graphics System.

DeSa, R. J. and Matheson, I. B. C. (2004). A Practical Approach to Interpretation of Singular Value Decomposition Results. *Methods Enzymol.*, 384:1–8.

Domìnguez, R., Souchon, H., Lascombe, M.-B., and Alzari, P. M. (1996). The crystal structure of a family 5 endoglucanase mutant in complexed and uncomplexed forms reveals an induced fit activation mechanism. *J. Mol. Biol.*, 257:1042–1051.

Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, 95:14863–14868.

Freire, E. and Biltonen, R. L. (1978). Statistical mechanical deconvolution of thermal transitions in macromolecules. I. Theory and application to homogeneous systems. *Biopolymers*, 17(2):463–479.

Freire, E. and Murphy, K. P. (1991). Molecular basis of co-operativity in protein folding. *J. Mol. Biol.*, 222:687–698.

Frishman, D. and Argos, P. (1995). Knowledge-based protein secondary structure assignment. *Proteins*, 23:566–579.

Godzik, A. (1996). The structural alignment between two proteins: Is there a unique answer? *Protein Sci.*, 53:1325–1338.

Goldman, D., Istrail, S., and Papadimitriou, C. H. (1999). Algorithmic Aspects of Protein Structure Similarity. *Found. of Comp. Sci.*, 1999:512–522.

Gribskov, M., McLachlan, A. D., and Eisenberg, D. (1987). Profile analysis: Detection of distantly related proteins. *Proc. Natl. Acad. Sci. USA*, 84:4355–4358.

Guda, C., Lu, S., Scheeff, E. D., Bourne, P. E., and Shindyalov, I. N. (2004). CE-MC: a multiple protein structure alignment server. *Nucl. Acids Res.*, 32.

Guess, M. J. and Wilson, S. B. (2002). Introduction to Hierarchical Clustering. *J. Clin. Neurophys.*, 19(2):144–151.

Hand, D. J. (1997). Data Mining: Statistics and More. *The American Statistician*, 52(2).

Hand, D. J. (1999). Statistics and Data Mining: Intersecting Disciplines. *SIGKDD Explorations*, 1(1):16–19.

Henikoff, S. and Henikoff, J. G. (1992). Amino Acid Substitution Matrices from Protein Blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919.

Henry, E. R. and Hofrichter, J. (1992). Singular value decomposition: Application to analysis of experimental data. 210:129–192.

Hill, M. O. (1974). Correspondence analysis: A neglected multivariate method. *Appl. Stat.*, 23(3):340–354.

Hilser, V. J. and Freire, E. (1996). Structure-based Calculation of the Equilibrium Folding Pathway of Proteins. Correlation with Hydrogen Exchange Protection Factors. *J. Mol. Biol.*, 262:756–772.

Hilser, V. J., Gomez, J., and Freire, E. (1996). The Enthalpy Change in Protein Folding and Binding: Refinement of Parameters for Structure-Based Calculations. *Protein-Struct. Funct. Genet.*, 26:123–133.

Holm, L. and Sander, C. (1993). Protein Structure Comparison by Alignment of Distance Matrices. *J. Mol. Biol.*, 233:123–138.

Holm, L. and Sander, C. (1996). Mapping the Protein Universe. *Science*, 273(5275):595–602.

Holm, L. and Sander, C. (1999). Protein folds and families: sequence and structure alignments. *Nucl. Acids Res.*, 27:244–247.

Honig, B. (2007). Protein structure space is much more than the sum of its folds. *Nat. Struct. Mol. Biol.*, 14(6).

Hou, J., Sims, G. E., Zhang, C., and Kim, S.-H. (2003). A global representation of the protein fold space. *Proc. Natl. Acad. Sci. USA*, 100:2386–2390.

Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics. *J. of Comp. and Graph. Stat.*, 5(3):299–314.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323.

Jia, Y., Dewey, T. G., Shindyalov, I. N., and Bourne, P. E. (2004). A New Scoring Function and Associated Statistical Signicance for Structure Alignment by CE. *J. Comp. Biol.*, 11(5):787–799.

Kabsch, W. (1976). A solution for the best rotation to relate two vector sets. *Acta Cryst. Sec. A*, 32:922–923.

Kabsch, W. (1978). A discussion of the solution for the best rotation to relate two vector sets. *Acta Cryst. Sec. A*, 34A:827–828.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York.

Kolodny, R., Petrey, D., and Honig, B. (2006). Protein structure comparison: implications for the nature of fold space, and structure and function prediction. *Curr. Opin. Struct. Biol.*, 16:393–398.

Larson, S. A. and Hilser, V. J. (2004). Analysis of the Thermodynamic Information Content of a Homo sapiens structural database reveals hierarchical thermodynamic organization. *Protein Sci.*, 13:1787–1801.

Lay, D. C. (2003). *Linear Algebra and Its Applications*.

Liu, T., Whitten, S. T., and Hilser, V. J. (2006). Ensemble-Based Signatures of Energy Propagation in Proteins: A New View of an Old Phenomenon. *Proteins*, 62:728–738.

Metropolis, N. and Ulam, S. (1949). The monte carlo method. *J Am Stat Assoc*, 44:335–341.

Murphy, K. P. and Freire, E. (1992). Thermodynamics of structural stability and cooperative folding behavior in proteins. *Adv. Prot. Chem.*, 43:313–361.

Murphy, K. P. and Gill, S. J. (1990). Group additivity thermodynamics for dissolution of solid cyclic dipeptides into water. *Thermochim. Acta*, 172:ll–20.

Murphy, K. P. and Gill, S. J. (1991). Solid model compounds and the thermodynamics of protein unfolding. *J. Mol. Biol.*, 3:699–709.

Murphy, K. P., Privalov, P. L., and Gill, S. J. (1990). Common features of protein unfolding and dissolution of hydrophobic compounds. *Science*, 247(4942):559–561.

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. *Adv. Protein Chem.*, 48(3):443–453.

Ponomarenko, J. V., Bourne, P. E., and Shindyalov, I. N. (2005). Assigning New GO Annotations to Protein Data Bank Sequences by Combining Structure and Sequence Homology. *Proteins*, 58:855–865.

Qian, J., Luscombe, N. M., and Gerstein, M. (2001). Protein Family and Fold Occurrence in Genomes: Power-law Behaviour and Evolutionary Model. *J. Mol. Biol.*, 313:673–681.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Schrödinger, Erwin (1952). *Statistical Thermodynamics*. Dover.

Schwartz, R. M. and Dayhoff, M. O. (1979). Protein and Nucleic Acid Sequence Data and Phylogeny. *Science*, 205(4410):1038–1039.

Shindyalov, I. N. and Bourne, P. E. (1998). Combinatorial Extension (CE) using a Composite Property Description. A New Approach to 3-D Structure Alignment and its Application to the Protein Kinase Family. *Proteins*, 11:739–747.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197.

Szustakowski, J. D. and Weng, Z. (2000). Protein Structure Alignment Using a Genetic Algorithm. *Protein-Struct. Funct. Genet.*, 38:428–440.

Taylor, W. R. (1999). Protein structure comparison using iterated double dynamic programming. *Protein Sci.*, 8:654–665.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.*, 22:4673–4680.

Urzhumtsev, A., Tête-Favier, F., Mitschler, A., Barbanton, J., Barth, P., Urzhumtseva, L., Biellmann, J. F., Podjarny, A., and Moras, D. (1997). A 'specificity' pocket inferred from the crystal structures of the complexes of aldose reductase with the pharmaceutically important inhibitors tolrestat and sorbinil. *Structure*, 5:601–612.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244.

Whitten, S. T., Garca-Moreno, B., and Hilser, V. J. (2005). Local conformational fluctuations can modulate the coupling between proton binding and global structural transitions in proteins. *Proc. Natl. Acad. Sci. USA*, 102:4282–4287.

Wrabl, J. O., Larson, S. A., and Hilser, V. J. (2001). Thermodynamic propensities of amino acids in the native state ensemble: Implications for fold recognition. *Protein Sci.*, 10:1032–1045.

Wrabl, J. O., Larson, S. A., and Hilser, V. J. (2002). Thermodynamic environments

in proteins: Fundamental determinants of fold specificity. *Protein Sci.*, 11:1945–1957.

Xie, D. and Freire, E. (1994). Structure Based Prediction of Protein Folding Intermediates. *J. Mol. Biol.*, 242:62–80.

Zha, H., He, X., Ding, C., and Simon, H. (2002). Spectral Relaxation for K-means Clustering. *NIPS*, 10:1299–1219.

Zhao, L., Cox, A. G., Ruzicka, J. A., Bhat, A. A., Zhang, W., and Taylor, E. W. (2000). Molecular modeling and in vitro activity of an HIV-1-encoded glutathione peroxidase. *Proc. Natl. Acad. Sci. USA*, 97:6356–6361.

Appendix A

C++ Source Code for the CE Variant

A.1 Our Original CE Variant

Source Code A.1: C++ Source Code for Our CE Variant

```
1  /*************************************************
2   *            CE.cpp
3   *
4   *  Mon Apr 24 09:47:54 2006
5   *  Copyright (C) 2006-2007 Jason Vertrees
6   *  javertre@utmb.edu
7   *  $Id: CE.cpp 12 2006-12-21 23:49:14Z tree $
8   *************************************************/
9  #include "CE.h"
10 #include <tnt_array2d.h>
11
12 #include <float.h>
13 #include <sstream>
14 #include <iomanip>
15 #include "QKabsch.h"
16
17 namespace CE {
18
19   //========================================
20   // Class typicals (cstrs, dstr, etc.)
```

```
21  //=========================================
22  CE::CE():
23    coordsA(*(new Coords())),
24    coordsB(*(new Coords())),
25    dmA(*(new DM())),
26    dmB(*(new DM())),
27    S(*(new SM(0,0))),
28    paths(),
29    D0(3.0),
30    D1(4.0),
31    winSize(8),
32    winSum((winSize-2)*(winSize-1)/2),
33    gapMax(30)
34  {}
35
36    //FIXME: ADD NEW options to CE and test
37
38  //=============================================
39  // CE copy constructor
40  //=============================================
41  CE::CE( const CE& rhs ):
42    coordsA( rhs.coordsA ),
43    coordsB( rhs.coordsB ),
44    dmA(rhs.dmA),
45    dmB(rhs.dmB),
46    S( *(new SM(rhs.S)) ),
47    paths(),
48    D0(rhs.D0),
49    D1(rhs.D1),
50    winSize(rhs.winSize),
51    winSum(rhs.winSum),
52    gapMax(rhs.gapMax)
53  {}
54
55  //=============================================
56  // CE constructor from 2 coordinate sets
57  //=============================================
58  CE::CE( const Coords& c1,
59      const Coords& c2,
60      double D0,
61      double D1,
62      int winSize,
63      int winSum,
64      int gapMax  ):
65    coordsA(*(new Coords(c1))),
66    coordsB(*(new Coords(c2))),
```

```
67      dmA(*(new DM(c1))),
68      dmB(*(new DM(c2))),
69      S(*(new SM(c1.size(), c2.size()))),
70      paths(),
71      D0(D0),
72      D1(D1),
73      winSize(winSize),
74      winSum(winSum),
75      gapMax(gapMax)
76    {
77      assert( (unsigned int) coordsA.size() == dmA.getDim1() );
78      assert( (unsigned int) coordsB.size() == dmB.getDim1() );
79      calcS();
80    }
81
82    //============================================
83    // CE constructor from 2 distance matrices
84    //============================================
85    CE::CE( const DM& dmA,
86        const DM& dmB,
87        double D0,
88        double D1,
89        int winSize,
90        int winSum,
91        int gapMax ):
92      coordsA( *(new Coords())),
93      coordsB( *(new Coords())),
94      dmA(*(new DM(dmA))),
95      dmB(*(new DM(dmB))),
96      S(*(new SM(dmA.getDim1(), dmB.getDim1()))),
97      paths(),
98      D0(D0),
99      D1(D1),
100     winSize(winSize),
101     winSum(winSum),
102     gapMax(gapMax)
103   {
104     calcS();
105   }
106
107   //============================================
108   // CEs Destructor
109   //============================================
110   CE::~CE()
111   {
112     delete &dmA;
```

```
113       delete &dmB;
114       delete &coordsA;
115       delete &coordsB;
116       delete &S;
117     }
118
119
120
121     //============================================
122     // gets coordinates of the atoms that will be aligned
123     //============================================
124     std::pair<Coords, Coords> CE::getAlignedCoords( unsigned int
            pathNo ) const
125     {
126       assert( paths.size() > 0 );
127       assert( pathNo < paths.size() );
128       assert( coordsA.size()>0 );
129       assert( coordsB.size()>0 );
130       assert( (unsigned int) coordsA.size() == dmA.getDim1() );
131       assert( (unsigned int) coordsB.size() == dmB.getDim1() );
132
133       if ( paths[pathNo].size() == 0 ) {
134         return std::pair<Coords,Coords>();
135       }
136
137       // get the path
138       Path path = paths[pathNo];
139
140       // get the true size of the path
141       unsigned int maxPathSize = 0;
142       while ( maxPathSize < path.size() )
143         if ( path[(maxPathSize++)+1].first == -1 )
144           break;
145
146       Coords rC1( coordsA );
147       Coords rC2( coordsB );
148
149       typedef std::set<unsigned int> uintSet;
150       uintSet incA;
151       uintSet incB;
152       // remove those to align from erasure
153       for ( unsigned int i = 0; i < maxPathSize; i++ ) {
154         for ( unsigned int j = 0; j < winSize; j++ ) {
155           incA.insert( path[i].first+j );
156           incB.insert( path[i].second+j );
157         }
```

```
158       }
159
160       rC1.setSlice(incA);
161       rC2.setSlice(incB);
162       assert( rC1.size() == rC2.size() );
163
164       return std::pair<Coords, Coords>( rC1,rC2 );
165     }
166
167
168     //================================================
169     // main public alignment algorithm; the first will operator on a
170     // well-constructed CE class using its own members.  The two-
          param
171     // version will work on two givem Distance Matrices
172     //================================================
173     void CE::align() {
174       align(this->dmA, this->dmB);
175     }
176
177     //================================================
178     // Main invocation for CE, just converts arguments to DMs.
179     //================================================
180     void CE::align( const Coords& c1, const Coords& c2 ) {
181       align( DM(c1), DM(c2) );
182     }
183
184     void CE::align( const DM& dmA, const DM& dmB ) {
185
186       assert( dmA.getDim1() == (unsigned int) coordsA.size() );
187       assert( dmB.getDim1() == (unsigned int) coordsB.size() );
188
189       // calculate the Similarity matrix
190       calcS();
191
192       assert( dmA.getDim1() == (unsigned int) coordsA.size() );
193       assert( dmB.getDim1() == (unsigned int) coordsB.size() );
194
195       //find the optimal path through S
196       findPath();
197
198       // align the paths to get the best structure
199       findOptimalPath();
200     }
201
202     //================================================
```

```
203   // Calculates the Similarity matrix, S.
204   //=====================================================
205   void CE::calcS() {
206     double sumSize = (winSize-1)*(winSize-2) / 2;
207     double score = 0.0;
208     unsigned int lenA = dmA.getDim1(); //- winSize;
209     unsigned int lenB = dmB.getDim1(); // - winSize;
210
211     VVD d1 = *(dmA.getData());
212     VVD d2 = *(dmB.getData());
213     //
214     // This is where the magic of CE comes out.  In the similarity
          matrix,
215     // for each i and j, the value of ceSIM[i][j] is how well the
          residues
216     // i - i+winSize in protein A, match to residues j - j+winSize
          in protein
217     // B.  A value of 0 means absolute match; a value >> 1 means
          bad match.
218     //
219     for ( unsigned int iA = 0; iA < lenA; iA++ ) {
220       for ( unsigned int iB = 0; iB < lenB; iB++ ) {
221         S[iA][iB] = -1.0;
222
223         // the boundary of -1's are needed.
224         if ( iA > lenA-winSize || iB > lenB-winSize )
225           continue;
226         score = 0.0;
227
228         //
229         // We always skip the calculation of the distance from
              THIS
230         // residues, to the next residue.  This is a time-saving
              heur-
231         // istic decision.  Almost all alpha carbon bonds of
              neighboring
232         // residues is 3.8 Angstroms.  Due to entropy, S = -k ln
              pi * pi,
233         // this tell us nothing, so it doesn't help so ignore it.
234         //
235
236         // UNCOMMENT BELOW FOR ORIGINAL ALGORITHM
237 /*          for ( unsigned int row = 0; row < winSize - 2; row++ )
      {
238             for ( unsigned int col = row + 2; col < winSize; col++
                  ) {
```

```
239                 score += fabs( d1[iA+row][iA+col] - d2[iB+row][iB+
                        col] );
240               }
241             }
242   */
243
244         // FIXME!!!!!!! The above works well for standard protein
              data
245         // FIXME!!!!!!! but, we don't have standard data, so I
              need to
246         // FIXME!!!!!!! start from the neighboring residues
              afterall.
247
248         // find the max in this submatrix;
249         double localMaxA = 1.0;
250         double localMaxB = 1.0;
251         for ( unsigned int row = 0; row < winSize; row++ ) {
252           for ( unsigned int col = 0; col < winSize; col++ ) {
253             localMaxA = ( d1[iA+row][iA+col] > localMaxA ) ? d1[iA
                    +row][iA+col] : localMaxA;
254             localMaxB = ( d2[iB+row][iB+col] > localMaxB ) ? d2[iB
                    +row][iB+col] : localMaxB;
255           }
256         }
257
258         // N^2 all against all in the submatrix
259         for ( unsigned int row = 0; row < winSize; row++ ) {
260           for ( unsigned int col = 0; col < winSize; col++ ) {
261             score += fabs( (d1[iA+row][iA+col]/localMaxA) - (d2[iB
                    +row][iB+col]/localMaxB) );
262           }
263         }
264
265         S[iA][iB] = score / sumSize;
266       }
267     }
268
269     // dump an S matrix to image
270     if ( args["verbose"] == "true" ) {
271       for ( int i = 0; i < S.dim1(); i++ ) {
272         if ( (unsigned) i > lenA-winSize ) break;
273         for ( int j = 0; j < S.dim2(); j++ ) {
274           if ( (unsigned) j > lenB-winSize ) break;
275           std::cout << std::setprecision(5) << S[i][j] << "\t";
276         }
277         std::cout << "\n";
```

```
278       }
279     }
280   }
281
282   //================================================
283   // finds the all paths through S, the similarity matrix
284   //================================================
285   void CE::findPath() {
286
287     // for neatness; boundaries
288     const unsigned int lenA = dmA.getDim1();
289     const unsigned int lenB = dmB.getDim1();
290
291     VVD dA = *(dmA.getData());
292     VVD dB = *(dmB.getData());
293
294     // create the holder for the BEST path
295     int maxPathLength = std::min( lenA, lenB );
296     Path bestPath( maxPathLength, std::pair<int,int>(-1,-1) );
297     // the best Path's score
298     double bestPathScore = FLT_MAX;
299     unsigned int bestPathLength = 0;
300
301     //================================================
302     // for storing the best 20 paths
303     const unsigned int MAX_KEPT = 20;
304     // Just copies in a blank path, MAX_KEPT times.
305     std::vector<Path> pathBuffer(MAX_KEPT, Path(maxPathLength, std
          ::pair<int,int>(-1,-1)));
306     std::vector<double> scoreBuffer(MAX_KEPT, FLT_MAX);
307     std::vector<unsigned int> lenBuffer(MAX_KEPT,0);
308     unsigned int bufferBest = 0;
309
310     //================================================
311
312     // a holds the number of residues compared for i windows of
          winSize
313     // residues, plus the comparison of the i+1'st window.
314     //
315     std::vector<int> winCache(maxPathLength,-1);
316     for ( int i = 0; i < maxPathLength; i++ ) {
317       winCache[i] = ((i+1)*i*winSize/2 + (i+1)*winSum);
318     }
319
320     //
321     // Keeps a matrix of all gapped scores
```

```
322     //
323     std::vector< std::vector<double> > allScoreBuffer;
324     for ( int indexA = 0; indexA < maxPathLength; indexA++ ) {
325       allScoreBuffer.push_back(std::vector<double>(gapMax*2+1,FLT_
            MAX));
326     }
327
328     std::vector<int> traceIndex(maxPathLength);
329     int gapBestIndex = -1;
330
331     //
332     // Investigate all the possible paths starting at [iA][iB]...
333     //
334     for ( unsigned int iA = 0; iA < lenA; iA++ ) {
335       // these are brilliant: if this path will be by definition
336       // shorter than the best, ignore it.  Very DFBNB-ish.
337       if ( iA > lenA - winSize*(bestPathLength) )
338         break;
339
340       for ( unsigned int iB = 0; iB < lenB; iB++ ) {
341
342         // heuristic short cuts; if this AFP's average match
343         // is worse than 3Ang, ignore it.
344         if ( S[iA][iB] >= D0 )
345           continue;
346
347         if ( S[iA][iB] == -1.0 )
348           continue;
349
350         // these are brilliant: if this path will be by definition
351         // shorter than the best, ignore it.  DFBNB.
352         if ( iB > lenB - winSize*(bestPathLength) )
353           break;
354
355         // current/best path seen in the while loop
356         Path curPath = Path( maxPathLength, std::pair<int,int
              >(-1,-1));
357         curPath[0].first = iA;
358         curPath[0].second = iB;
359         unsigned int curPathLength = 1;
360         traceIndex[0] = 0;
361
362         // total score; eq 11
363         double curTotalScore = FLT_MAX;
364
365         // stop when we have our longest possible path based from
```

```
366           // [iA][iJ]
367           bool done = false;
368
369           //
370           // Build all possible paths satisfying the requirements.
                 Keep
371           // the longest & best path.
372           //
373           while ( ! done ) {
374             double gapBestScore = FLT_MAX;
375             gapBestIndex = -1;
376             //
377             // Search three things: (i) in place, that is
378             // [iA+winSize-1][iB+winSize-1]; (ii) gap right
379             // [iA+g][iB] and (iii) gap down, [iA][iB+g].  Keep the
380             // best path found and add the pair to the end of the
381             // curBest path.
382             for ( unsigned int g = 0; g < (gapMax*2)+1; g++ ) {
383
384               unsigned int jA = curPath[curPathLength-1].first +
                     winSize;
385               unsigned int jB = curPath[curPathLength-1].second +
                     winSize;
386
387               if ( (g+1) % 2 == 0 ) {
388                 jA += (g+1)/2;
389               }
390               else { // ( g odd )
391                 jB += (g+1)/2;
392               }
393
394               // heuristic checking
395               if ( jA > lenA-winSize-1 || jB > lenB-winSize-1 ){
396                 continue;
397               }
398
399               // heuristic to control quality
400               if ( S[jA][jB] > D0 )
401                 continue;
402               // There are rows/cols of -1's around the similarity
403               // matrix -- if we're into those, we've gone to the
404               // end of the matrix -- no more gapping possible.
405               if ( S[jA][jB] == -1.0 )
406                 continue;
407
```

```
408             // Here is some sample output on two small proteins
                   for
409             // following code:
410             //
411             /* 3--11 vs. 3--11          (first pairs)
412              * 10--18 vs. 10--18        (end pairs)
413              * 4--17 vs 4--17     start+1 vs. end-1
414              * 5--16 vs 5--16     start+2 vs. end-2
415              * 6--15 vs 6--15     start+3 vs. end-3
416              * 7--14 vs 7--14     start+4 vs. end-4
417              * 8--13 vs 8--13     start+5 vs. end-5
418              * 9--12 vs 9--12     start+6 vs. end-6
419              */
420
421             //
422             // CE METHOD
423             //
424             double curScore = 0.0;
425
426             // for normalizing
427             double localMaxA = FLT_MIN;
428             double localMaxB = FLT_MIN;
429             for ( unsigned int s = 0; s < curPathLength; s++ ) {
430               localMaxA = (dA[curPath[s].first][jA] > localMaxA) ?
                      dA[curPath[s].first][jA] : localMaxA;
431               localMaxB = (dB[curPath[s].second][jB] > localMaxB )
                      ? dB[curPath[s].second][jB] : localMaxB;
432
433               localMaxA = (dA[curPath[s].first  + (winSize-1)][jA
                     +(winSize-1)] > localMaxA) ? dA[curPath[s].first
                       + (winSize-1)][jA+(winSize-1)] : localMaxA;
434               localMaxB = (dB[curPath[s].second + (winSize-1)][jB
                     +(winSize-1)] > localMaxB) ? dB[curPath[s].
                     second + (winSize-1)][jB+(winSize-1)]: localMaxB
                     ;
435
436               for ( unsigned int k = 1; k < winSize-1; k++ ) {
437                 localMaxA = (dA[curPath[s].first  + k][ jA + (
                        winSize-1) - k ] > localMaxA ) ? dA[curPath[s
                        ].first  + k][ jA + (winSize-1) - k ] :
                        localMaxA;
438                 localMaxB = (dB[curPath[s].second + k][ jB + (
                        winSize-1) - k ] > localMaxB ) ? dB[curPath[s
                        ].second + k][ jB + (winSize-1) - k ] :
                        localMaxB;
439               }
```

```
440             }
441
442             for ( unsigned int s = 0; s < curPathLength; s++ ) {
443               curScore += fabs( (dA[curPath[s].first][jA]/
                     localMaxA) - (dB[curPath[s].second][jB]/
                     localMaxB));
444               curScore += fabs( (dA[curPath[s].first  + (winSize
                     -1)][jA+(winSize-1)]/localMaxA) -
445                         (dB[curPath[s].second + (winSize-1)][jB
                           +(winSize-1)]/localMaxB) );
446
447               for ( unsigned int k = 1; k < winSize-1; k++ )
448                 curScore += fabs( (dA[curPath[s].first  + k][ jA +
                       (winSize-1) - k ]/localMaxA) -
449                         (dB[curPath[s].second + k][ jB + (
                           winSize-1) - k ]/localMaxB) );
450             }
451
452             curScore /= (double) winSize * (double) curPathLength;
453             //
454             // CE METHOD
455             //
456
457             // if this gapped path is over the limit, try another.
458             if ( curScore >= D1 ) {
459               continue;
460             }
461
462             if ( curScore < gapBestScore ) {
463                 curPath[curPathLength].first = jA;
464               curPath[curPathLength].second = jB;
465               gapBestScore = curScore;
466               gapBestIndex = g;
467               allScoreBuffer[curPathLength-1][g] = curScore;
468             }
469           } /// ROF -- END GAP SEARCHING
470           // CHECK ENTIRE PATH SCORE AGAINST D1.
471           int jGap, gA, gB;
472           double score1=0.0;
473
474           if ( gapBestIndex != -1 ) {
475             jGap = (gapBestIndex + 1 ) / 2;
476             if ((gapBestIndex + 1 ) % 2 == 0) {
477               gA = curPath[ curPathLength-1 ].first + winSize +
                     jGap;
478               gB = curPath[ curPathLength-1 ].second + winSize;
```

```
479               }
480               else {
481                 gA = curPath[ curPathLength-1 ].first + winSize;
482                 gB = curPath[ curPathLength-1 ].second + winSize +
                        jGap;
483               }
484
485               // perfect
486               score1 = (allScoreBuffer[curPathLength-1][gapBestIndex
                    ] * winSize * curPathLength
487                 + S[gA][gB]*winSum)/(winSize*curPathLength+winSum);
488
489               // perfect
490               curTotalScore = ((curPathLength>1 ? (allScoreBuffer[
                    curPathLength-2][traceIndex[curPathLength-1]]):(S[
                    iA][iB]))
491                 * winCache[curPathLength-1] + score1*(winCache[
                      curPathLength] - winCache[curPathLength-1]))
492                 / winCache[curPathLength];
493
494               // heuristic -- path is getting sloppy, stop looking
495               if ( curTotalScore > D1 ) {
496                 done = true;
497                 gapBestIndex=-1;
498                 break;
499               }
500               else {
501                 allScoreBuffer[curPathLength-1][gapBestIndex] =
                      curTotalScore;
502                 traceIndex[curPathLength] = gapBestIndex;
503                 curPathLength++;
504               }
505             }
506             else {
507               // if here, then there was no good gapped path
508               done = true;
509               curPathLength--;
510               break;
511             }
512
513             // if our currently best gapped path from iA and iB is
                  LONGER
514             // than the current best; or, it's equal length and the
                  score's
515             // better, keep the new path.
516             if ( curPathLength > bestPathLength ||
```

```
517             (curPathLength == bestPathLength && curTotalScore <
                   bestPathScore )) {
518             bestPathLength = curPathLength;
519             bestPathScore = curTotalScore;
520             bestPath = curPath;
521           }
522         } /// END WHILE
523
524         // record this path in the path buffer
525         if ( curPathLength > lenBuffer[bufferBest] ||
526           ( curPathLength == lenBuffer[bufferBest] &&
527             curTotalScore < scoreBuffer[bufferBest] )) {
528           bufferBest = ( bufferBest == MAX_KEPT-1 ) ? 1 :
                 bufferBest+1;
529           Path pathCopy = Path(bestPath);
530           pathBuffer[bufferBest-1] = pathCopy;
531           scoreBuffer[bufferBest-1] = curTotalScore;
532           lenBuffer[bufferBest-1] = curPathLength;
533         }
534       } /// ROF -- end for iB
535     } /// ROF -- end for iA
536
537     std::cout << std::endl;
538     std::ostringstream ss1, a, b;
539
540     // store the best paths in CE's internal data structure.
541     for ( unsigned int pC = 0; pC < MAX_KEPT; pC++ )
542       if ( lenBuffer[pC] != 0 )
543         paths.push_back( Path(pathBuffer[pC]) );
544   }
545
546
547   //============================================
548   // finds the best of all the paths
549   //============================================
550   void CE::findOptimalPath() {
551     /* Now, this->paths has all the good-scoring paths in it.  I
          must now
552      * search through all these paths and find the one with the
          lowest RMSD
553      * and do some gap wiggling.
554      */
555     if ( paths.size() == 0 )
556     {
557       //FIXME:
558       std::cout << "\nBEST ALIGNMENT\n";
```

```
559         std::cout << "==============\n";
560         std::cout << "Number aligned: 0 -- NO GOOD ALIGNMENT!" <<
                std::endl;
561         std::cout << "RMSD: 10" << std::endl;
562         std::cout << "CESCORE: 10" << std::endl;
563         std::cout << "optAlign MOL1 and i. " << ", MOL2 and i. " <<
                std::endl;
564         exit(EXIT_SUCCESS);
565       }
566       assert( paths.size() > 0 );
567       // if they are, you were naughty and created a CE without
568       // coordinates; maybe you skipped directly to a distance
             matrix?!
569       assert( coordsA.size() != 0 );
570       assert( coordsB.size() != 0 );
571
572       unsigned int bestIndex = 0;
573       double bestScore = FLT_MAX;
574       double bestCEScore = FLT_MAX;
575       double curScore = FLT_MAX;
576       TA2<double> bestRot;
577
578       for ( unsigned int i = 0; i < paths.size(); i++ ) {
579         // ignore if something wrong.
580         if ( paths[i].size() == 0 )
581           continue;
582
583           // get the true size of the path
584           unsigned int maxPathSize = 0;
585         while ( maxPathSize < paths[i].size() ) {
586           if ( paths[i][(maxPathSize++)+1].first == -1 )
587             break;
588         }
589
590         // do a quick align using the QKabsch
591         std::pair<Coords, Coords> c = this->getAlignedCoords(i);
592         TA1<double> w( c.first.size(), 1.0 );
593         QKabsch::QKabsch QK(c.first, c.second);
594         QK.align();
595
596         //
597         // Jia Y.,  Dewey G.T., Shindyalov I.N., Bourne P.E.
598         // A new scoring function and associated statistical
599         // significance for structure alignment by CE. J. Comp.
600         // Biol., 2004, 11, 787-799.
601         double internalGaps = 0.0;
```

```
602         for ( unsigned int g = 0; g < maxPathSize-1; g++ ) {
603           internalGaps += (double) ( paths[i][g].first + (int)
                winSize == paths[i][g+1].first )
604             ? 0.0 : (paths[i][g+1].first - paths[i][g].first - (int)
                  winSize);
605           internalGaps += (double) ( paths[i][g].second + (int)
                winSize == paths[i][g+1].second )
606             ? 0.0 : (paths[i][g+1].second - paths[i][g].second - (
                  int) winSize);
607         }
608         double aliLen = (double) c.first.size();
609         double numGap = internalGaps;
610
611         curScore = (QK.getRMSD()/aliLen)*(1.0+(numGap/aliLen));
612
613         // find best path based upon RMSD
614         if ( curScore < bestScore ) {
615           bestIndex = i;
616           bestCEScore = curScore;
617           bestScore = QK.getRMSD();
618           bestRot = QK.getU();
619         }
620
621
622
623
624 /*        // find best path based upon RMSD
625         if ( QK.getRMSD() < bestScore ) {
626           bestScore = QK.getRMSD();
627           bestIndex = i;
628           double internalGaps = 0.0;
629           for ( unsigned int g = 0; g < maxPathSize-1; g++ ) {
630             internalGaps += (double) ( paths[i][g].first + (int)
                  winSize == paths[i][g+1].first )
631               ? 0.0 : (paths[i][g+1].first - paths[i][g].first - (
                    int) winSize);
632             internalGaps += (double) ( paths[i][g].second + (int)
                  winSize == paths[i][g+1].second )
633               ? 0.0 : (paths[i][g+1].second - paths[i][g].second - (
                    int) winSize);
634           }
635           double aliLen = (double) c.first.size();
636           double numGap = internalGaps;
637
638           // here I use the CE score as reported in:
639           //
```

```
640         // Jia Y.,  Dewey G.T., Shindyalov I.N., Bourne P.E.
641         // A new scoring function and associated statistical
642         // significance for structure alignment by CE. J. Comp.
643         // Biol., 2004, 11, 787-799.
644         //
645         bestCEScore = (bestScore/aliLen)*(1.0+(numGap/aliLen));
646         //std::cout << "aliLen: " << aliLen << "; numGap: " <<
                numGap << std::endl;
647       }
648 */
649
650
651     }
652
653     //
654     // FIXME:
655     //
656     // INSERT CODE HERE TO MAKE THE GAP-SHIFTED PATHS
657     //
658     // """
659     // A final optimization has been added which contributes to
660     // up to 2A improvement in the RMSD between two protein
          structures.
661     // Each gap in this single alignment is evaluated for possible
          re-
662     // location in both directions up to m/2 positions, where m is
          the
663     // AFP size and if the RMSD of superimposed structures
          indicates
664     // improvement, the modified gap positions are adopted.
665     // """
666     //
667
668     // Pseudo-code for optimization
669     // ============================
670
671     //
672     // Report on the best alignment found
673     //
674     std::ostringstream e, d, f;
675
676     unsigned int maxPathSize = 0;
677     while ( maxPathSize < paths[bestIndex].size() )
678       if ( paths[bestIndex][(maxPathSize++)+1].first == -1 )
679         break;
680
```

```
681     for ( unsigned int q = 0; q < maxPathSize; q++ ) {
682       if ( q != maxPathSize-1 ) {
683         e << paths[bestIndex][q].first << "-" << paths[bestIndex][
                q].first+(winSize-1) << "+";
684         d << paths[bestIndex][q].second << "-" << paths[bestIndex
                ][q].second+(winSize-1) << "+";
685       }
686       else {
687         e << paths[bestIndex][q].first << "-" << paths[bestIndex][
                q].first+(winSize-1);
688         d << paths[bestIndex][q].second << "-" << paths[bestIndex
                ][q].second+(winSize-1);
689       }
690
691       for ( unsigned int indiv = 0; indiv < winSize; indiv++ )
692         f << paths[bestIndex][q].first+indiv << "\t" << paths[
                bestIndex][q].second+indiv << "\n";
693     }
694
695     // for analysis, I sometimes just want the chosen paths --
           nothing else.  If so,
696     // just print out the paths or full data, here.
697     if ( args["j"] == "false" ) {
698       std::cout << "\nBEST ALIGNMENT\n";
699       std::cout << "==============\n";
700       std::cout << "Number aligned: " << maxPathSize*winSize <<
              std::endl;
701       std::cout << "RMSD: " << bestScore << std::endl;
702       std::cout << "CESCORE: " << bestCEScore << std::endl;
703       std::cout << "optAlign MOL1 and i. " << e.str() << ", MOL2
              and i. " << d.str() << std::endl;
704       std::cout << "bestRotation: " << bestRot << std::endl;
705     } else {
706       std::cout << f.str() << std::endl;
707     }
708   }
709 }
```

A.2 Source Code for Taking the SVD of a Similarity Matrix

This code replaces the matrix `S` calculated at line 265 in the above code. This code may be inserted after line 265 to test the SVD of a CE matrix.

Source Code A.2: C++ Source code for Calculating the SVD of a Similarity Matrix

```
1  //================================
2  // Test the SVD of the CE matrix.
3  // ================================
4
5  // left singular vectors
6  TA2<double> W = TA2<double>(lenA, lenA);
7  TA2<double> Ws = *(new TA2<double>( lenA, lenB ));
8
9  // right singular vectors
10 TA2<double> Vt = TA2<double>(lenB, lenB);
11 JAMA::SVD<double> svd = JAMA::SVD<double>( S );
12 svd.getU(W);
13 svd.getV(Vt);
14
15 // need to make this like R does.
16 for ( unsigned int i = 0; i < lenA; i++ )
17   for ( unsigned int j = 0; j < lenB; j++ )
18     Ws[i][j] = W[i][j];
19
20 Vt = QKabsch::transpose(Vt);
21 S = (*new SM(TNT::matmult(Ws, Vt)));
22
23 // find the min of S and add abs(min) to this -- make this a
     positive matrix (for CE align)
24 double sMin = FLT_MAX;
25 for ( int i = 0; i < S.dim1(); i++ )
26   for ( int j = 0; j < S.dim2(); j++ )
27     if ( sMin > S[i][j] ) { sMin = S[i][j]; }
28
29 for ( int i = 0; i < S.dim1(); i++ )
30   for ( int j = 0; j < S.dim2(); j++ )
31     S[i][j] += fabs(sMin);
32
```

```
33 //std::cout << "Dims: " << S.dim1() << " and " << S.dim2() << std
       ::endl;
34
35 //
36 // END
37 //
38 ////////////////////////////////////////////////
```

Vita

Jason Vertrees was born in Peoria, Illinois on March 6th, 1977, to the proud parents Roger and Susan Vertrees (and soon to be included step-mother Trudy Vertrees). Jason attended the University of Texas at Austin, where he received a Bachelor of Arts in both Computer Science and Japanese. For a few years Jason also worked for a small technology firm in Austin, Texas. Unfortunately that company became a casualty of economic downturn. After his undergraduate graduation, Jason then worked in the Optical Microscopy laboratory at UTMB to gain laboratory experience before joining the Biophysical, Structural and Computational Biology Educational Track at the University of Texas Medical Branch, in Galveston.

Permanent Address: 331 Poverty Lane, Lebanon, NH 03766

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, Ayman El-Khashab, and Jason Vertrees.