UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE FÍSICA

EFFECTS OF CONFINEMENT IN THE FOLDING OF KNOTTED PROTEINS

João Nuno Correia Especial

MESTRADO EM FÍSICA
Especialização em Física da Matéria Condensada e Nano-Materiais

Dissertação orientada por:
Prof.ª Doutora Patrícia Ferreira Neves Faísca

2018

Acknowledgements

A part of this work was performed on the computational resource of INCD (http://www.incd.pt) funded by FCT and UE under project LISBOA-01-0145-FEDER-022153.

Resumo

A compreensão detalhada do processo de enrolamento e dobragem de proteínas enodadas permanece um problema em aberto.

É consensual que o complexo de chaperoninas GroEL-GroES promove a dobragem correcta de proteínas *in vivo*. Tem sido defendido que este efeito se deve a que o complexo providencia isolamento do ambiente congestionado do citoplasma, protegendo, desta forma, a proteína de processos de *misfolding* e agregação. No entanto, experiências *in vitro* mostram que no caso das proteínas YibK e YbeA, ambas contendo um nó de trevo (3_1) na sua estrutura nativa, se observa uma aceleração considerável (por um factor de 20) da taxa do processo de dobragem, quando este ocorre no ambiente confinado da chaperonina.

O mecanismo de acção da chaperonina sobre o processo de dobragem e enodamento permanece desconhecido.

O presente trabalho determinou as consequências do confinamentos estérico e hidrofóbico, que ocorrem durante o ciclo de funcionamento da chaperonina, no contexto de modelos de rede e simulações pelo método de Monte Carlo. Os resultados indicam que quando, para além de confinamento estérico, se passa a considerar interações hidrofóbicas entre a proteína enodada e a superfície interna da chaperonina, se verifica que a temperatura de transição de *folding* diminui de forma sistemática à medida que aumenta a intensidade da interacção hidrofóbica. Este resultado sugere que uma das funções da primeira parte do ciclo de funcionamento da chaperonina é diminuir a estabilidade termodinâmica de eventuais estados iniciais mal formados, assegurando que, com elevada probabilidade, a proteína entra na segunda parte do ciclo numa conformação desdobrada. À medida que o ciclo progride e a interação hidrofóbica com a superfície interna da chaperonina enfraquece, a temperatura de transição de *folding* aumenta progressivamente, o que conjugado com a protecção contra processos de *misfolding* e agregação, assegura que a proteína tenha alta probabilidade (aproximadamente 40%) de se encontrar na sua conformação nativa quando o ciclo da chaperonina termina.

Demonstra-se ainda no presente trabalho que este nível de eficácia pode explicar as rápidas taxas do processo de *folding* observadas.

Palavras-chave: Dobragem de proteínas, Proteínas enodadas, Monte Carlo, Física Estatística, Chaperões moleculares.

Abstract

Detailed understanding of the knotted protein folding process remains an open problem.

It is consensual that the GroEL-GroES chaperonin complex promotes correct protein folding *in vivo*. It has been argued that this effect may be due to the chaperonin complex providing isolation from the crowded cytoplasm environment, thus protecting the protein from misfolding processes and aggregation. However, *in vitro* experiments have also shown that in the case of proteins YibK and YbeA, both of which contain a trefoil knot (3_1) in their native structure, a considerable acceleration (20-fold) is observed in the folding process when it takes place in the confined environment of this chaperonin complex.

The mechanism through which the chaperonin acts upon the folding and knotting process is still unknown.

The present work determined the consequences of steric and hydrophobic confinement, that occur throughout the working cycle of the chaperonin, in the context of lattice models and Monte Carlo simulations. The results indicate that when hydrophobic interactions between the knotted protein and the internal surface of the chaperonin are considered in addition to steric confinement, the temperature of the folding transition steadily decreases as the intensity of the hydrophobic interaction increases. This finding suggests that one of the functions of the first part of the chaperonin working cycle is to diminish the thermodynamic stability of any misfolded initial states, ensuring that the protein has a high probability of entering the second part of the cycle in an unfolded conformation. As the cycle advances and hydrophobic interaction with the internal surface of the chaperonin weakens, the temperature of the folding transition steadily increases and this, together with protection from misfolding and aggregation, ensures a high probability (approximately 40%) of the protein having its native conformation at the end of the chaperonin cycle.

The present work also shows that this effectiveness level can explain the observed fast folding rates.

Keywords: Protein folding, Knotted proteins, Monte Carlo, Statistical Physics, Molecular chaperones.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Proteins and protein folding

Proteins are the essence of life, playing crucial roles in virtually every biological process. Amongst many other functions, proteins drive and control our metabolism, protect us against viruses and bacteria, and allow us to breathe, to move and to see. In order to work properly, these extraordinary nanorobots, formed by many thousands of atoms, must acquire a specific biologically functional structure through the process of protein folding.

Due to this pivotal role of proteins it is not surprising that many pathological conditions, at organ and organism scale, have been shown to be direct consequences of protein misfolding and aggregation (e.g. Alzheimer's disease, cataracts and many others) and consequently the design of root-cause preventive and therapeutic measures for these conditions requires a thorough understanding of their complex physical processes.

Physically, protein molecules are heteropolymers i.e. long linear chains made from approximately 50 to 3000 monomers, in which each monomer is one of the 20 naturally occurring amino acids (see Table 1.1), these being connected sequentially via peptide bonds. The protein molecule's backbone is thus a polypeptide chain.

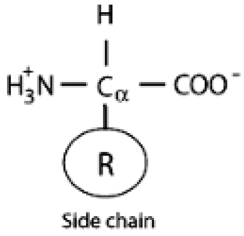

Figure 1.1: Generic structure of an amino acid molecule. A specific side chain molecular structure characterizes each particular amino acid. Reproduced from [1].

In vivo (and now quite often also *in vitro*), protein molecules are synthesized in macro-molecular structures called ribosomes through a process known as translation. As it is synthesized, the protein molecule is released into the cytoplasm in its linear conformation and, from it, spontaneously self-assembles, through the folding process, into its functional conformation.

According to the properties of the functional conformation under physiological conditions, three

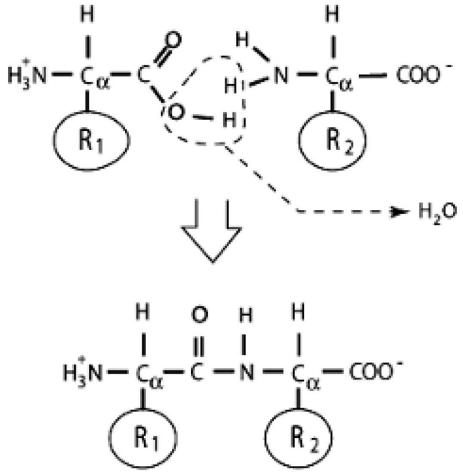

Figure 1.2: Peptide bond formation between two amino acid molecules. Reproduced from [1].

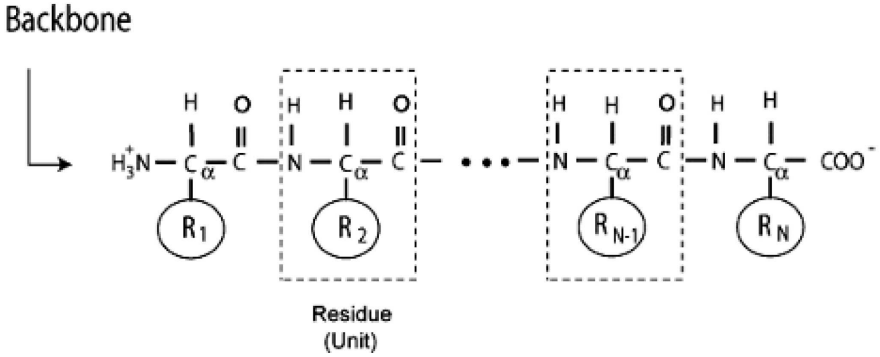

Figure 1.3: Generic structure of a protein molecule. Reproduced from [1].

classes of proteins have been recognized: globular proteins (GPs) fold to a unique three-dimensional conformation, known as its native state; intrinsically unstructured proteins (IUPs) rather than fold to a unique conformation, in their entirety or in part, populate a conformational ensemble; finally, metamorphic proteins can assume several conformations of approximately equal energy, each having a different biochemical function. The present work discusses only globular proteins.

Depending on the protein, this spontaneous folding process may take from less than a second to about 15 minutes.

The dominant driving force in protein folding is the hydrophobic effect [1, 5]. The water molecules in the cytoplasmatic medium seek to form hydrogen bonds with each other or with other polar molecular regions and the presence of nonpolar molecular regions in the medium obstructs such bond formation. The net effect is that nonpolar molecular regions are pushed against each other as the network of water molecules attempts to minimize the total contact surface with these regions. These regions thus appear to behave as if they are trying to avoid contact with the aqueous medium and hence the name ”hydrophobic” for this effect.

The study of the protein folding process can be traced back to the 1930s, when Anson and Mirsky [6] first reported that denaturation was a reversible process accompanied by a sudden increase in the viscosity of the concentrated protein solution and proposed a two-state (i.e. single-exponential) kinetic model to

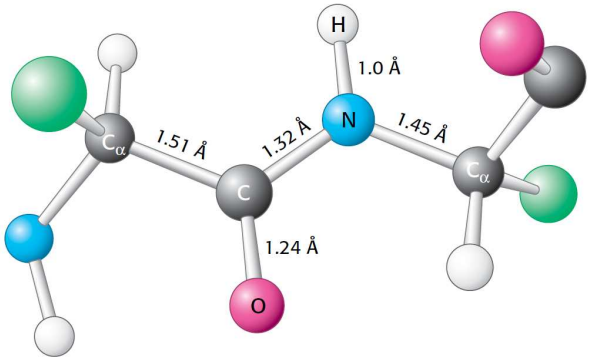

Figure 1.4: Tridimensional structure of the peptide bond. Reproduced from [2]

describe it. Nevertheless, it was only in the 1960s that Christian Anfinsen performed a series of *in vitro* experiments with bovine pancreatic enzyme ribonuclease A, that led to the now called thermodynamic hypothesis [7, 8]. Anfinsen was able to show that chemically denatured proteins are able to regain their native conformation spontaneously, without requiring any added chemical. In thermodynamic terms this implies that the native state is the global minimum of the free energy of the protein/solvent system. Anfinsen's experiments thus show that all the information required to fold the protein to its native state is embedded in the protein's primary structure (i.e. in its amino acid sequence). For these experiments Anfinsen was awarded the Nobel Prize in Chemistry in 1972.

Shortly after Anfinsen's work, Cyrus Levinthal (a once nuclear physicist that had moved to molecular biology) claimed that a random search of the conformational space (as implied by the two-state character of the folding transition) should be expected to take longer than the age of the Universe to find the protein's native conformation, even for small (< 100 amino acids long) proteins [9, 10]. Because proteins are in fact able to find their native state in, at most, a few minutes, as previously mentioned, this argument is known in the folding literature as Levinthal's paradox.

This issue has been challenging researchers for decades. Levinthal himself proposed that instead of being under thermodynamic control, folding must be under kinetic control. This means that folding protein molecules, instead of randomly exploring the entire conformational space, follow a mostly descendent path along the free energy hyper-surface of the protein/solvent system [9]. According to Levinthal, folding should thus proceed along pathways, formed by sequences of intermediate states (i.e. partially folded conformations) with increasing amounts of native structure, en-route to the native state. The idea of a folding pathway that joins the linear conformation to the native state is quite attractive since it strongly restricts the volume of the relevant conformational space explored and thus reconciles the stochastic nature of the process with the folding timescale.

In the present work we will be addressing not the general protein folding problem but a particular sub-problem: the folding of knotted proteins.

1.2 Knotted proteins

Knotted proteins are the particular class of proteins whose native state is arranged in the form of a knot [11]. As seen in the previous section, a protein's backbone is an open curve which, thus, has two *termini*. Since topological knots are closed curves in space, in a strict mathematical sense, knots in proteins

Table 1.1: The 20 naturally occurring amino acids.

Amino acid	Three-letter abbreviation	One-letter abbreviation	Hydropathy index [4]
Alanine	Ala	A	1.8
Arginine	Arg	R	-4.5
Asparagine	Asn	N	-3.5
Aspartic acid	Asp	D	-3.5
Cysteine	Cys	C	2.5
Glutamine	Gln	Q	-3.5
Glutamic acid	Glu	E	-3.5
Glycine	Gly	G	-0.4
Histidine	His	H	-3.2
Isoleucine	Ile	I	4.5
Leucine	Leu	L	3.8
Lysine	Lys	K	-3.9
Methionine	Met	M	1.9
Phenylalanine	Phe	F	2.8
Proline	Pro	P	-1.6
Serine	Ser	S	-0.8
Threonine	Thr	T	-0.7
Tryptophan	Trp	W	-0.9
Tyrosine	Tyr	Y	-1.3
Valine	Val	V	4.2

are not actually topological knots and it is, therefore, more accurate to say that protein conformations embed physical (or open) knots. However, since the vast majority of knotted proteins have their termini located close to the protein's surface, they can be unambiguously connected by a curve external to the protein's surface to form a closed loop, hence becoming a strict sense topological knot. Even when one or both of the termini is/are located deeper into the protein structure, shrinking the protein backbone while keeping the termini fixed will preserve the topology and always eventually lead to the termini emerging from the protein's surface, becoming unambiguously connectable and hence enabling unambiguous determination of the knotted nature of any protein's backbone. The knottiness is, consequently, always a well defined property of any protein.

The simplest knot is the circle, which in this context is known as the *unknot*, and the simplest non-trivial knots are the trefoil, the figure-eight, and the three-twist knot. The latter differ in their crossing-number (3, 4 and 5, respectively), which is a knot invariant defined as the minimal number of crossings that a planar projection of the knot can have.

The first knotted protein was reported in 1977 [12] but it was only in 2000 that these intricate molecules came into the spotlight, following the development of computational methods to detect knotted topologies in proteins [13, 14], which we will be describing in detail later.

Presently it is known that about 1% of the available Protein Data Bank (PDB) entries correspond to knotted proteins [15]. The trefoil is by far the most common knot type in the PDB, but it is possible to

find a few proteins with more complex knots, including the Stevedore's knot, with crossing-number 6 [16]. The knotted core is the minimum segment of the protein's backbone that contains the knot and the knot is called deep or shallow according to whether the number of peptide bonds that lie between the knotted core and the termini is large or small.

If determining how 'regular' proteins fold is already a challenging problem, doing so for knotted proteins is an even more formidable one, and so this became, during the last decade, a much studied subject.

Computer simulations based on a wide array of models and sampling strategies, ranging from Monte Carlo simulations of lattice models [17, 18] to Molecular Dynamics simulations of realistic force fields [19, 20], have been playing a decisive role in this endeavour. According to the current picture, the folding mechanism of trefoil proteins is a highly ordered process, where the formation of the so-called knotting loop precedes the threading step upon which the protein gets knotted. Two scenarios have been put forward for the threading step based on molecular simulations. One proposes that the chain terminus that lies closer to the knotted core threads the knotting loop directly. The other proposes that the chain terminus arranges itself into a hairpin that threads the knotting loop while transiently forming a slipknotted conformation (reviewed in [21]).

Only five articles exist in the literature addressing the folding and knotting mechanisms of tangled proteins with more complex topologies: A molecular dynamics simulation study of the 6_1 knotted DehI protein [16], a theoretical study from our group, that looked into the folding mechanism of a lattice protein embedding a three-twist knot [18], two *in vitro* studies [22, 23] (one that explored the folding of three-twist knotted protein UCH-L3 [22] and the other that used single molecule experiments to mechanically unfold the protein ubiquitin C-terminal hydrolase isoenzyme L1 [23]) and, finally, a recent molecular dynamics simulation study of the effects of steric confinement on three 5_2 knotted proteins of the UCH family, 3IRT, 2LEN and 4I6N [24].

Both *in vitro* studies [22, 23] highlighted the complex nature of the folding pathway, with intermediate states, and the second study [23] provided direct evidence that a threading event associated with formation of a knot, significantly slows down the folding of UCH-L1, in line with lattice predictions.

The structural complexity associated with the knotting process typically leads to slow folding rates for knotted proteins, both in molecular simulations and in experiments *in vitro* (reviewed in [21]). This is due in part to the need to break and re-establish specific native contacts, something which must occur whenever folding has followed an incorrect sequence of events that has lead to malformed knots and other topologically trapped conformations [25]. However, experiments *in vitro* [26, 27] have shown that knotted trefoil proteins YibK and YbeA can efficiently self-tie without populating misfolded species and, even more interestingly, that their folding rates are substantially enhanced (approximately 20-fold) when folding occurs in the presence of the GroEL-GroES chaperonin system. It is exactly the origin of this folding rate enhancement that the present work aims to elucidate.

1.3 Protein folding *in vivo*

So far we have been mostly discussing protein folding *in vitro* or *in silico*. In living systems, protein folding occurs within the cell and this represents a significant deviation from the ideal (i.e. highly diluted and 'clean') environment of the test tube.

Indeed, if on one hand the cell cytoplasm is a highly crowded environment (with macromolecular

concentrations of up to 300-400 mg/ml) [28], which, due to excluded volume interactions, significantly increases the probability of protein misfolding and aggregation, on the other, the cell has a series of control mechanisms that provide fault-tolerance to the protein folding process. One such error correcting mechanism is based on the so-called molecular chaperones [29] of which the GroEL-GroES bacterial chaperonin system is a paradigmatic example [3].

By solving the structure of GroEL-GroES, researchers have found that it contains two cylindrical chambers with a variable diameter of 8.0 - 9.5 nm that, each, can accommodate a protein molecule of up to 60 KDa (i.e. with a chain length of up to 550 amino acids) [30].

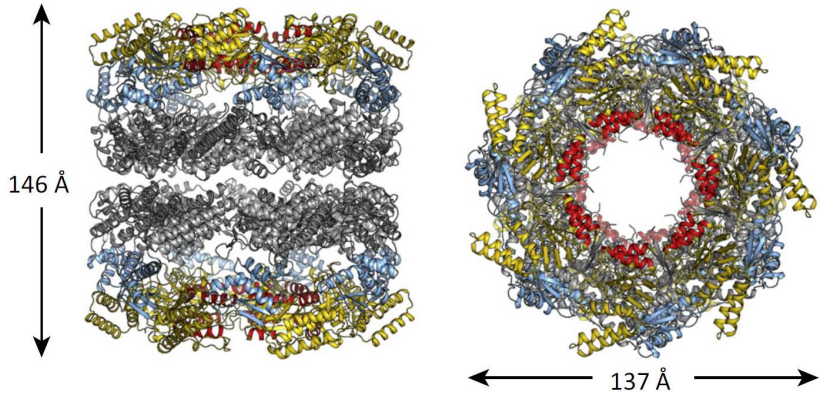

Figure 1.5: The crystal structure of GroEL. Reproduced from [3]

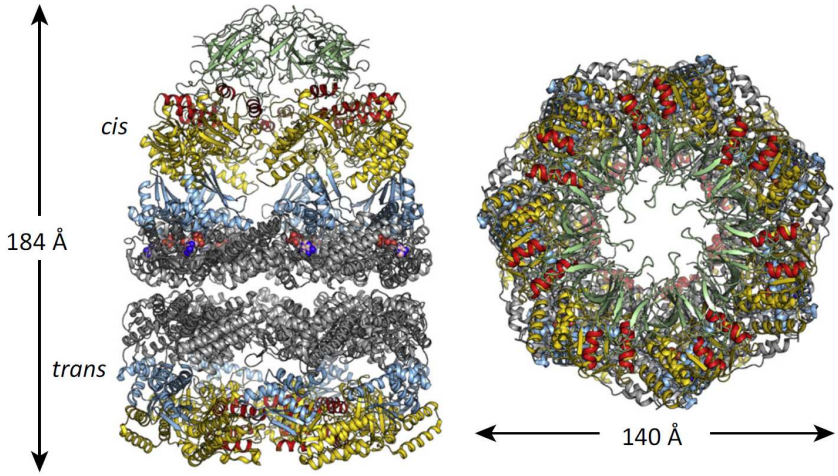

Figure 1.6: The crystal structure of the GroEL - GroES complex. Reproduced from [3]

The GroEL-GroES chaperonin is an ATP-driven molecular machine whose function is to assist the folding of unfolded proteins or fix the structure of misfolded proteins and prevent aggregation by sequestering the folding/damaged protein in the confined environment of one of its chambers, where (re)folding to the correct native state is allowed to take place in a series of ATP-driven cycles at infinite dilution.

In the state in which no ATP or GroES are bound to GroEL, the unfolded/misfolded protein is inserted into one of its chambers and interacts with the hydrophobic residues that, in this state, line the interior of the GroEL cavity. Extremely large unfolded proteins can even extend out of GroEL at this stage, since,

when GroES is not bound to it, its chambers are open at the ends of the cylinder. Upon ATP and GroES binding, the volume of the chamber more than doubles, and a physical change occurs in the chamber's inner walls, which become hydrophilic and acquire a net negative electric charge of -42 (189 negatively and 147 positively charged amino acid residues). The protein remains inside the GroEL-GroES cage for ≈6 seconds, which is the time necessary for 7 ATP molecules to hydrolize and be released.

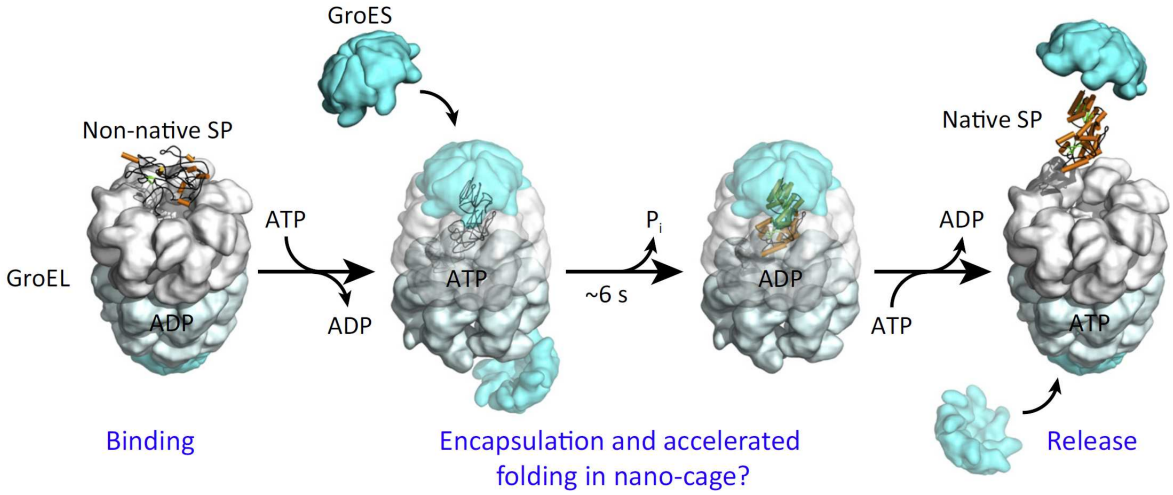

Figure 1.7: The GroEL/ES chaperonin protein folding cycle. Reproduced from [3]

The exact details of how the chaperonin cage induces (re)folding to the correct native structure has remained elusive and two models have been proposed for the mechanism according to which GroEL acts to enhance the yield of correctly folded proteins: the passive-cage [31, 32] and the iterative annealing [33, 34, 35] models. In the passive-cage (also known as the Anfinsen's cage) hypothesis, the chaperonin does not actively influence folding, and only provides a restricted environment that protects the folding molecule against aggregation in the cytosol. In the iterative annealing model, on the other hand, the chaperonin plays an active role in the folding process, via repeated denaturation of misfolded conformations, in addition to providing the protective environment so crucial for the protein molecule to achieve the native state.

The present work investigates both hypotheses through simulation of the effects of physical and hydrophobic confinement on the thermodynamics and kinetics of the folding transition of small lattice proteins whose backbones are arranged in the form of a knotted trefoil (a 3_1 knot) and a three-twist (a 5_2) knot. We frame our investigation on the use of tools from computational statistical physics. In particular, we use the Metropolis Monte Carlo [36] sampling engine combined with a replica-exchange protocol [37] that allows us to access the equilibrium distribution of the folding transition at several temperatures, and we evaluate maximum likelihood estimates of all thermodynamic properties of thermal equilibria states with the weighted histogram analysis method (WHAM) [38].

In the following chapter we develop a kinetic theory for chaperone assisted protein folding and in the next we provide a detailed description of the models employed and a careful and comprehensive explanation of the simulation and data analysis methods. Then we present and discuss the results obtained and finally take some conclusions and provide suggestions for future investigations.

Chapter 2

Kinetic theory of chaperone-assisted protein folding

In chaperone-assisted protein folding each protein molecule folds inside the cage of a chaperone.

Each chaperone continually repeats a cycle in which a protein molecule is inserted into its cage at the beginning of the cycle and ejected from its cage at the end.

Assuming that all protein molecules fold with the assistance of a chaperone, we calculate in the following sections the kinetics of the folding process.

2.1 Single protein molecule folding process

The single protein molecule folding process that takes place within the chaperone cage during one chaperone cycle can be regarded as a binary random trial, the two possible outcomes and their respective probabilities being:

Success : The protein molecule exits the cage in its native conformation.
Probability : p.

Failure : The protein molecule exits the cage in a non-native conformation.
Probability : $q = 1 - p$.

When the protein molecule exits the chaperone in a non-native conformation, the cellular machinery reintroduces it into a chaperone cage, for it to undergo a new folding process.

The chaperone cycle is composed of two stages. In the first the protein molecule becomes unfolded, thereby eliminating any trace of its initial conformation and hence of any misfolded features it might initially have had and that might have hindered folding into the native conformation. In the second, the unfolded protein molecule slowly relaxes towards its native conformation.

The consequence of making the protein molecule first undergo unfolding before refolding is that successive chaperone-assisted folding processes are statistically independent and hence, the probability that the protein molecule remains in a non-native conformation at the end of n folding processes is

$$Q_n = q^n. \tag{2.1}$$

The probability that the protein molecule becomes correctly folded in the nth cycle thus is

$$p_n = p\,Q_{n-1} = p\,q^{n-1} = p\,(1-p)^{n-1}. \tag{2.2}$$

The mean number of cycles required for the protein molecule to become correctly folded is

$$\langle n \rangle = \sum_{n=1}^{\infty} n\, p_n = p \sum_{n=1}^{\infty} n\, (1-p)^{n-1}. \tag{2.3}$$

Given that

$$\sum_{n=1}^{\infty} (1-p)^n = \frac{1-p}{1-(1-p)} = \frac{1-p}{p}, \tag{2.4}$$

its derivative in order to p is

$$\frac{d}{dp}\left(\sum_{n=1}^{\infty} (1-p)^n\right) = \frac{(-1)p - (1-p)1}{p^2} \Leftrightarrow \tag{2.5}$$

$$\Leftrightarrow \sum_{n=1}^{\infty} n\, (1-p)^{n-1}(-1) = -\frac{1}{p^2} \Leftrightarrow \tag{2.6}$$

$$\Leftrightarrow \sum_{n=1}^{\infty} n\, (1-p)^{n-1} = \frac{1}{p^2}. \tag{2.7}$$

Hence

$$\langle n \rangle = p\, \frac{1}{p^2} = \frac{1}{p}. \tag{2.8}$$

If the time that the chaperone takes to perform one cycle is τ, the mean time a protein molecule takes to correctly fold through the chaperone-assisted process is

$$\langle t_f \rangle_1 = \langle n \rangle\, \tau = \frac{\tau}{p}. \tag{2.9}$$

2.2 The folding process for N protein molecules

In a system composed of N protein molecules, if $N_f(t)$ is the number of protein molecules that at time t is in the native conformation (i.e. already correctly folded) and $N_u(t)$ the number of protein molecules that at time t is still in a non-native conformation (i.e. still 'unfolded'), then, for all t,

$$N_f(t) + N_u(t) = N \Leftrightarrow \tag{2.10}$$

$$\Leftrightarrow \frac{dN_f}{dt} + \frac{dN_u}{dt} = 0 \Leftrightarrow \tag{2.11}$$

$$\Leftrightarrow \frac{dN_u}{dt} = -\frac{dN_f}{dt}. \tag{2.12}$$

Assuming $N \ggg 1$, in a short time interval, dt, the increase in number of folded protein molecules, dN_f, should be proportional to the number of still unfolded protein molecules, $N_u(t)$, and to the duration of the interval. Let's call k_f the proportionality constant. The master equation for the process is

$$dN_f = k_f\, N_u\, dt \Leftrightarrow \tag{2.13}$$

$$\Leftrightarrow \frac{dN_f}{dt} = k_f\, N_u \Leftrightarrow \tag{2.14}$$

$$\Leftrightarrow \frac{dN_u}{dt} = -k_f\, N_u, \tag{2.15}$$

With initial condition $N_u(0) = N$

$$N_u(t) = N e^{-k_f t}, \tag{2.16}$$

the constant k_f hence being the folding rate and the fraction unfolded $f_u(t) = N_u(t)/N$ being

$$f_u(t) = e^{-k_f t}. \tag{2.17}$$

With $f_f(t) = N_f(t)/N = (N - N_u(t))/N = 1 - f_u(t)$ being the folded fraction,

$$\frac{df_f}{dt} = -\frac{df_u}{dt} = k_f e^{-k_f t}, \tag{2.18}$$

and the mean folding time for the system of N protein molecules is

$$\begin{aligned}
\langle t_f \rangle_N &= \int_0^1 t \, df_f = && (2.19)\\
&= \int_0^\infty t \, \frac{df_f}{dt} \, dt = && (2.20)\\
&= \int_0^\infty t \, k_f \, e^{-k_f t} \, dt = && (2.21)\\
&= k_f \int_0^\infty t \, e^{-k_f t} \, dt = && (2.22)\\
&= k_f \left(\left[-t \, \frac{e^{-k_f t}}{k_f} \right]_0^\infty - \int_0^\infty -\frac{e^{-k_f t}}{k_f} \, dt \right) = && (2.23)\\
&= \int_0^\infty e^{-k_f t} \, dt = && (2.24)\\
&= \left[-\frac{e^{-k_f t}}{k_f} \right]_0^\infty = && (2.25)\\
&= \frac{1}{k_f} && (2.26)
\end{aligned}$$

Assuming that the time the cellular machinery takes to insert the protein molecule into the chaperone cage is negligible in comparison to the duration of the chaperone cycle and that the total number of chaperones available is larger than the total number of protein molecules, N, the collective folding process is neither constrained by delays nor by unavailability of chaperones and hence the collective folding process is a concurrent repetition of N independent single protein molecule chaperone-assisted folding processes. Consequently, the mean folding time is the mean of N repetitions of the same random trial and the mean of the distribution of N repetitions is identical to the mean of the distribution of a single random trial

$$\langle t_f \rangle_N = \langle t_f \rangle_1 = \frac{\tau}{p}, \tag{2.27}$$

and

$$k_f = \frac{1}{\langle t_f \rangle_N} = \frac{p}{\tau}. \tag{2.28}$$

This result enables a reformulation of the problem we are addressing. In chapter 1 we stated the problem in terms of folding rates, k_f. We can now reformulate it in terms of the probability of the protein system being in its native conformation at the end of one chaperone cycle, p, i.e. the effectiveness of the chaperone cycle at producing correctly folded proteins.

For the 3_1 knotted proteins YibK and YbeA it has been measured *in vitro* [26, 27] that under assistance of GroEL-GroES chaperonins, at physiologic temperature, $k_f > 2\,\mathrm{min}^{-1}$. Given that the duration

of the cycle of the GroEL-GroES chaperonin, τ, is approximately $6\,\mathrm{s}$ [29, 3], this folding rate can be explained by an effectiveness of

$$p = k_f\,\tau > \frac{2}{60}\,6 = 0.2. \tag{2.29}$$

The calculation of probabilities in complex contexts is the natural domain of application of Monte Carlo methods and hence, this reformulation brings the problem under consideration within the scope of these methods. The simulation work presented in the following chapters thus aims to determine whether an effectiveness higher than 20% could result from confinement effects due to the chaperone cage.

Chapter 3

Models and Methods

3.1 Models

3.1.1 Protein Models

A model combines two components: the system's representation, that describes its instantaneous state, and the system's Hamiltonian, that describes its dynamics.

The protein representations used are simple lattice conformations, configured as self-avoiding walks on the simple cubic lattice (coordination number 6) in which amino acids (also referred to as residues) are reduced to beads of uniform size that occupy the lattice sites, successive amino acids along the backbone occupying near-neighbor sites and the backbone peptide bond connecting them being represented by the lattice edge that joins the two sites. To satisfy excluded volume constraints only one amino acid is allowed per lattice site (self-avoidance). This implies that backbone peptide bond length in our model is fixed, being always one lattice unit in length, this unit thus representing the average distance between successive alpha-Carbon atoms in a protein backbone which is approximately 0,38 nm (see Fig. 1.4). Yet another approximation is that only bonds in the (0,0,1) direction and its five rotations and inversions are allowed; bonds in the (0,1,1) and (1,1,1) directions and their rotations and inversions not being allowed.

Each bead thus has two backbone bond contacts (with the exception of terminal residues which have only one) and thus may have up to four non-backbone contacts (or five in the case of the termini).

Polymer lattice representations have been in use for more than 70 years [39] and have been instrumental to many fundamental developments in polymer theory [40] and knot theory [41].

In this work we consider two lattice systems representing two knotted proteins. The first, which we will henceforth call k31, has chain length $N = 41$, has 40 native contacts and was designed to embed a trefoil (or 3_1) knot in its native structure (Fig. 3.1 A). The second, which we will henceforth call k52, has chain length $N = 52$, 52 native contacts and a native structure which embeds a three-twist (or 5_2) knot (Fig. 3.1 B). The knotted core (KC), i.e. the minimal segment of the chain that contains the knot, is highlighted in blue in the three-dimensional representations of the proteins. Both knots classify as shallow since removing 3 beads from the native structure of k31 or 2 beads from that of k52 is enough to unknot the fold.

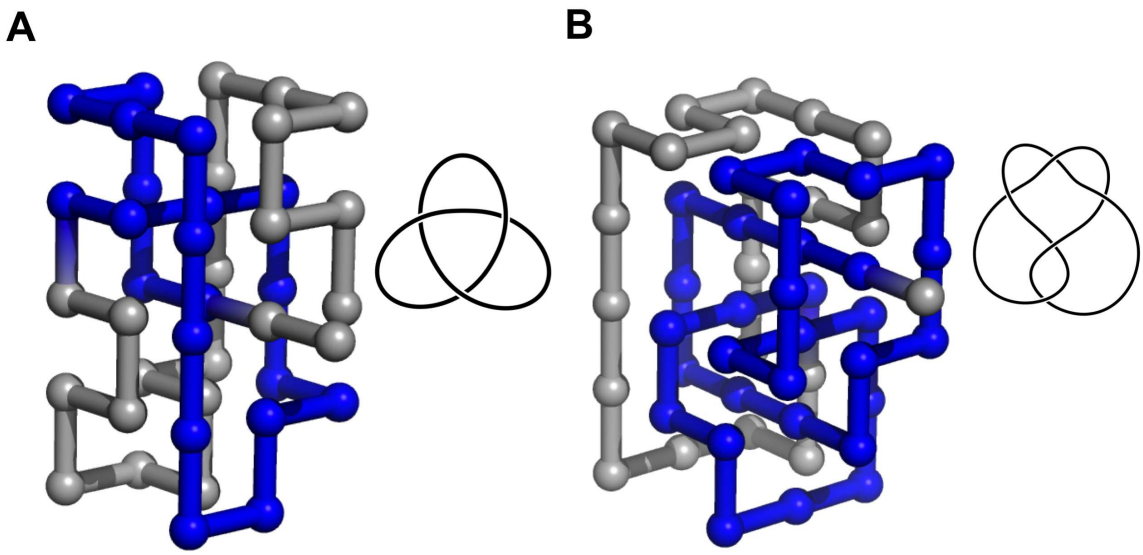

Figure 3.1: Native conformations of test proteins k31, with embedded 3_1 knot (A), and k52, with embedded 5_2 knot (B). The knotted core (KC), extending from residue 3 to 22 (k31) and from residue 21 to 51 (k52), is highlighted in blue.

Intramolecular interactions, are modeled with the native centric Gō potential [42], i.e., the total energy (Hamiltonian) of a conformation with bead coordinates $\{\vec{r_i}\}$ is given by:

$$H\left(\{\vec{r_i}\}\right) = -\epsilon \sum_{i,j>i+2}^{N} \Delta(\vec{r_i} - \vec{r_j}), \tag{3.1}$$

where N is the chain length measured in number of beads, ϵ is the uniform interaction energy parameter (equal in this study to the average energy required to break a native contact), and the contact function, Δ, is unity only if beads i and j form a native contact, i.e., a contact that is present in the native structure (see Fig. 3.2), being zero otherwise.

A model that only takes into account native interactions is called native centric. Protein lattice representations with native centric potentials were introduced by Gō in 1975 [42] and have enabled important conceptual progress in the understanding of minimal frustration in protein folding [43] and protein folding kinetics [44].

The Hamiltonian can be made adimensional through division by ϵ, becoming:

$$H^*\left(\{\vec{r_i}\}\right) \equiv \frac{H\left(\{\vec{r_i}\}\right)}{\epsilon} = -\sum_{i,j>i+2}^{N} \Delta(\vec{r_i} - \vec{r_j}). \tag{3.2}$$

The intramolecular energy value is now the symmetric of the number of native contacts present in the conformation.

3.1.2 Chaperone Models

Steric (or physical) confinement is modeled by placing the protein inside an elementary geometry. For symmetry reasons we choose a rigid cubic box, which restricts the conformation of the lattice system and its movements in three dimensions. The linear size of the box, L, is measured in lattice units. The

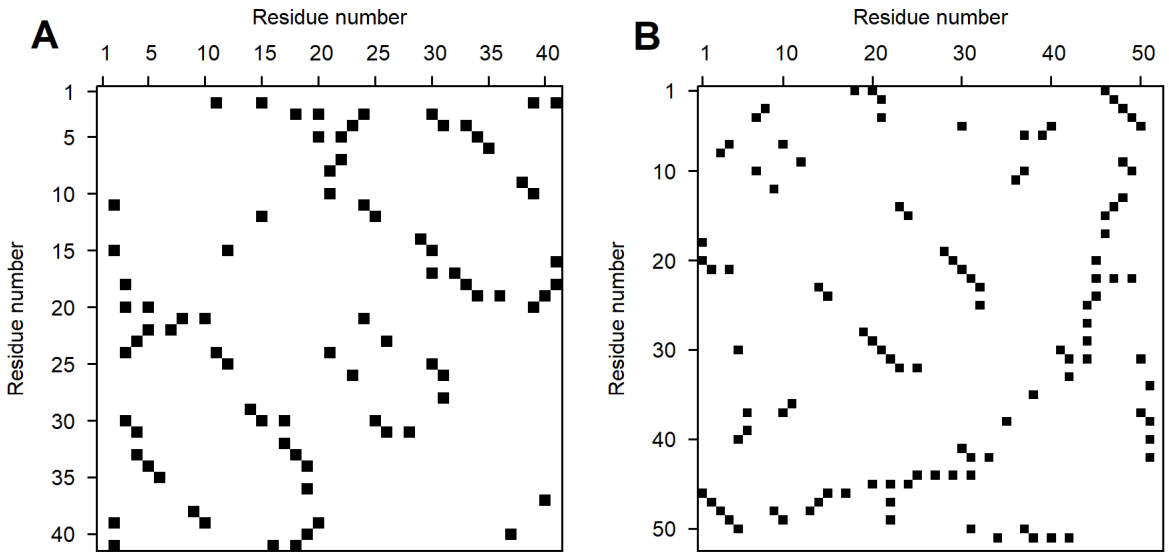

Figure 3.2: Native contact maps of test proteins k31 (A) and k52 (B).

largest dimensions of the proteins' native lattice conformation limits the smallest size of the box to $L = 6$ (for k31), and $L = 8$ (for k52) lattice units.

To model the intermolecular interactions between the beads and the confining box two scenarios are considered. In the first case, the interactions are simply excluded volume (i.e. steric) interactions, while in the second, the box's walls, besides being impenetrable, are also considered uniformly hydrophobic so that stabilizing interactions may be established between the hydrophobic beads in the protein and the box's walls. The intermolecular interactions compete with the intramolecular ones that drive the folding process. At this point it should be stressed that we do not explicitly consider intramolecular hydrophobic interactions, the driving forces for folding being entirely captured by the Gō potential.

Although the hydrophobic residues in real world proteins can be distinguished by their hydropathy index [4] (see Table 1.1), in our model systems all hydrophobic beads are considered equally hydrophobic. In this case the total Hamiltonian of the protein-box system is given by:

$$H\left(\{\vec{r_i}\},\{\sigma_i\}\right) = -\epsilon \sum_{i,j>i+2}^{N} \Delta(\vec{r_i} - \vec{r_j}) - \epsilon_{HP} \sum_{i}^{N} \Delta_{wall}(\vec{r_i}, L), \tag{3.3}$$

where $\{\sigma_i\}$ represents the set of hydrophobic residues in the protein chain (hydrophobic decoration), ϵ_{HP} is the uniform interaction energy parameter that describes interactions between the hydrophobic beads and the walls, and Δ_{wall} is unity if bead i is hydrophobic and lies one lattice spacing away from a wall, zero otherwise.

The Hamiltonian can again be made adimensional through division by ϵ, becoming:

$$H^*\left(\{\vec{r_i}\},\{\sigma_i\}\right) \equiv \frac{H\left(\{\vec{r_i}\},\{\sigma_i\}\right)}{\epsilon} = -\sum_{i,j>i+2}^{N} \Delta(\vec{r_i} - \vec{r_j}) - \eta_{HP} \sum_{i}^{N} \Delta_{wall}(\vec{r_i}, L), \tag{3.4}$$

where $\eta_{HP} = \frac{\epsilon_{HP}}{\epsilon}$ is the relative strength of the hydrophobic interaction with the wall to the native contact interaction.

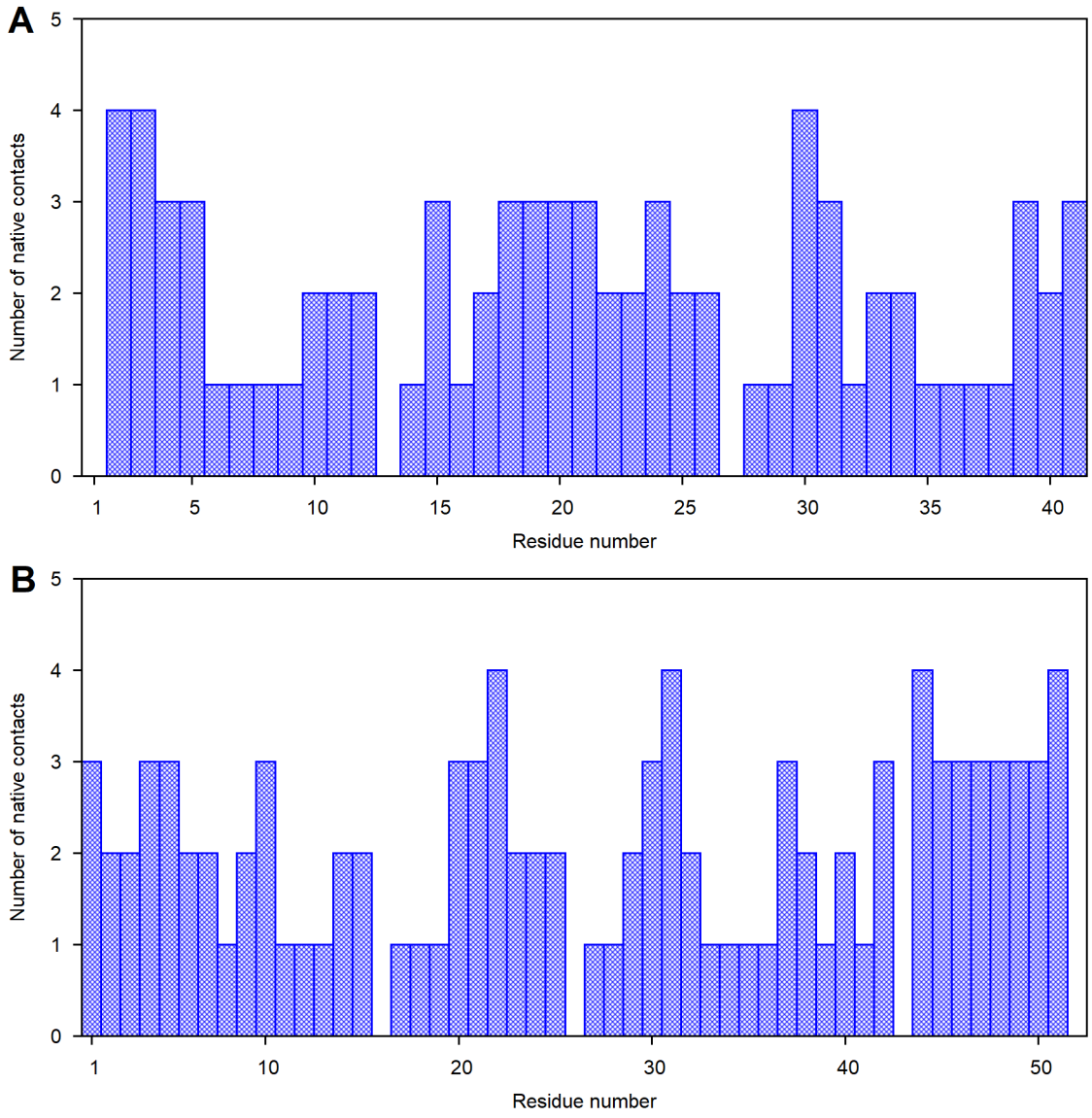

Figure 3.3: Native contact histograms of test proteins k31 (A) and k52 (B).

Given that, throughout its cycle, the hydropathic nature of the chaperonin's walls varies from hydrophobic to hydrophilic and the internal volume of its chambers also changes, we explore a range of relative interaction strength parameter values between $0 \leqslant \eta_{HP} \leqslant 1$, and a range of box sizes, from the above mentioned smallest sizes to $L = 100$.

3.1.3 Hydrophobic decorations

Since we use generalized protein models, it is necessary to decide the amount of hydrophobic content to be assigned to each model system and how it should be distributed over the protein sequence (i.e. its hydrophobic decoration).

As mentioned in Chapter 1, protein folding is mainly driven by the hydrophobic effect [5, 1] and this effect tends to bring hydrophobic residues into contact with each other. Consequently, it appears natural to assume that the residues with higher number of native contacts should be more hydrophobic.

The assignment of hydrophobicity to residues was thus made in order of decreasing number of native contacts.

In the present study, hydrophobicity assignment is coarse grained into two levels only (hydrophobic and neutral) and hence the total number of hydrophobic residues should amount to the same percentage of total residues that is observed in nature for proteins of the same type (same knot type in this specific case, since knotted proteins are being considered). Hence we analyzed the ensembles of knotted proteins in the PDB with 3_1 knots (606 protein sequences) and 5_2 knots (19 protein sequences), to determine the fraction of hydrophobic residues present in each case. According to the hydropathy scale of Kyte and Doolitle [4] amino acids: Isoleucine (Ile), Valine (Val), Leucine (Leu), Phenylalanine (Phe), Cysteine (Cys), Methionine (Met), and Alanine (Ala) are hydrophobic, since they have a positive hydropathy index (see Table 1.1). We found that these seven residues represent 33% of the total amino acid content in the ensemble of 3_1 knotted proteins, and 40% in the ensemble of 5_2 knotted proteins extracted from the PDB.

Given that test protein k31 has 14 residues with either 3 or 4 native contacts (see Fig. 3.3 A), which represent 34% of its residues, and test protein k52 has 19 residues also with either 3 or 4 native contacts (see Fig. 3.3 B), which represent 36.5% of its residues, and that these percentage values are close to the values found for the hydrophobic content of similarly knotted natural proteins, we decided to name these residues the contact core (ctc) of the model proteins and adopt a base hydrophobic decoration scenario, for each model protein, in which these residues are all equally hydrophobic and all other residues are neutral. We named this hydrophobic decoration scenario ctc01. From this base hydrophobic decoration scenario we then prepared a sequence of hydrophobic decorations for each test protein by making neutral the two most connected hydrophobic residues of the previous decoration and making hydrophobic the two most connected neutral residues that had not yet been hydrophobic in previous decorations. This resulted in 13 additional decorations for k31 and 16 additional decorations for k52 for a total of 14 decorations for k31, ctc01 to ctc14, and 17 decorations for k52, ctc01 to ctc17.

In the considered sequences the average number of native contacts of the hydrophobic residues decreases as the sequence number increases, enabling us to study the dependence of the thermodynamic properties of the test proteins on this number.

3.2 Methods

We study the thermodynamic properties of thermal equilibrium states, hence, let's quickly review how thermal equilibrium may be characterized and the formalism specifically applied to protein folding.

3.2.1 Thermal equilibrium

Let's consider a system, S, that may be decomposed into, at least, two subsystems, S_1 and S_2. In the present case let's take S_1 to be the protein molecule and S_2 the aqueous medium in which it resides.

If $\vec{S}$ is the state of system S and $\vec{S_1}$ and $\vec{S_2}$ are the states of subsystems S_1 and S_2, then

$$\vec{S} = (\vec{S_1}, \vec{S_2}). \tag{3.5}$$

Each state has a well defined energy,

$$E \;=\; H_S(\vec{S}), \tag{3.6}$$

$$E_1 = H_{S_1}(\vec{S}_1), \tag{3.7}$$
$$E_2 = H_{S_2}(\vec{S}_2). \tag{3.8}$$

where H_S, H_{S_1} and H_{S_2} are the Hamiltonians of the respective systems.

Several states may, however, have the same energy (degeneracy)

$$H_S(\vec{S}_{\,i}) = E, \qquad \text{for } i = 1 \text{ to } \Omega(E), \tag{3.9}$$
$$H_{S_1}(\vec{S}_{1_j}) = E_1, \qquad \text{for } j = 1 \text{ to } \Omega_1(E_1), \tag{3.10}$$
$$H_{S_2}(\vec{S}_{2_k}) = E_2, \qquad \text{for } k = 1 \text{ to } \Omega_2(E_2). \tag{3.11}$$

where $\Omega(E)$ is the number of different states $\vec{S}_{\,i}$ of system S that all have energy E (i.e. the degeneracy of energy value E) and $\Omega_1(E_1)$ and $\Omega_2(E_2)$ are likewise for systems S_1 and S_2.

For a system with a fixed number of particles, in thermal equilibrium, all degenerate states (microstates) are equiprobable.

Hence, if $P_S(E)$ is the probability of system S having energy E (macrostate probability), the probability of S being in each degenerate microstate is

$$p_S(\vec{S}_{\,i}) = \frac{P_S(E)}{\Omega(E)} = f(E), \tag{3.12}$$

and hence is only a function of the energy of the microstate. Likewise,

$$p_{S_1}(\vec{S}_{1_j}) = \frac{P_{S_1}(E_1)}{\Omega_1(E_1)} = f_1(E_1), \tag{3.13}$$
$$p_{S_2}(\vec{S}_{2_k}) = \frac{P_{S_2}(E_2)}{\Omega_2(E_2)} = f_2(E_2). \tag{3.14}$$

If the particular microstate instance $\vec{S}_{1_j}$ that S_1 assumes at some generic time instant is statistically independent of the particular microstate instance $\vec{S}_{2_k}$ that S_2 assumes at the same time instant (i.e. if interactions are short-range and internal to each subsystem only, thus negligible between subsystems), then $p_{S_1}(\vec{S}_{1_j})$ and $p_{S_2}(\vec{S}_{2_k})$ are statistically independent and taking $\vec{S}_{\,i} = (\vec{S}_{1_j}, \vec{S}_{2_k})$,

$$p_S(\vec{S}_{\,i}) = p_{S_1}(\vec{S}_{1_j})\, p_{S_2}(\vec{S}_{2_k}) \Leftrightarrow \tag{3.15}$$
$$\Leftrightarrow f(E) = f_1(E_1)\, f_2(E_2) \Leftrightarrow \tag{3.16}$$
$$\Leftrightarrow \log f(E) = \log f_1(E_1) + \log f_2(E_2). \tag{3.17}$$

In the present case we achieve this subsystem separation by incorporating the hydrophobic effect of the medium in the intramolecular Hamiltonian through the native centric Gō potential, the medium having no further influence on the protein molecule than constituting a heat reservoir.

These three functions can be expanded in power series on their single variables,

$$\log f(E) = \alpha - \beta E + \gamma E^2 + \text{h.o.t.} \tag{3.18}$$
$$\log f_1(E_1) = \alpha_1 - \beta_1 E_1 + \gamma_1 E_1^2 + \text{h.o.t.} \tag{3.19}$$
$$\log f_2(E_2) = \alpha_2 - \beta_2 E_2 + \gamma_2 E_2^2 + \text{h.o.t.}\,. \tag{3.20}$$

Since interactions between subsystems are negligible, energy is extensive,

$$E = E_1 + E_2, \tag{3.21}$$

and equality (3.18) becomes

$$\log f(E_1 + E_2) = \alpha - \beta(E_1 + E_2) + \gamma(E_1 + E_2)^2 + \text{h.o.t.}\,. \tag{3.22}$$

The second and higher order terms in this equality all involve products of powers of E_1 and E_2.

Since no such products can be formed on the right-hand side of equality (3.17), where only simple powers of these two variables may be present, we are forced to conclude that only the zero and first order terms can be present on equalities (3.22) and (3.18) and consequently also on equalities (3.19) and (3.20).

We thus conclude that

$$\log f(E) = \alpha - \beta E = \alpha - \beta E_1 - \beta E_2 \tag{3.23}$$

$$\log f_1(E_1) = \alpha_1 - \beta_1 E_1 \tag{3.24}$$

$$\log f_2(E_2) = \alpha_2 - \beta_2 E_2. \tag{3.25}$$

and hence, from (3.17), that

$$\alpha = \alpha_1 + \alpha_2, \tag{3.26}$$

$$\beta = \beta_1 = \beta_2. \tag{3.27}$$

Defining temperature, $T \equiv \frac{1}{k_B \beta} \Leftrightarrow \beta = \frac{1}{k_B T}$, where k_B is the Boltzmann constant, we conclude that, in thermal equilibrium, all systems have the same temperature, and that the probability of system S being in a microstate $\vec{S}$ of energy $E = H_S(\vec{S})$ is

$$p_S(\vec{S}) = f(E) = e^{\alpha - \beta E} = C\, e^{-\frac{E}{k_B T}} = C\, e^{-\frac{H_S(\vec{S})}{k_B T}}, \tag{3.28}$$

where $C = e^{\alpha}$ is a value that does not depend on the state energy E, yet to be determined.

This probability distribution is known as the Canonical distribution and the factor $e^{-\frac{H_S(\vec{S})}{k_B T}}$ is known as the Boltzmann factor.

Since the system must always be in some microstate, the sum of this probability over all possible microstates must add to unity,

$$\sum_{\vec{S}} p_S(\vec{S}) = 1 = C \sum_{\vec{S}} e^{-\frac{H_S(\vec{S})}{k_B T}}. \tag{3.29}$$

Defining the partition function, Z, as

$$Z = \sum_{\vec{S}} e^{-\frac{H_S(\vec{S})}{k_B T}}, \tag{3.30}$$

we conclude that $C = \frac{1}{Z}$ and hence that

$$p_S(\vec{S}) = \frac{e^{-\frac{H_S(\vec{S})}{k_B T}}}{Z}. \tag{3.31}$$

Likewise, for subsystem S_1, the protein molecule under study, we can conclude that the probability of it being in a microstate $\vec{S}_1$ of energy $E_1 = H_{S_1}(\vec{S}_1)$ is

$$p_{S_1}(\vec{S}_1) = f_1(E_1) = \frac{e^{-\frac{H_{S_1}(\vec{S}_1)}{k_B T}}}{Z_1}, \tag{3.32}$$

where Z_1 is

$$Z_1 = \sum_{\vec{S}_1} e^{-\frac{H_{S_1}(\vec{S}_1)}{k_B T}}. \tag{3.33}$$

and $H_{S_1}(\vec{S}_1)$ is either the Hamiltonian (3.1), when folding occurs in bulk or under steric confinement conditions, or the Hamiltonian (3.3), when folding takes place under hydrophobic confinement conditions.

To generate samples from this distribution we used the Metropolis Monte Carlo method and, hence, let's quickly revisit its foundations.

3.2.2 The Metropolis Monte Carlo method

The systems being studied have discrete degrees of freedom and no explicit kinetic component. Hence, their state vectors are simply their conformations, e.g. $a = \{\vec{r}_1, \ldots, \vec{r}_N\}$, where N is the number of residues in the lattice protein being considered.

Let's designate by $p(a,t)$ the probability of the system having conformation a at time t and by $w(a \to b)$ the time rate of transition from conformation a to conformation b, the probability of transition from a to b in a small time interval Δt being $w(a \to b)\,\Delta t$.

The master equation for $p(a,t)$ thus is

$$\Delta p = p(a, t+\Delta t) - p(a,t) = -\sum_{b \neq a} w(a \to b)\,\Delta t\, p(a,t) + \sum_{b \neq a} w(b \to a)\,\Delta t\, p(b,t) \Leftrightarrow \tag{3.34}$$

$$\Leftrightarrow \frac{\Delta p}{\Delta t} = -\sum_{b \neq a} w(a \to b)\, p(a,t) + \sum_{b \neq a} w(b \to a)\, p(b,t). \tag{3.35}$$

In thermal equilibrium at temperature T, $p(a,t)$ becomes time independent and acquires the Canonical distribution (3.32) derived in the previous section (all system denoting subscripts are now dropped since only one system, the protein molecule, remains of interest)

$$p(a,t) = p_{eq}(a) = \frac{e^{-\frac{H(a)}{k_B T}}}{Z}, \tag{3.36}$$

and

$$\frac{\Delta p}{\Delta t} = 0 \Leftrightarrow \sum_{b \neq a} w(a \to b)\, p_{eq}(a) = \sum_{b \neq a} w(b \to a)\, p_{eq}(b). \tag{3.37}$$

A sufficient condition for (3.37) to hold is the so called detailed balance condition

$$w(a \to b)\, p_{eq}(a) = w(b \to a)\, p_{eq}(b). \tag{3.38}$$

Combining this condition with the Canonical distribution (3.36) we conclude that transition rates should satisfy

$$\frac{w(a \to b)}{w(b \to a)} = \frac{p_{eq}(b)}{p_{eq}(a)} = e^{-\frac{H(b)-H(a)}{k_B T}}. \tag{3.39}$$

Two methods are commonly used to enforce this condition in a Monte Carlo simulation

1. The Metropolis method [36]

$$w(a \to b) = \begin{cases} e^{-\frac{H(b)-H(a)}{k_B T}} & \text{if} \quad H(b) > H(a) \\ 1 & \text{if} \quad H(b) \leqslant H(a) \end{cases} ; \tag{3.40}$$

2. The heat-bath method

$$w(a \to b) = \frac{e^{-\frac{H(b)}{k_B T}}}{e^{-\frac{H(a)}{k_B T}} + e^{-\frac{H(b)}{k_B T}}}. \tag{3.41}$$

We adopt the Metropolis method in this work.
The Monte Carlo simulation algorithm used was, therefore, as follows:

1. Initialize the data structures with an arbitrary initial conformation, a, and calculate $H(a)$;

2. Generate a trial conformation, b, by applying a move from an adequate move set to the current conformation (the particular move set adopted is described in section 3.2.4);

3. Check if the move has taken any residue to a position already occupied by another residue (self-avoidance check) or, if folding is taking place within a confining box, to a position occupied by the wall (steric confinement check). If so reject the trial conformation and return to the previous step to generate a new one;

4. Calculate $H(b)$ using (3.1), if folding is occurring in bulk or under steric confinement alone, or (3.3), if folding is occurring under hydrophobic confinement conditions;

5. Calculate $w(a \to b)$ using (3.40);

6. Generate a random number, r, uniformly distributed in the interval $[0, 1[$;

7. Adopt conformation b if $r < w(a \to b)$, otherwise keep conformation a;

8. Repeat steps 2 to 7, which constitute one Monte Carlo step, until the required number of Monte Carlo steps has been performed;

9. Every time the number of Monte Carlo steps already performed is an integer multiple of a pre-defined sampling interval, calculate relevant physical quantities for the current conformation and record them in a file for latter data analysis.

The relevant physical quantities calculated for each sampled conformation are discussed in section 3.2.5 and the required number of Monte Carlo steps and adequate sampling interval are discussed in section 3.2.8 below.

The software implementation of this method used in the present work had been previously developed in C by Faísca and co-workers [17, 18] and received only minor improvements in this work, among which, the hydrophobic wall interaction energy function and, due to its excellent statistical properties, notably a period of $2^{19937} - 1$ and adequate speed, the adoption of the Mersenne twister random number generator mt19937 [45], implemented in the Gnu Scientific Library (GSL).

3.2.3 The Replica Exchange method

When several replicas of the same system are simulated at different temperatures concurrently, there exists the opportunity to improve the performance of the Metropolis Monte Carlo method, described in the preceding section, through exchange of system conformations between replicas. This method, which reduces equilibration time and promotes faster and wider conformation space exploration, is known in the literature as Replica Exchange or Parallel Tempering, was first introduced for spin-glasses by Swendsen and Wang in 1986 [46] and was given the formulation about to be presented, by Hukushima and Nemoto ten years latter [47].

To understand how the method preserves the statistical properties of the distributions sampled, let's consider two replicas of the same system, both in thermal equilibrium, one at temperature T_1 and the other at temperature T_2. The probability of the T_1 replica having conformation a is

$$p_{eq,T_1}(a) = \frac{e^{-\frac{H(a)}{k_B T_1}}}{Z_1}, \tag{3.42}$$

and the probability of the T_2 replica having conformation b is

$$p_{eq,T_2}(b) = \frac{e^{-\frac{H(b)}{k_B T_2}}}{Z_2}. \tag{3.43}$$

Because the two replicas are non-interacting, the joint probability of them simultaneously having these two conformations is the product of these two probabilities

$$p_{eq,T_1,T_2}(a,b) = \frac{e^{-\left(\frac{H(a)}{k_B T_1}+\frac{H(b)}{k_B T_2}\right)}}{Z_1\, Z_2}. \tag{3.44}$$

The joint probability of them simultaneously having these same two conformations but exchanged between them is

$$p_{eq,T_1,T_2}(b,a) = \frac{e^{-\left(\frac{H(b)}{k_B T_1}+\frac{H(a)}{k_B T_2}\right)}}{Z_1\, Z_2}. \tag{3.45}$$

These joint probability distributions, and hence both replica's thermal equilibrium distributions, will be preserved if the conformations are exchanged between the replicas in compliance with the detailed balance condition

$$w((a,b)\to(b,a))\, p_{eq,T_1,T_2}(a,b) = w((b,a)\to(a,b))\, p_{eq,T_1,T_2}(b,a). \tag{3.46}$$

and hence,

$$\frac{w((a,b)\to(b,a))}{w((b,a)\to(a,b))} = \frac{p_{eq,T_1,T_2}(b,a)}{p_{eq,T_1,T_2}(a,b)} = \tag{3.47}$$

$$= e^{-\left(\frac{H(b)}{k_B T_1}+\frac{H(a)}{k_B T_2}-\frac{H(a)}{k_B T_1}-\frac{H(b)}{k_B T_2}\right)} = \tag{3.48}$$

$$= e^{-\left(\frac{H(b)-H(a)}{k_B T_1}+\frac{H(a)-H(b)}{k_B T_2}\right)} = \tag{3.49}$$

$$= e^{-\left(\frac{1}{k_B T_1}-\frac{1}{k_B T_2}\right)(H(b)-H(a))}. \tag{3.50}$$

The Metropolis method can, once again, be used to enforce this condition on the simulations. Defining $\Delta \equiv \left(\frac{1}{k_B T_1}-\frac{1}{k_B T_2}\right)(H(b)-H(a))$

$$w((a,b)\to(b,a)) = \begin{cases} e^{-\Delta} & \text{if} \quad \Delta > 0 \\ 1 & \text{if} \quad \Delta \leqslant 0 \end{cases}. \tag{3.51}$$

A Replica exchange step consists, therefore, in

1. Randomly select two replicas from the set of concurrent replica simulations;
2. From the two conformations and temperatures of the selected replicas calculate Δ;
3. Calculate $w((a,b) \rightarrow (b,a))$ using (3.51);
4. Generate a random number, r, uniformly distributed in the interval $[0,1[$;
5. Exchange conformations between the replicas if $r < w((a,b) \rightarrow (b,a))$, otherwise keep the conformations in the same replicas;
6. Perform steps 1 to 5 every time the number of Monte Carlo steps already performed is an integer multiple of a predefined replica exchange interval (the choice of an adequate replica exchange interval is discussed in section 3.2.8 below).

The software implementation of this method had been previously developed in C by Faísca and co-workers [17, 18] using the Message Passing Interface (MPI) stack, for inter-process synchronization and communication, and received important improvements, in terms of performance and functionality, in this work. The implementation used is presented in Appendix A.

3.2.4 Exploring conformation space: The kink-jump move set

To generate a trial conformation, as required by step 2 of the Monte Carlo simulation algorithm described in section 3.2.2, first of all, a random residue is selected on the lattice protein. If one of the terminal residues is selected the move attempted is an end-move (Fig. 3.4 A). This consists in rotating the end residue around its adjacent residue to one, randomly selected, of the other 5 positions that are also near neighbors of that residue. If none of these positions is vacant, another residue must be selected. If the selected residue is not a terminal one, then if it and its two adjacent residues lie in a straight line, it cannot be moved and another residue must be selected. If the selected residue and its two adjacent residues are not in a straight line they define three vertices of a square in the plane that contains them. In this case, first, a corner-flip move (Fig. 3.4 B) is attempted. This consists in moving the selected residue to the fourth corner of the square i.e. the one diagonally opposed to it. However, if this site is already occupied, the corner-flip move cannot be performed and if this blocking residue is not connected either to the preceding or the succeeding residue of the selected residue, the selected residue cannot be moved and a new residue must be selected. If the blocking residue is connected either to the preceding or the succeeding residue of the selected residue then the selected residue is paired with the residue connected to the blocking residue and an attempt is made to rotate the pair around the axis defined by the other two corners of the square, into one, randomly selected, of the other three possible positions in the lattice that this pair might occupy. This is known as the crank-shaft move (Fig. 3.4 C).

These three moves, end, corner-flip and crank-shaft, constitute the kink-jump move set. The first two moves were introduced by Verdier and Stockmayer in 1962 [48] in their studies of lattice polymers. However, simulation equilibration proved to be quite slow when only these two moves were used and this prompted Lax and Brender [49] to introduce the crank-shaft move in 1977.

Any possible conformation of the lattice protein can be reached from any other conformation using only these three moves and, thus, this move set is said to be ergodic, ensuring that the random walk in

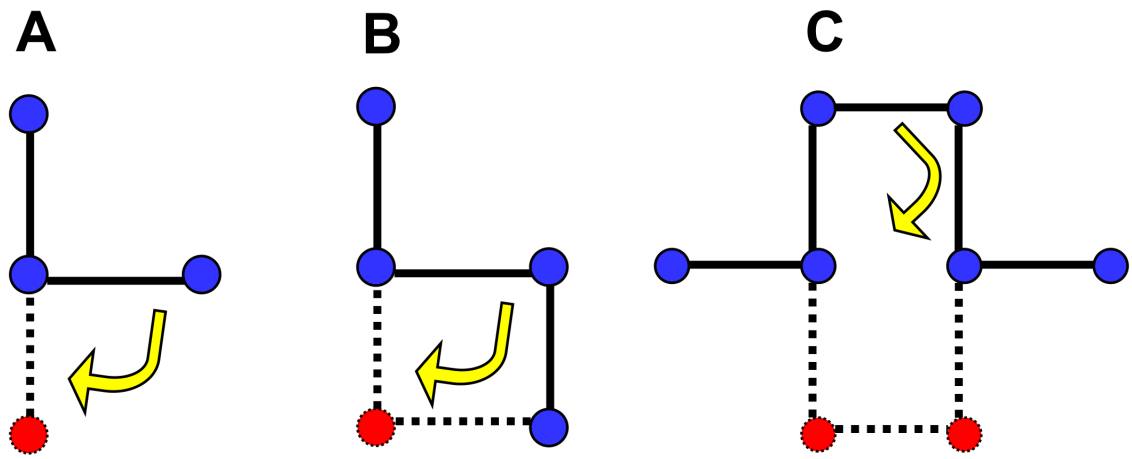

Figure 3.4: The kink-jump move set. The emd move (A), the corner-flip move (B) and the crank-shaft move (C).

conformation space created by the successive application of these moves, converges to the equilibrium distribution irrespective of the initial conformation.

3.2.5 Relevant physical quantities

The relevant physical quantities, mentioned in section 3.2.2, that are calculated and recorded for each sampled conformation are the intramolecular energy, the wall interaction energy (if folding is taking place under hydrophobic confinement), the total energy (the value of the Hamiltonian), which is the sum of both, the knottiness of the conformation and its gyration radius. The intramolecular energy is the first term in equality (3.4) and the wall interaction energy the second term in this equality.

From the intramolecular energy a reaction coordinate, Q, can be defined by dividing the intramolecular energy of a conformation by the intramolecular energy of the native conformation. The reaction coordinate thus varies between 0, for an unfolded conformation with no native contacts, and 1, for the fully folded native conformation,

Since no particular mass is assigned to each residue, the gyration radius, R_g, is just the root-mean-square distance of the residues to the geometric center of the conformation and, hence, is just a rough indicator of how folded or unfolded the conformation is.

The knottiness of the conformation, λ, is determined using the Koniaris-Muthukumar-Taylor (KMT) method [13, 14]. This method is applicable both to lattice and off-lattice representations and consists in successively scanning the conformation from end to end, selecting three consecutive residues (beads) at each step and applying to them in succession two operations: bead deletion and bead movement. The bead deletion operation was introduced by Koniaris and Muthukumar in 1991 [13] and the bead movement operation by Taylor in 2000 [14]. Throughout the method, the terminal beads are kept fixed.

Bead deletion consists in first checking if the three beads are on a straight line. If they are, then the middle bead can be deleted and the first bead linked directly to the third by a straight line without changing the topology of the conformation. If they are not in a straight line, then they form a triangle. If this triangle is not crossed by any link between two other beads then again the middle bead can be deleted and the first bead linked directly to the third by a straight line without changing the topology of

the conformation.

If the triangle is crossed by at least one link, then bead movement is attempted. This consists in calculating the geometric center of the triangle and then checking if the chevron shaped area enclosed by the straight line segments joining the first and last beads to the middle bead and to the center of the triangle is crossed by any link between any other two beads. If the chevron area is not crossed by any link, the middle bead can be moved to the center of the triangle without changing the topology of the conformation. If the chevron area is crossed then the middle bead cannot be moved. Each of these operations, if successful, shortens the overall length of the conformation backbone.

The method iterates until no further bead deletions or movements are possible. If at this point only two beads remain, the two terminal beads, of course, we can conclude that the conformation is not knotted. If, on the other hand, more than two beads remain, we can conclude that the conformation is knotted.

The software implementation of this method was totally redeveloped in C in the course of the present work and is presented in Appendix B.

3.2.6 Simulation temperature

In sections 3.1.1 and 3.1.2 we introduced the adimensiomal Hamiltonian H^* through equalities (3.2) and (3.4). This is convenient since, as then emphasized, the intramolecular energy value of a conformation becomes simply the symmetric of the number of native contacts present in that conformation.

Because we are studying thermal equilibrium states, in all methods, the Hamiltonian always appears divided by the product $k_B\,T$ and hence it is convenient to introduce also an adimensional temperature, T^*, such that, for a generic conformation a,

$$\frac{H(a)}{k_B T} = \frac{H^*(a)}{T^*} \Leftrightarrow T^* = \frac{H^*(a)}{H(a)} k_B T = \frac{k_B T}{\epsilon}, \tag{3.52}$$

where ϵ is, let us recall, the average energy required to break a native contact.

The energy involved in protein contacts has been extensively studied by Miyazawa and Jernigan. Their first approach to this problem, published in 1985 [50], involved only an attractive potential but in 1996 [51] they revised their inter-residue contact energies and introduced an additional repulsive energy term to account for repulsive forces at high packing densities.

We used their 1996 results to estimate the average energy of a protein native contact by taking the simple arithmetic average over the 20 naturally occurring amino acids of the attractive contact energy of each amino acid due to hydrophobicity. This average attractive energy was then corrected by an estimate of the average effect of the repulsive term. Our estimate is that $\epsilon \approx 1$ kcal mol^{-1}.

Given that 1 cal = 4.184 J, $N_A = 6.0221 \times 10^{23}$ mol^{-1}, $k_B = 1.3806 \times 10^{-23}$ J K^{-1}, adimensional temperature becomes

$$T^* = \frac{T}{503\,\mathrm{K}}. \tag{3.53}$$

Since physiologic temperature is $T_{physiologic} \approx 310\,\mathrm{K}$, we adopt an adimensional value for the physiologic temperature of

$$T^*_{physiologic} = 0.6. \tag{3.54}$$

Hence we investigate an adimensional temperature interval symmetric around this value of $0.2 \leqslant T^* \leqslant 1.0$. A total of 41 replicas was used in each simulation, their adimensional temperatures being separated by increments of 0.02.

3.2.7 Data analysis

Data analysis is performed entirely using the Weighted Histogram Analysis Method (WHAM). WHAM is an extension of the Multiple Histogram Technique introduced by Ferrenberg and Swendsen in 1989 [52] and was applied for the first time to the analysis of a complex biomolecular problem by Kumar et al. in 1992 [53].

Before we discuss the method's foundations let's introduce the notation used.

Notation

Energy bins

- Energy: H (Hamiltonian);
- Number of H bins: I;
- Index of H bin: i (varies from 0 to $I-1$);
- Central value of bin i: H_i.

Property bins

- Property can be any quantity that depends only on the system conformation (e.g. reaction coordinate,Q, knottiness, λ, *etc.*). Q will be used as example below.
- Number of Q bins: J;
- Index of Q bin: j (varies from 0 to $J-1$);
- Central value of bin j: Q_j.

Discrete temperature values

- Number of temperature values (replicas): K;
- Index of T value: k (varies from 0 to $K-1$);
- kth temperature value: T_k.
- Inverse temperature parameter, $\beta = \frac{1}{k_B T}$, for replica k: $\beta_k = \frac{1}{k_B T_k}$.

Samples

- Each sample is a tuple composed of the values of the relevant physical properties: ($E_{intramolecular}$, Q, $E_{interaction}$, H, λ, R_g);
- Number of samples taken in replica k: N_k
- Index of sample: n (varies from 0 to $N_k - 1$);
- Property values in sample n of replica k: e.g. H_{kn}, Q_{kn}

Histograms

- N_{ijk}: Number of samples in replica k that have H value within bin i and property value (e.g. Q value) within bin j;
- Summation over all the values of an index removes it from the histogram symbol. Hence, for instance, $N_{ij} = \sum_k N_{ijk}$ is the number of samples in all replicas that have H value within bin i and property value (e.g. Q value) within bin j.

Method foundations

The purpose of WHAM is to determine the number of microstates in macrostate ensembles, thereby enabling direct calculation of the partition function and all thermodynamic properties of the system.

By definition, entropy, S, is

$$S \equiv k_B \left(\log Z + \beta U \right), \tag{3.55}$$

where k_B is Boltzmann's constant, Z is the system's partition function, $\beta \equiv \frac{1}{k_B T}$ is the inverse temperature parameter and $U \equiv \langle H \rangle$ is the system's internal energy.

Hence,

$$TS = k_B T \log Z + U, \tag{3.56}$$

and the Helmholtz free energy is

$$F \equiv U - TS = -k_B T \log Z. \tag{3.57}$$

The system's partition function may, consequently, be written as

$$Z = e^{-\frac{F}{k_B T}}, \tag{3.58}$$

and, introducing the reduced free energy

$$f \equiv \frac{F}{k_B T} = \beta F, \tag{3.59}$$

we may write

$$Z = e^{-f}. \tag{3.60}$$

At temperature T_k of the k replica and defining $f_k \equiv f(T_k)$

$$Z(T_k) = e^{-f_k}. \tag{3.61}$$

The probability of the system being in a microstate of total energy, H, within bin i, at temperature T_k, as long as the number of uncorrelated samples, N_k, drawn from replica k is much larger than the number of bins ($N_k \ggg I$), may be estimated as

$$p_{ik} = \frac{N_{ik}}{N_k}. \tag{3.62}$$

In thermal equilibrium the system's total energy is canonically distributed

$$p_{ik} = \frac{\Omega_i \, e^{-\beta_k H_i}}{Z(T_k)}, \tag{3.63}$$

where Ω_i is the number of microstates that have total energy, H, within bin i.

Substituting (3.61) into this expression we obtain

$$p_{ik} = \Omega_i \, e^{f_k - \beta_k H_i}. \tag{3.64}$$

Ω_i is independent of the temperature and hence, from each replica we may obtain an estimate of Ω_i. Let's name it Ω_{ik}

$$\Omega_{ik} = \frac{p_{ik}}{e^{f_k - \beta_k H_i}} = \frac{N_{ik}}{N_k \, e^{f_k - \beta_k H_i}}. \tag{3.65}$$

From this set of k estimates a maximum likelihood estimate of Ω_i can be derived through choosing the vector f_k that minimizes the variance of Ω_{ik} around the Ω_i estimate.

The maximum likelihood estimator of Ω_i is the weighted average [54]

$$\Omega_i = \frac{\sum_k \frac{1}{\delta^2\Omega_{ik}}\Omega_{ik}}{\sum_k \frac{1}{\delta^2\Omega_{ik}}}, \tag{3.66}$$

where $\delta^2\Omega_{ik}$ is the uncertainty associated with the Ω_{ik} estimate. Chodera [38] has shown this uncertainty to be

$$\delta^2\Omega_{ik} = \frac{\Omega_i}{N_k\, e^{f_k-\beta_k H_i}}, \tag{3.67}$$

thus

$$\frac{1}{\delta^2\Omega_{ik}} = \frac{N_k\, e^{f_k-\beta_k H_i}}{\Omega_i}, \tag{3.68}$$

and from (3.65)

$$\frac{1}{\delta^2\Omega_{ik}}\Omega_{ik} = \frac{N_{ik}}{\Omega_i}. \tag{3.69}$$

Hence

$$\sum_k \frac{1}{\delta^2\Omega_{ik}} = \frac{\sum_k N_k\, e^{f_k-\beta_k H_i}}{\Omega_i}, \tag{3.70}$$

$$\sum_k \frac{1}{\delta^2\Omega_{ik}}\Omega_{ik} = \frac{\sum_k N_{ik}}{\Omega_i}, \tag{3.71}$$

and the maximum likelihood estimator of Ω_i, (3.66), becomes

$$\Omega_i = \frac{\sum_k N_{ik}}{\sum_k N_k\, e^{f_k-\beta_k H_i}}, \tag{3.72}$$

which depends on the reduced free energy vector f_k.

On the other hand, from (3.57) and (3.59)

$$f = -\log Z. \tag{3.73}$$

The partition function being

$$Z = \sum_i \Omega_i\, e^{-\beta H_i}, \tag{3.74}$$

at temperature T_k

$$Z(T_k) = \sum_i \Omega_i\, e^{-\beta_k H_i}, \tag{3.75}$$

and the reduced free energy vector, f_k, can thus be obtained from the number of states vector, Ω_i, through

$$f_k = f(T_k) = -\log Z(T_k) = -\log \sum_i \Omega_i\, e^{-\beta_k H_i}. \tag{3.76}$$

The number of states vector, Ω_i, and the reduced free energy vector, f_k, are thus the solutions of the system of equations

$$\begin{cases} \Omega_i &= \frac{\sum_k N_{ik}}{\sum_k N_k\, e^{f_k-\beta_k H_i}} \\ f_k &= -\log \sum_i \Omega_i\, e^{-\beta_k H_i}. \end{cases} \tag{3.77}$$

This system of equations can be solved iteratively by starting with the condition $f_k = 0, \forall_k$, obtaining a first estimate of Ω_i and then iterating until the relative change in all Ω_i is smaller than a specified tolerance.

Knowledge of the number of states vector, Ω_i, enables direct calculation of several of the system's thermodynamic properties.

Moments of the energy random variable

The mth order moment of the energy random variable, H, is the expected value of its mth power, $\langle H^m \rangle$.

Given that the partition function is

$$Z = \sum_i \Omega_i \, e^{-\beta H_i}, \tag{3.78}$$

and the probability of the system being in a microstate of total energy, H, within bin i, at temperature T is

$$p_i = \frac{\Omega_i \, e^{-\beta H_i}}{Z}, \tag{3.79}$$

the mth moment of H is

$$\langle H^m \rangle = \sum_i H_i^m p_i = \frac{\sum_i H_i^m \Omega_i \, e^{-\beta H_i}}{Z} = \frac{\sum_i H_i^m \Omega_i \, e^{-\beta H_i}}{\sum_i \Omega_i \, e^{-\beta H_i}}. \tag{3.80}$$

All moments of H may thus be directly calculated once we know the number of states vector, Ω_i.

On the other hand we may also write

$$\frac{\partial^m Z}{\partial \beta^m} = \sum_i (-H_i)^m \Omega_i \, e^{-\beta H_i}, \tag{3.81}$$

and hence,

$$\frac{1}{Z} \frac{\partial^m Z}{\partial \beta^m} = (-1)^m \langle H^m \rangle . \tag{3.82}$$

Internal energy

$$U \equiv \langle H \rangle = \frac{\sum_i H_i \Omega_i \, e^{-\beta H_i}}{\sum_i \Omega_i \, e^{-\beta H_i}} = -\frac{1}{Z} \frac{\partial Z}{\partial \beta}. \tag{3.83}$$

Specific heat

$$\begin{aligned}
C_V(T) &\equiv \left(\frac{\partial U}{\partial T} \right)_V = \frac{\partial U}{\partial \beta} \frac{d\beta}{dT} = \frac{\partial}{\partial \beta} \left(-\frac{1}{Z} \frac{\partial Z}{\partial \beta} \right) \frac{d\beta}{dT} && (3.84) \\
&= \left[\left(\frac{1}{Z^2} \frac{\partial Z}{\partial \beta} \right) \frac{\partial Z}{\partial \beta} - \frac{1}{Z} \frac{\partial^2 Z}{\partial \beta^2} \right] \frac{d\beta}{dT} && (3.85) \\
&= -\frac{d\beta}{dT} \left[\frac{1}{Z} \frac{\partial^2 Z}{\partial \beta^2} - \left(\frac{1}{Z} \frac{\partial Z}{\partial \beta} \right)^2 \right] && (3.86) \\
&= \frac{1}{k_B T^2} \left[\langle H^2 \rangle - \langle H \rangle^2 \right] && (3.87) \\
&= \frac{1}{k_B T^2} \left[\frac{\sum_i H_i^2 \Omega_i \, e^{-\beta H_i}}{\sum_i \Omega_i \, e^{-\beta H_i}} - U^2 \right]. && (3.88)
\end{aligned}$$

Melting temperature

Melting temperature, T_m, is the temperature at which C_V peaks, hence is the temperature at which $\frac{\partial C_V}{\partial T} = 0$.

$$\frac{\partial C_V}{\partial T} = \frac{\partial^2 U}{\partial T^2} = \frac{\left\langle H^3 \right\rangle - \left\langle H \right\rangle^3 - (3\left\langle H \right\rangle + 2k_B T)(\left\langle H^2 \right\rangle - \left\langle H \right\rangle^2)}{k_B^2 T^4}. \tag{3.89}$$

The zero of this function can be determined using a root finding algorithm such as the Brent-Dekker method [55, 56] available in the Gnu Scientific Library (GSL).

Entropy

$$S = k_B \left(\log Z + \beta U\right) = k_B \left(\log \sum_i \Omega_i \, e^{-\beta H_i} + \beta U\right). \tag{3.90}$$

Free energy

$$F = -k_B T \log Z = -k_B T \log \sum_i \Omega_i \, e^{-\beta H_i}. \tag{3.91}$$

Conformation dependent properties

The number of states vector, Ω_i, can be projected over the bins, j, of a conformation dependent system property, e.g. Q, to become a matrix, Ω_{ij}, by replacing the two dimensional histogram N_{ik} in (3.72) by the three dimensional histogram N_{ijk} [57]

$$\Omega_{ij} = \frac{\sum_k N_{ijk}}{\sum_k N_k \, e^{f_k - \beta_k H_i}}. \tag{3.92}$$

Ω_{ij} is now the number of microstates that have both total energy, H, within bin i and property value, Q, within bin j.

Hence, since $N_{ik} = \sum_j N_{ijk}$, from (3.72) and (3.92) we conclude that

$$\Omega_i = \sum_j \Omega_{ij}. \tag{3.93}$$

The partition function thus remains

$$Z = \sum_j \sum_i \Omega_{ij} \, e^{-\beta H_i} = \sum_i \Omega_i \, e^{-\beta H_i}, \tag{3.94}$$

and the reduced free energy vector also remains

$$f_k = f(T_k) = -\log Z(T_k) = -\log \sum_j \sum_i \Omega_{ij} \, e^{-\beta_k H_i} = -\log \sum_i \Omega_i \, e^{-\beta_k H_i}. \tag{3.95}$$

Once the number of states vector Ω_i is known, its projection along any property dimension may consequently be found through

$$\Omega_{ij} = \Omega_i p_{j|i} = \Omega_i \frac{N_{ij}}{N_i} = \Omega_i \frac{\sum_k N_{ijk}}{\sum_k N_{ik}}. \tag{3.96}$$

From the number of states matrix, Ω_{ij}, all relevant thermodynamic properties of the system can be easily calculated.

Probability distributions

Joint probability

$$p_{ij} = \frac{\Omega_{ij}\, e^{-\beta H_i}}{Z} = \frac{\Omega_{ij}\, e^{-\beta H_i}}{\sum_j \sum_i \Omega_{ij}\, e^{-\beta H_i}} = \frac{\Omega_{ij}\, e^{-\beta H_i}}{\sum_i \Omega_i\, e^{-\beta H_i}}. \tag{3.97}$$

Marginal probabilities

$$p_i = \frac{\sum_j \Omega_{ij}\, e^{-\beta H_i}}{Z} = \frac{\sum_j \Omega_{ij}\, e^{-\beta H_i}}{\sum_j \sum_i \Omega_{ij}\, e^{-\beta H_i}} = \frac{\Omega_i\, e^{-\beta H_i}}{\sum_i \Omega_i\, e^{-\beta H_i}}. \tag{3.98}$$

$$p_j = \frac{\sum_i \Omega_{ij}\, e^{-\beta H_i}}{Z} = \frac{\sum_i \Omega_{ij}\, e^{-\beta H_i}}{\sum_j \sum_i \Omega_{ij}\, e^{-\beta H_i}} = \frac{\sum_i \Omega_{ij}\, e^{-\beta H_i}}{\sum_i \Omega_i\, e^{-\beta H_i}}. \tag{3.99}$$

Marginal partition functions

$$Z_i = \sum_j \Omega_{ij}\, e^{-\beta H_i} = \Omega_i\, e^{-\beta H_i}. \tag{3.100}$$

$$Z_j = \sum_i \Omega_{ij}\, e^{-\beta H_i}. \tag{3.101}$$

$$p_i = \frac{Z_i}{Z}. \tag{3.102}$$

$$p_j = \frac{Z_j}{Z}. \tag{3.103}$$

Mean values

$$\begin{aligned} \langle Q \rangle &= \sum_j Q_j p_j = \frac{\sum_j Q_j Z_j}{Z} && (3.104)\\ &= \frac{\sum_j Q_j \sum_i \Omega_{ij}\, e^{-\beta H_i}}{\sum_j \sum_i \Omega_{ij}\, e^{-\beta H_i}} && (3.105)\\ &= \frac{\sum_j Q_j \sum_i \Omega_{ij}\, e^{-\beta H_i}}{\sum_i \Omega_i\, e^{-\beta H_i}}. && (3.106) \end{aligned}$$

Free energy

$$Z_j = \sum_i \Omega_{ij}\, e^{-\beta H_i}. \tag{3.107}$$

$$F_j = -k_B T \log Z_j. \tag{3.108}$$

$$F_j = -k_B T \log \sum_i \Omega_{ij}\, e^{-\beta H_i}. \tag{3.109}$$

Viewed as a function of the equilibrium temperature, $F_j(T)$ is a line bundle.

Transition temperature

The transition temperature, T_f (thermal (un)folding temperature), is the value of T for which the cross section of the free energy line bundle has two minima with equal free energy value.

$$\begin{aligned} F_j &= -k_B T \log Z_j && (3.110)\\ &= -k_B T \log (p_j\, Z) && (3.111)\\ &= -k_B T \log p_j - k_B T \log Z && (3.112)\\ &= F - k_B T \log p_j. && (3.113) \end{aligned}$$

$$\begin{aligned} F_j = F_{j'} &\Leftrightarrow -k_B T \log p_j = -k_B T \log p_{j'} && (3.114)\\ &\Leftrightarrow p_j = p_{j'}. && (3.115) \end{aligned}$$

At the transition temperature, T_f, the probability of the molecule being in the most probable unfolded ensemble is equal to the probability of it being in the native state.

Method output

The software implementation of the WHAM data analysis method was totally redeveloped in C during this project and is presented in Appendix C. It generates 16 output files for each simulation run. Two of these contain only a single numeric value: the melting temperature, T_m^*, and the transition temperature, T_f^*. From the other 14 files 23 plots are produced. We present below an example of each of these plots for the simulation that involves the k31 lattice protein system with hydrophobic decoration ctc01, confined in a box of side length $L = 33$ and relative strength of hydrophobic interaction with the wall $\eta_{HP} = 0.50$.

1. Convergence plots

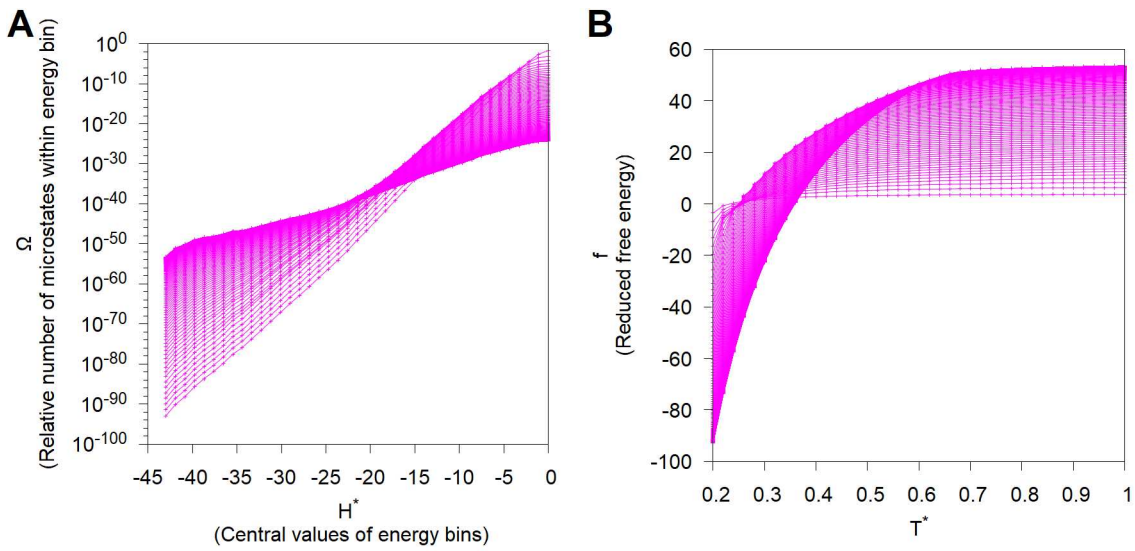

Figure 3.5: Convergence of the number of states vector (A) and of the reduced free energy vector (B) for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

Fig. 3.5 shows the number of states vector and the reduced free energy vector for each iteration of the method. Convergence tolerance was set at 10^{-6}. The points of each iteration are joined by straight line segments for easier visualization of the convergence process.

2. Number of states

   Fig. 3.6 shows the core output of the WHAM method: the number of states vector. Again the points are joined by straight line segments for easier visualization.

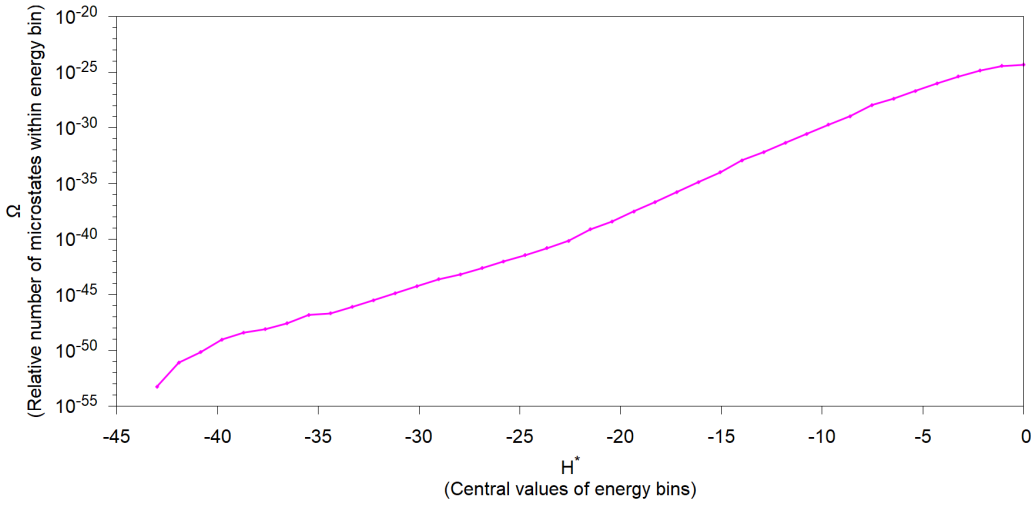

Figure 3.6: The number of states vector for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

3. Properties that depend only on temperature

   Fig. 3.7 A shows the reduced free energy, now as a continuous function of temperature. The red points are the values of the reduced free energy vector, shown only for validation.

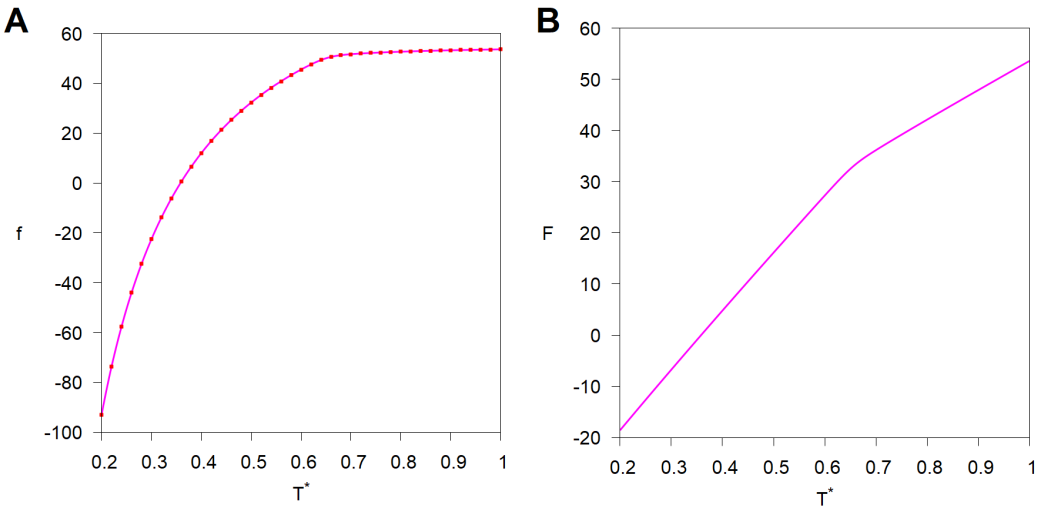

Figure 3.7: Reduced free energy (A) and Helmholtz free energy (B) for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

Fig. 3.7 B shows the Helmholtz free energy. It has two linear regions of different slopes, implying that the folding transition is similar to a first order phase transition. The derivative does not exhibit a discontinuity at the transition temperature because the protein is a finite system.

Fig. 3.8 A shows the internal energy, Fig. 3.8 B the intramolecular energy, Fig. 3.8 C the wall interaction energy and Fig. 3.8 D the specific heat. Red points are calculated by direct averages over the samples drawn from each replica and are shown for validation.

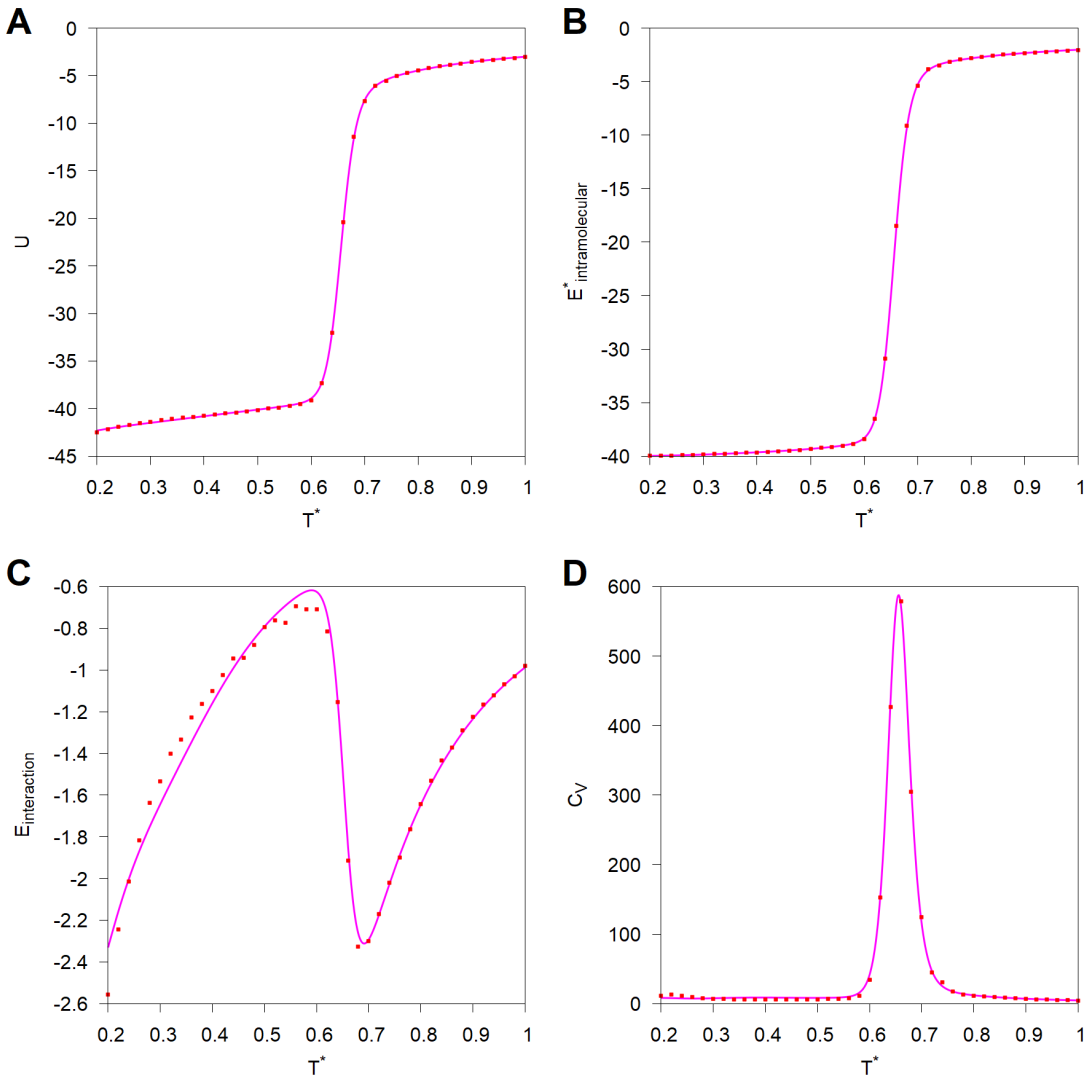

Figure 3.8: Internal energy (A), intramolecular energy (B), wall interaction energy (C) and specific heat (D) for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

Fig. 3.9 A shows the probability of a knotted conformation, Fig. 3.9 B the gyration radius, Fig. 3.9 C the system's entropy and Fig. 3.9 D the system's partition function. Once more, red points are calculated by direct averages over the samples drawn from each replica and are shown for validation.

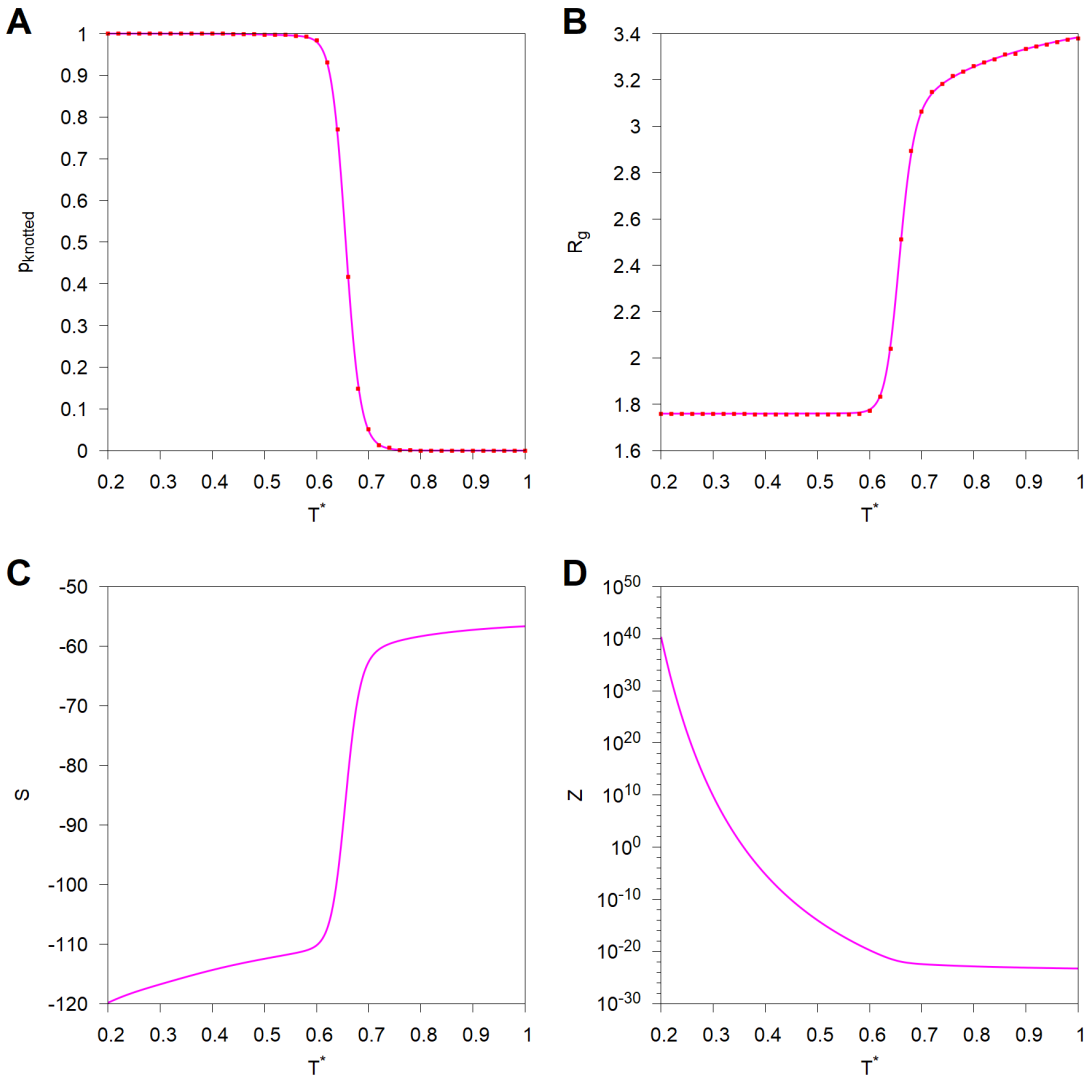

Figure 3.9: Probability of a knotted conformation (A) gyration radius (B), entropy (C) and the partition function (D) for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

4. Number of states matrices

Since, in addition to the total energy, four conformation dependent properties are recorded for every sample, namely, intramolecular energy, wall interaction energy, the knottiness of the conformation and its gyration radius, the number of states vector can be projected over the bins of these four properties giving rise to four number of states matrices. Examples of these from the same simulation are presented in Figures 3.10 A-D.

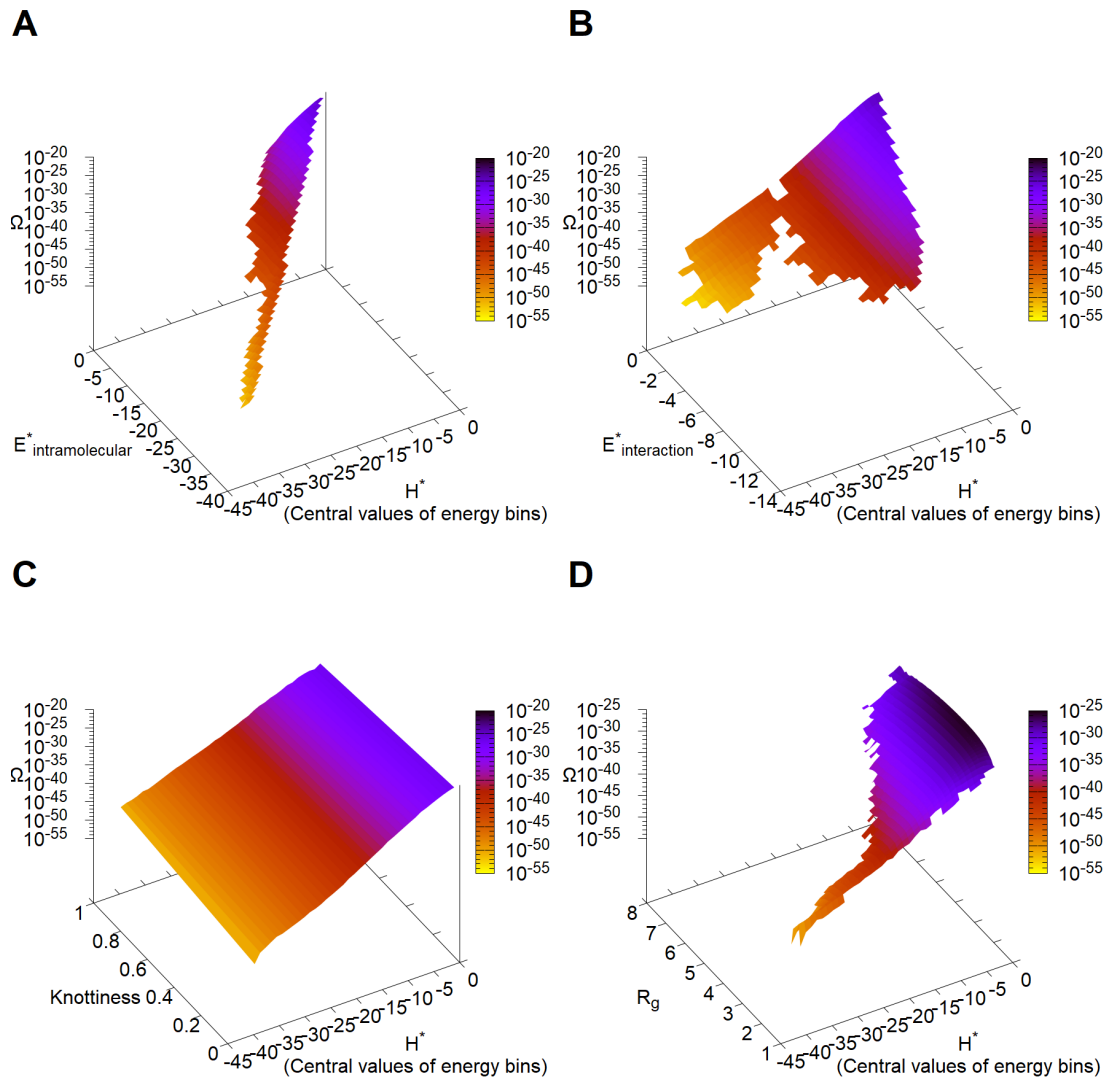

Figure 3.10: The number of states matrix that results from projection over the intramolecular energy bins (A), over the wall interaction energy bins (B), over the knottiness bins (C) and over the gyration radius bins (D) for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

5. Helmholtz free energy

The Helmholtz free energy line bundle is represented in Fig. 3.11 A, with the lines for the discrete values of the reaction coordinate, Q, joined into a surface for ease of visualization. To study the Helmholtz free energy profile for a specific temperature it is helpful to take the value of this energy for the native state as reference. Fig. 3.11 B redraws the Helmholtz free energy line bundle taking the line for the native state, $Q = 1$, as reference (i.e. as zero).

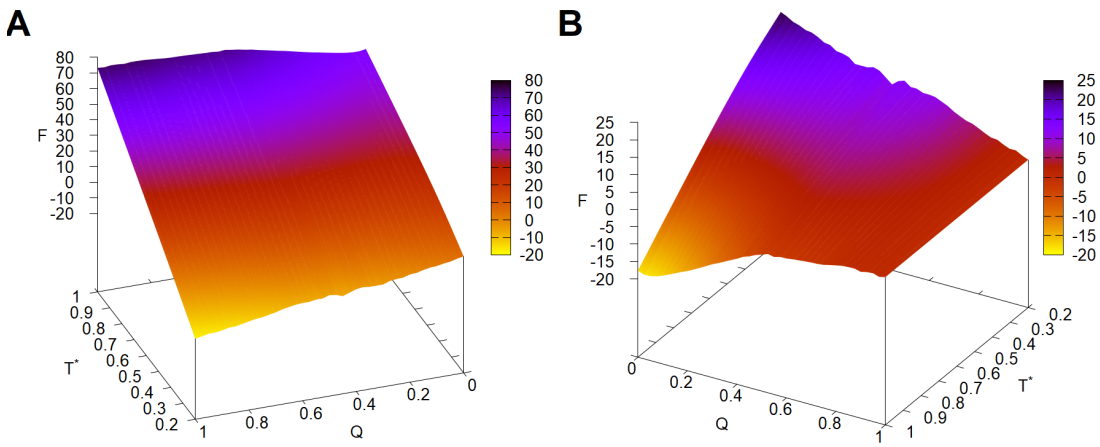

Figure 3.11: The Helmholtz free energy (A) and the Helmholtz free energy relative to the native state (B) for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

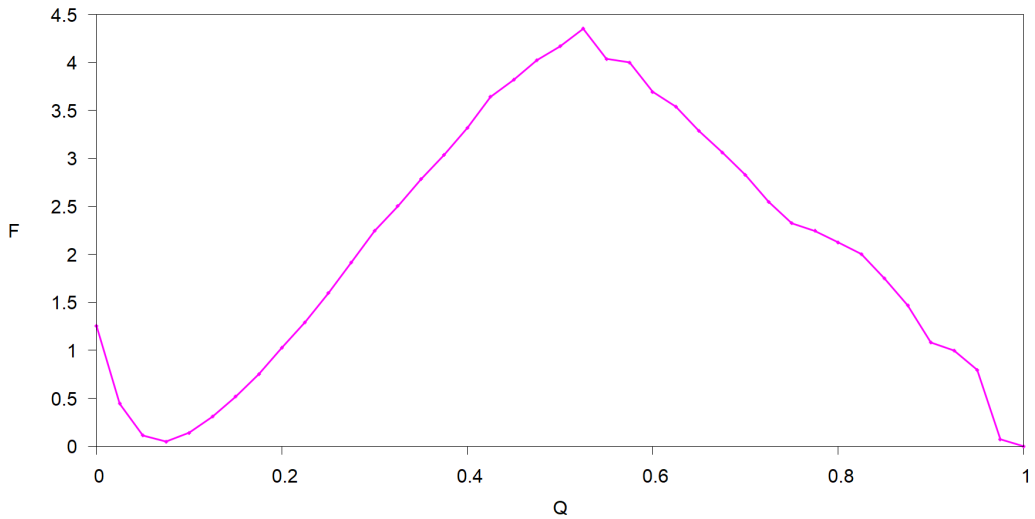

Figure 3.12: The cross-section of the Helmholtz free energy relative to the native state at transition temperature $T_f^* = 0.664$ for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

Fig. 3.12 presents the cross-section of the Helmholtz free energy line bundle at the transition temperature (in this example case $T_f^* = 0.664$). This cross-section is crucial to the study of the folding transition process, enabling identification of any eventual intermediate states that this transition may involve (these appear as local valleys on the profile).

6. Probability distribution of intramolecular energy

   Another way to visualize the folding transition process is as a running probability distribution diagram (name which we will henceforth abbreviate as *probablogram*). Fig. 3.13 shows the probability distribution of the intramolecular energy as a function of temperature. The mean value of the distribution is shown as the green curve, which is identical to the curve in Fig. 3.8 B.

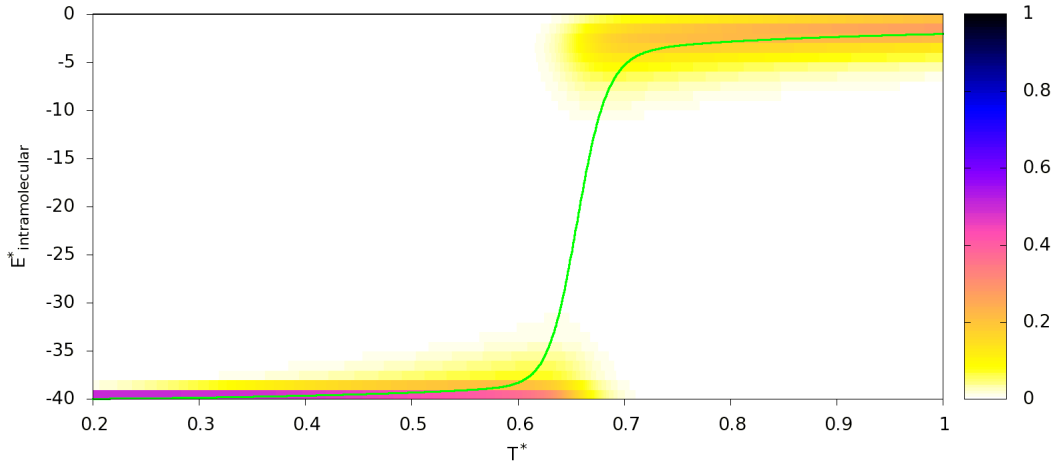

Figure 3.13: The probability distribution of intramolecular energy for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

7. Conditional probability of knotted conformation

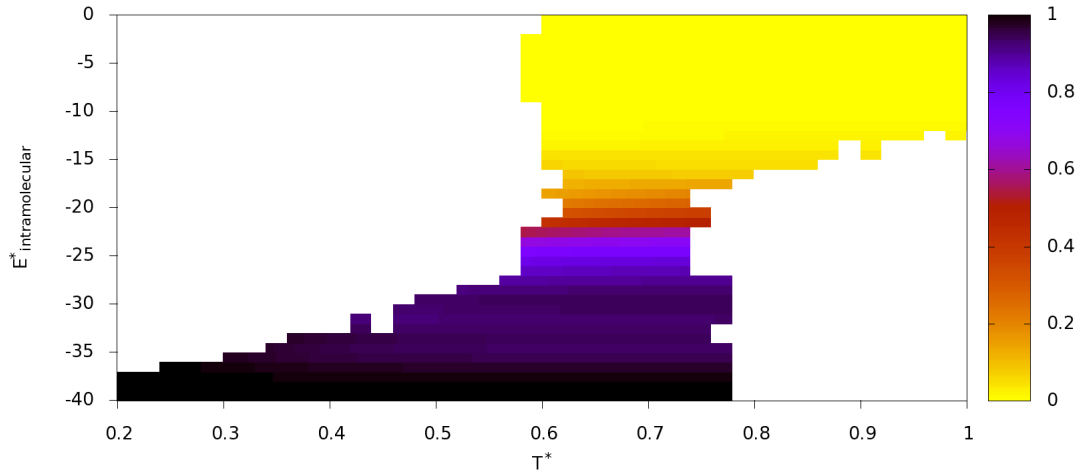

Figure 3.14: The conditional probability of the conformation being knotted given the intramolecular energy, for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

To study the knotting process, the conditional probability of the conformation being knotted given the intramolecular energy, as a function of temperature, is presented as a *probablogram* in Fig. 3.14.

As may be seen, this conditional probability does not depend on the temperature and hence may be fully described by its cross section at the melting temperature. Figure 3.15 exemplifies this cross-section.

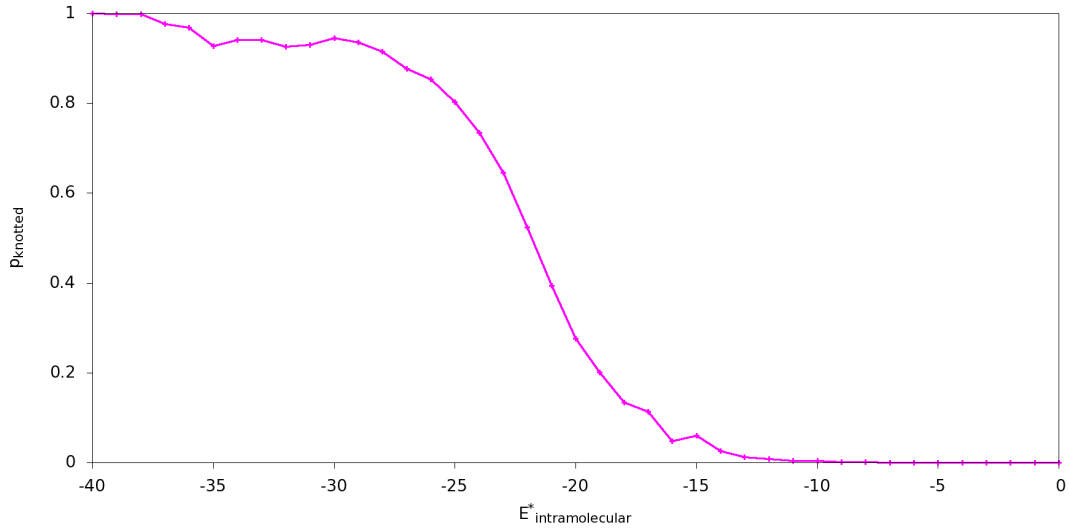

Figure 3.15: The cross-section at melting temperature $T_m^* = 0.656$ of the conditional probability of the conformation being knotted for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$.

3.2.8 Equilibration and simulation structure

Earlier work performed by Faísca and co-workers [17, 18] had shown that a total of 10^{10} Monte Carlo steps (mcs), with samples being drawn every 10^5 mcs, were adequate simulation parameters to study the thermal equilibrium distributions of the test lattice proteins being considered, when their folding occurred in bulk or under steric confinement. Our first concern was, thus, to test if these same parameters remained adequate when studying folding under hydrophobic confinement.

Once the simulation software had been extended with the wall interaction component, test runs were performed to observe its convergence to the thermal equilibrium distribution. The k31 lattice protein system with hydrophobic decoration ctc01 was used, confined in a box of side length $L = 33$ and relative strength of hydrophobic interaction with the wall $\eta_{HP} = 0.50$.

Three scenarios were tested for the number of mcs between Replica Exchange steps (REs): 10^5, 10^4 and 10^3. Equilibration occurred faster when 10^3 mcs separated REs and was slowest at 10^5 mcs separation.

However simulation time increased considerably when separation was reduced from 10^4 to 10^3. A separation of 10^4 mcs between REs was thus adopted as this offered the most adequate balance between simulation performance and simulation time.

Figures 3.16-18 present running energy histograms of the entire simulation at this REs separation, for replicas at $T^* = 0.300$ (Fig. 3.16), $T^* = 0.660$ (Fig. 3.17) and $T^* = 0.900$ (Fig. 3.18). In the

x-axis is the Monte Carlo step number in increments of 10^5. In the y-axis is the binned variable, the adimensional intramolecular energy. Each vertical cross-section is the distribution of the intramolecular energy of the conformations over the energy bins during the respective 10^5 mcs.

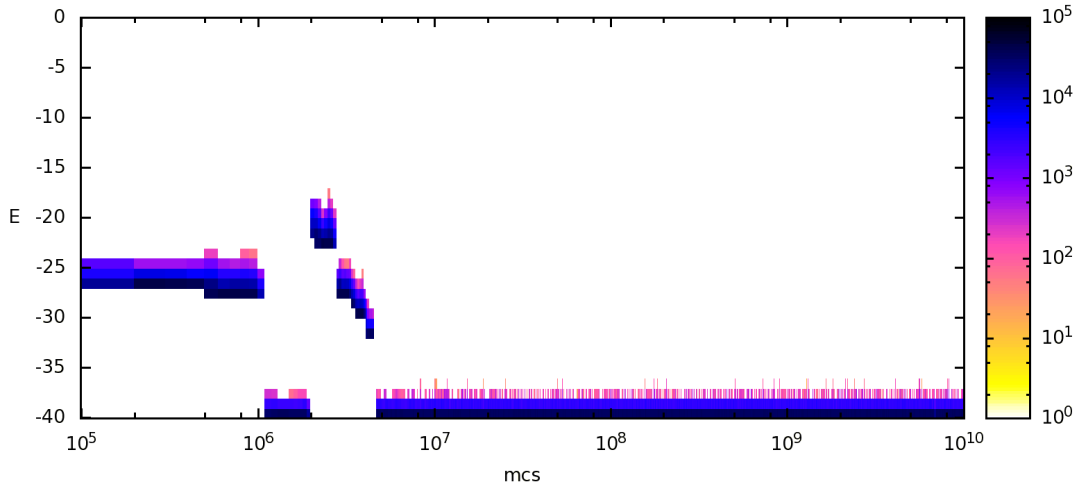

Figure 3.16: Running energy histogram for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$ at $T^* = 0.300$. Each vertical cross-section is the histogram of the 10^5 values of the intramolecular energy of all the conformations that occurred during the mcs that separate successive samples.

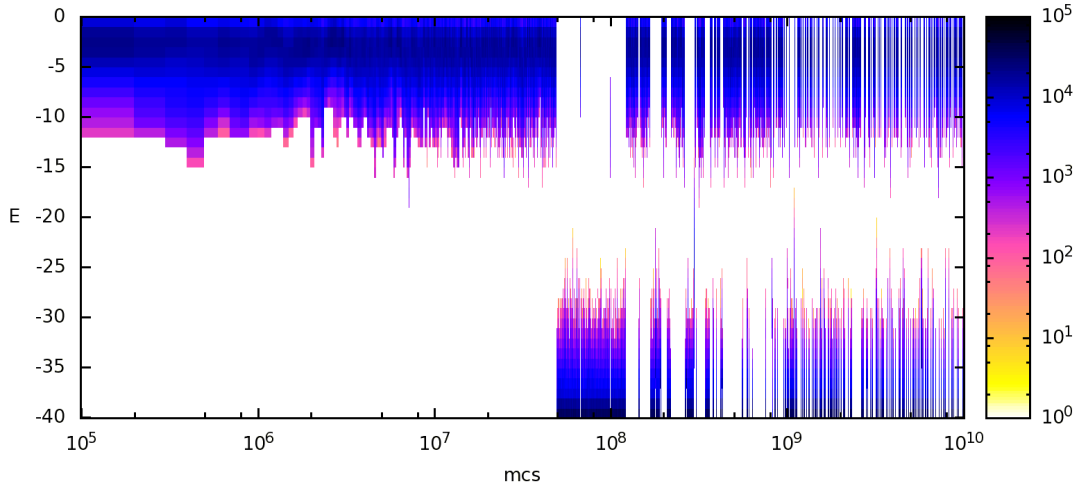

Figure 3.17: Running energy histogram for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$ at $T^* = 0.660$. Each vertical cross-section is the histogram of the 10^5 values of the intramolecular energy of all the conformations that occurred during the mcs that separate successive samples.

Equilibration clearly occurs sooner at $T^* = 0.300$ and $T^* = 0.900$, which are considerably below and above the melting temperature, than at $T^* = 0.660$ which is close to $T^*_m = 0.656$. In this most unfavorable case, equilibration is established after approximately 5×10^8 mcs.

Since we would be exploring large ranges of simulation parameters we decided, conservatively, to allocate one order of magnitude more mcs to initial equilibration than this observed requirement value.

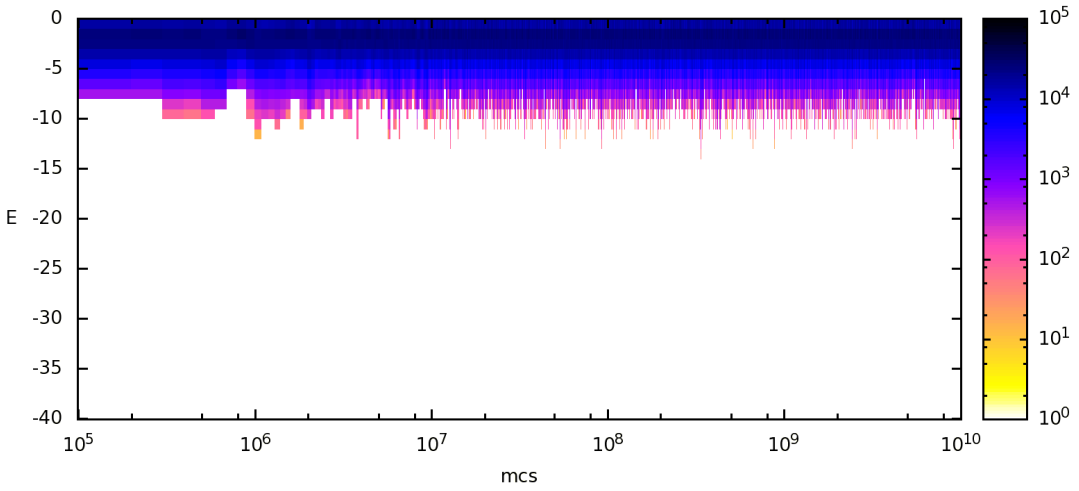

Figure 3.18: Running energy histogram for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$ at $T^* = 0.900$. Each vertical cross-section is the histogram of the 10^5 values of the intramolecular energy of all the conformations that occurred during the mcs that separate successive samples.

Hence, of the total 10^{10} simulation mcs, the initial 5×10^9 were adopted as equilibration steps, and during the remaining 5×10^9 steps, samples were taken for data analysis at an interval of 10^5 mcs, leading to a total of 5×10^4 samples per replica in each simulation run.

3.2.9 Autocorrelation

In the formulation presented in section 3.2.7, WHAM requires uncorrelated samples. Since, in all simulations, samples were drawn with a 10^5 mcs interval between them, it was important to test if this interval was sufficient to ensure low correlation between successive samples.

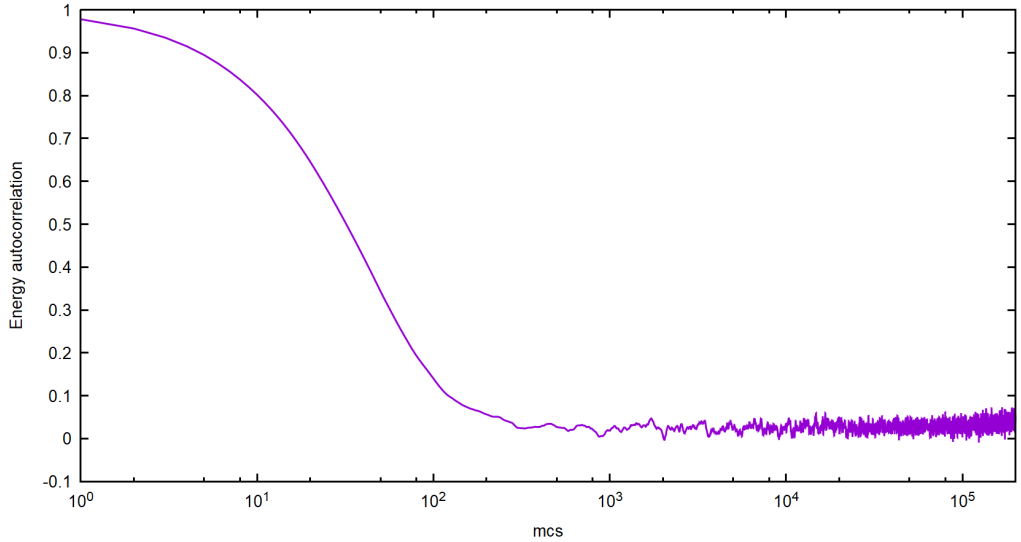

Figure 3.19: Autocorrelation of the total energy random variable for simulation k31, ctc01, $\eta_{HP} = 0.5$ and $L = 33$ at $T^* = 0.300$.

Figure 3.19 shows the Pearson's autocorrelation coefficient of the total energy random variable for the replica at $T^* = 0.300$ of the simulation of k31 lattice protein system with hydrophobic decoration ctc01, confined in a box of side length $L = 33$ and relative strength of hydrophobic interaction with the wall $\eta_{HP} = 0.50$.

Autocorrelation can be seen to be negligible at separations higher than 10^3 mcs, hence confirming that our choice of sampling interval was adequate.

3.2.10 Uncertainties

Many of the physical quantities of interest in this work, such as internal energy, specific heat or knottiness probability, are averages over the 50 000 uncorrelated sample values drawn from each replica for the respective quantities.

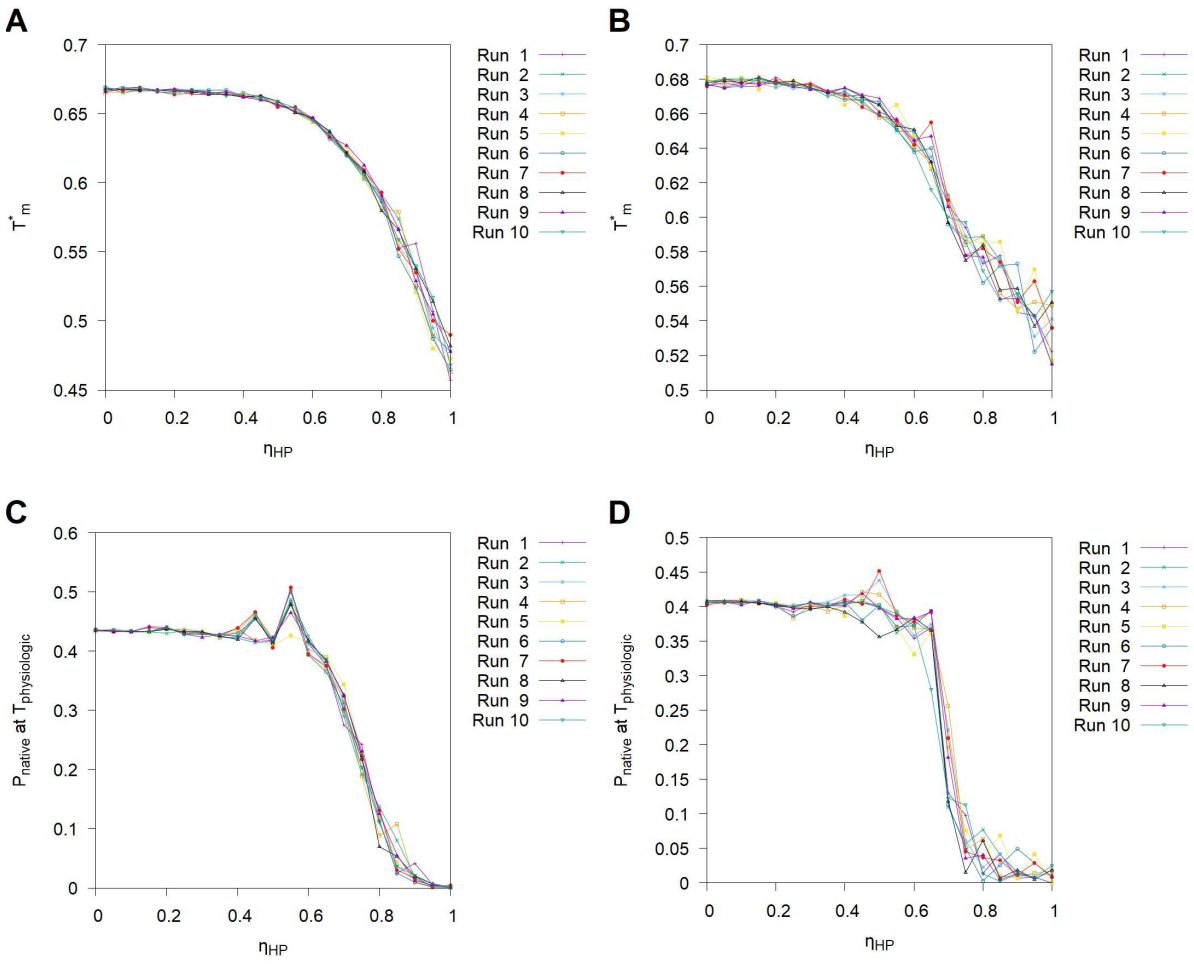

Figure 3.20: The melting temperature as a function of hydrophobic wall interaction strength for 10 sets of 21 simulations of k31 (A) and k52 (B) and the probability of having native conformation at physiologic temperature as a function of hydrophobic wall interaction strength for the same 10 sets of 21 simulations of k31 (C) and k52 (D), ctc01, $0 \leqslant \eta_{HP} \leqslant 1$ and $L = 33$.

Since the uncertainty of an average decreases in inverse proportion to the square root of the number of uncorrelated samples involved, in this study, these average quantities have negligible uncertainties and

no error bars are shown in their plots since these would be smaller than the width of the plotted line.

However, not all physical quantities of interest are averages. In fact, the two most important quantities discussed, melting temperature and probability of having native conformation at physiologic temperature, are not averages and hence require ensembles of repeated simulations to determine their uncertainties.

The present study, however, does not aim to make quantitative predictions to be statistically compared to observational values in hypothesis tests. The lattice test protein systems used are generic and do not intend to rigorously represent any specific natural proteins, to which they might be compared.

This study aims only at establishing clear trends in the physical quantities of interest that occur as the relevant simulation parameters vary.

To test if clear trends were obtained, a set of 42 simulations (21 involving k31 ctc01 within hydrophobic box of $L = 33$ for 21 wall interaction strengths between 0 and 1 and another 21 using k52 under identical conditions) were repeated 10 times. Melting temperature and probability of having native conformation at physiologic temperature, are shown in Figures 3.20 A-D.

Even though some dispersion in the simulation results can be seen, particularly at higher wall interaction strengths, the trend remains clear and consistent throughout all simulations. For this reason, the results presented in the next chapter involve only one simulation for each specific system and condition, and no error bars are shown.

3.2.11 Simulation software architecture

The software architecture for this study is depicted as a dataflow diagram in Fig. 3.21. Input data files are colored blue, executable programs are colored green, output data files are colored black and final graphical output plots are colored red.

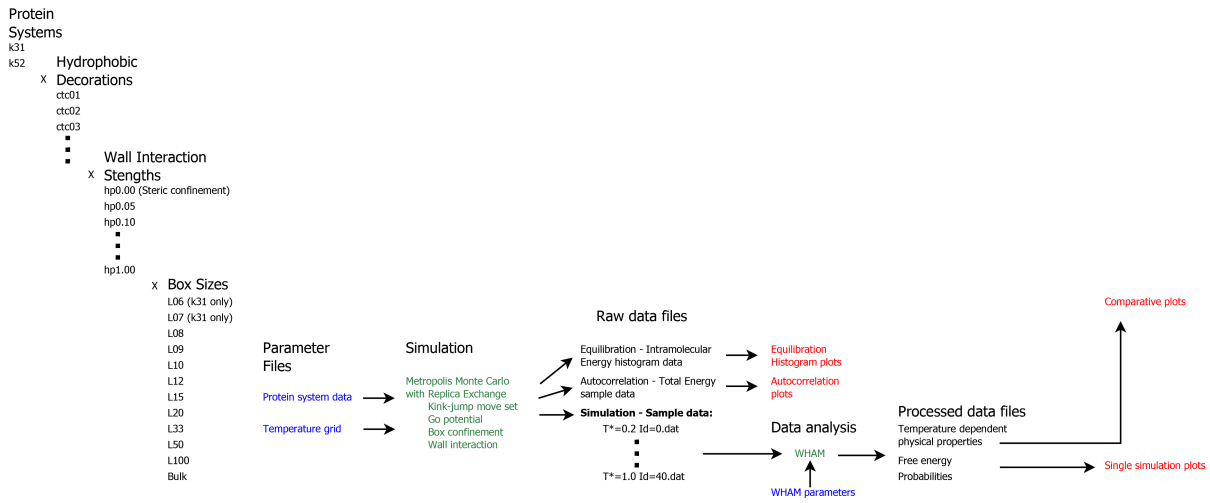

Figure 3.21: Dataflow diagram for the present protein folding simulation study.

The simulation runs are organized into a hierarchy, each run involving the concurrent simulation of 41 temperatures, enumerated in the temperature grid parameter file. At simulation run level each protein system is described in a data file that details its initial conformation, its native conformation and the particular hydrophobic decoration adopted for that run. Folding in Bulk conditions is implemented as a

box of size $L = 33$ with conditions equivalent to periodic boundary conditions.

The Monte Carlo simulation software implements parallel processing through usage of the Message Passing Interface (MPI) software stack and simulates the 41 temperature values concurrently in processes with MPI rank from 0 to 40.

Every 10^5 mcs an energy histogram slice is produced for later analysis of the equilibration process. Starting at mcs 7×10^9 and for 5×10^5 mcs all values of the total energy at all temperatures and for all conformations generated are stored in files for later autocorrelation analysis. From these files graphical plots, such as those in Figures 3.16-19, are produced for simulation quality assessment.

The 41 single temperature simulation output files are then post-processed by the implementation of the WHAM data analysis method described in section 3.2.7. From its output data files the 23 plots exemplified in the same section are automatically produced for every simulation through use of gnuplot scripts, also for simulation quality assessment.

The processed data was finally combined into comparative plots, presented in the next chapter.

Using this simulation infrastructure we investigated the effects of hydrophobic intermolecular interactions of different intensities in the folding process of both the knotted trefoil, and knotted twisted-three lattice proteins for cages with different sizes, mimicking the confinement conditions occurring during the chaperonin cycle.

Chapter 4

Results

The results of all simulations regarding the model protein with embedded trefoil knot (3_1), k31, are entirely consistent with the corresponding results obtained for the model protein with embedded twisted-three knot (5_2), k52. Hence we will discuss the results pertaining to both model proteins jointly, all figures having two columns of plots, the left column always presenting results for k31 and the right column the corresponding results for k52, as we already did in Fig. 3.20.

4.1 Folding in bulk conditions

The baseline scenario for the present study is the folding behavior of the model proteins in bulk conditions. Hence, we begin by presenting the simulation results for both model proteins in these conditions.

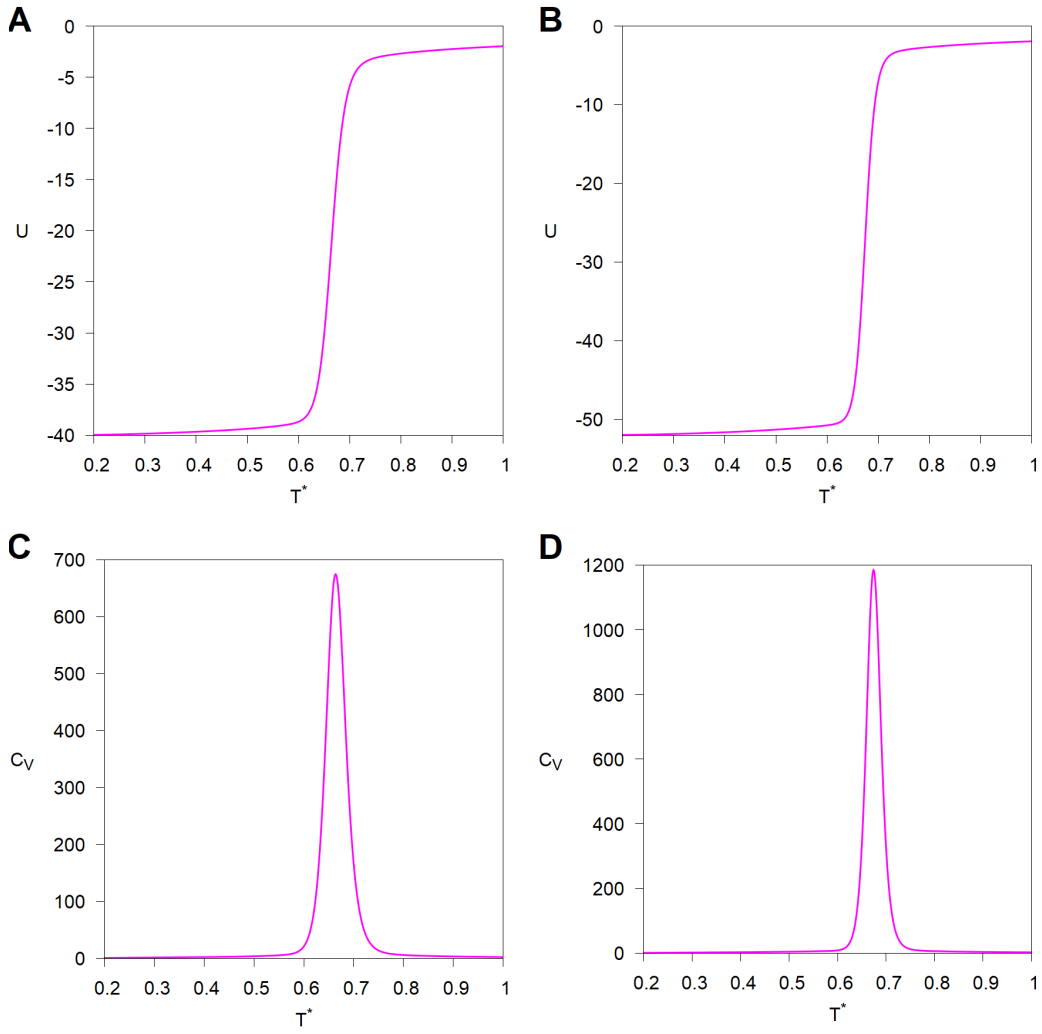

Figure 4.1: Internal energy and specific heat for the folding of k31 (left column) and k52 (right column) in bulk conditions. Specific heat peaks occur at $T_m^* = 0.664$ for k31 (C) and at $T_m^* = 0.674$ for k52 (D).

Figure 4.1 shows internal energy and specific heat as functions of simulation temperature. Melting can be seen to occur at $T_m^* = 0.664$ for k31 and at $T_m^* = 0.674$ for k52, the temperature at which specific heat peaks. When folding in bulk conditions, both model proteins display a sharp transition at T_m from unfolded to native.

Figure 4.2 shows the cross-section of the Helmholtz free energy relative to the native state at transition temperature, T_f, the temperature at which the most probable unfoded state has the same probability as the native state, and the cross-section of the conditional probability of a knotted conformation at melting temperature, both as functions of the reaction coordinate Q, which, let us recall, is the ratio of the number of native contacts that are formed to the total number of native contacts. Both phases coexist with equal probability at T_f.

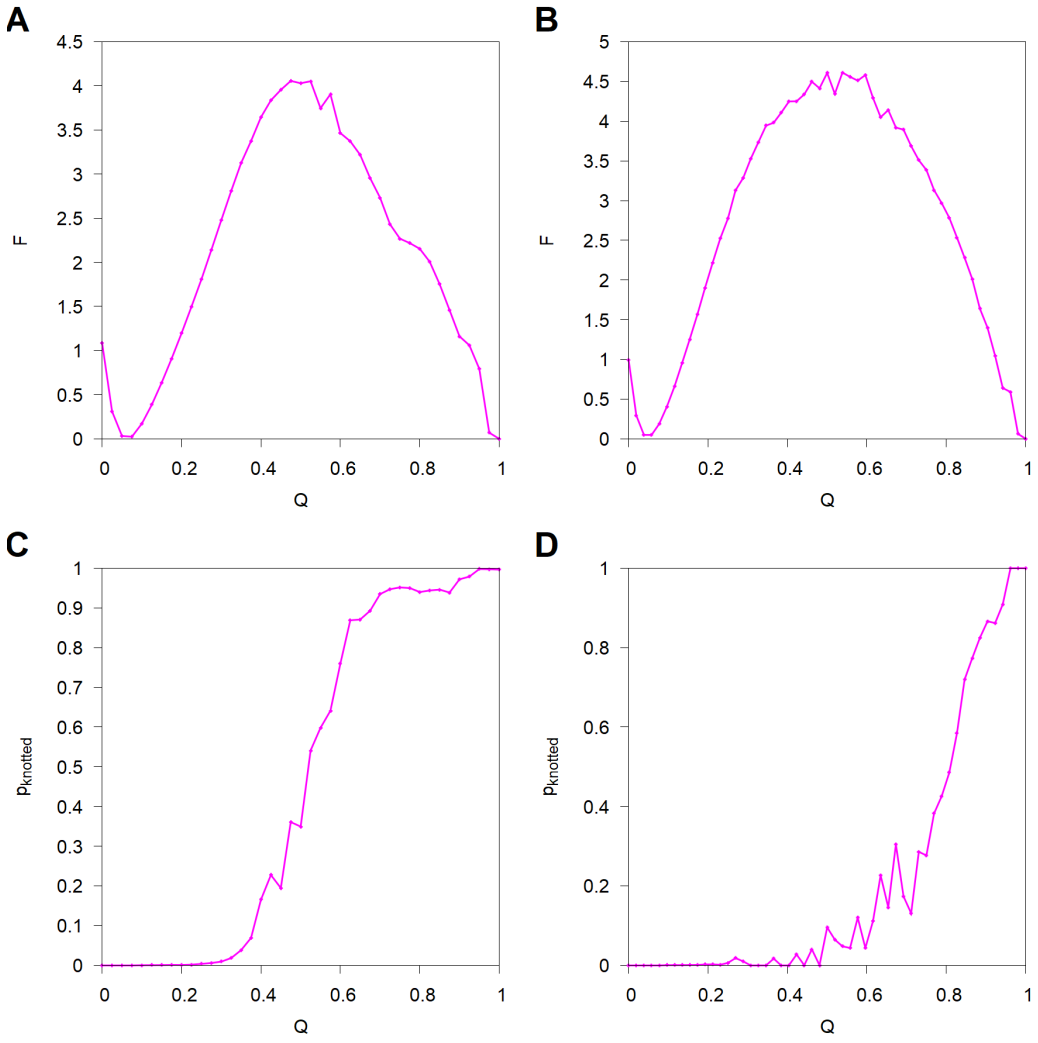

Figure 4.2: The cross-section of the Helmholtz free energy relative to the native state at transition temperature $T_f^* = 0.672$ for k31 (A) and at transition temperature $T_f^* = 0.679$ for k52 (B) and the cross-section of the conditional probability of the conformation being knotted at melting temperature $T_m^* = 0.664$ for k31 (C) and at $T_m^* = 0.674$ for k52 (D) in bulk conditions.

The projection of the free energy on Q at T_f shows a two-state transition, with the unfolded basin separated from the native one by a free energy barrier that corresponds to the transition state. The free energy projection peaks at $Q = 0.500$ for k31 and at $Q = 0.538$ for k52, the ensemble of conformations of each model protein having this value of Q being its Transition State Ensemble (TSE). Also, since the transition is two-state T_m is nearly identical to T_f ($0.664 \approx 0.672$ for k31 and $0.674 \approx 0.679$ for k52).

From Fig. 4.2 C and D we can see that k52 requires a higher percentage of formed native contacts than k31 to achieve the same knotting probability. This is likely to be due to the higher structural complexity of the twisted-three knot when compared to the trefoil knot.

4.2 Folding under steric confinement

The effects of steric (i.e. physical) confinement on the folding transition of the two considered model proteins are presented next.

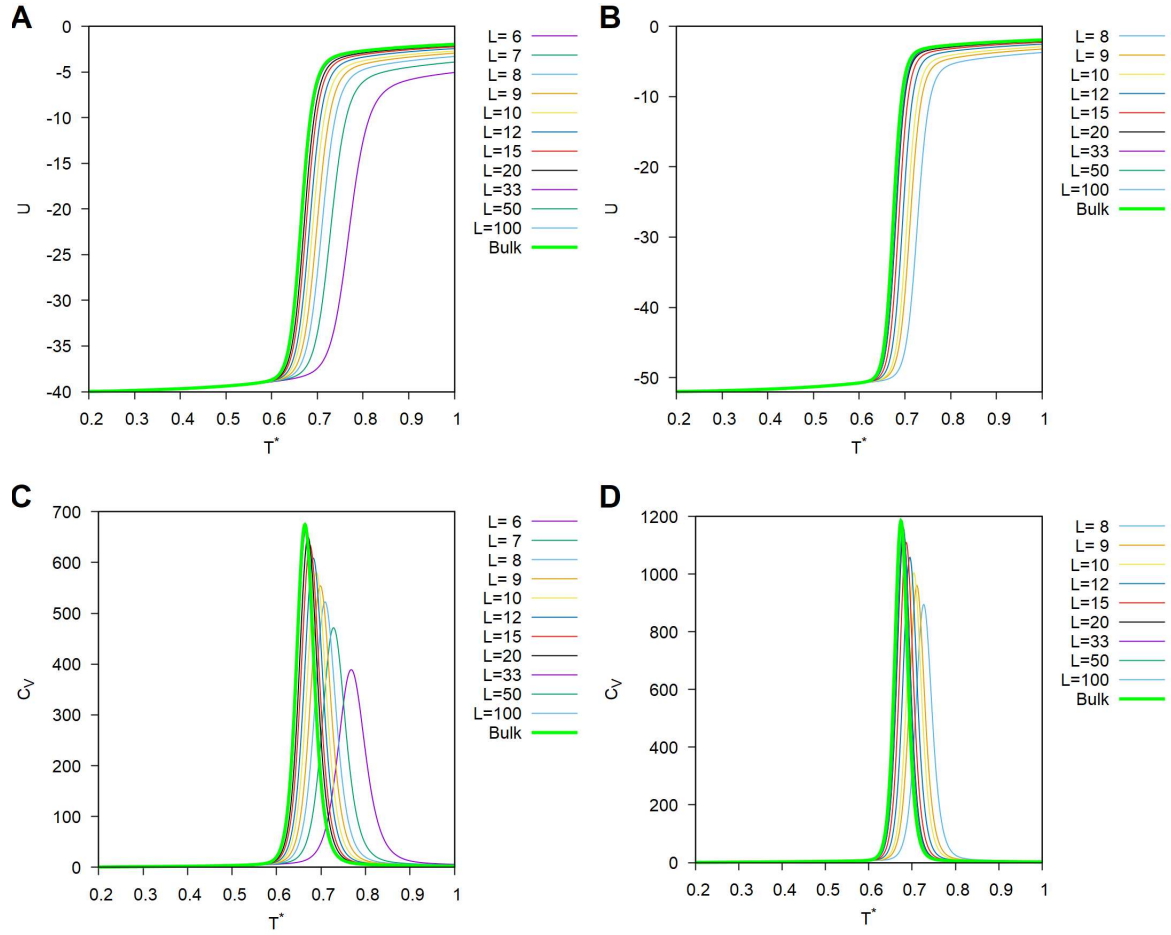

Figure 4.3: Internal energy and specific heat for the folding of k31 (left column) and k52 (right column) under steric confinement conditions.

From Fig. 4.3 it can be seen that the melting temperature drifts towards higher temperatures as the size of the constraining box, L, is decreased. The steepness of the internal energy curves also decreases. This is due to the unfolded population getting structurally consolidated (lowering the unfolded plateau in the internal energy curves, Fig. 4.3 A and B) and, as may be seen in Fig. 4.4 A and B, the distribution over Q of the two populations of unfolded and native conformations still being bimodal but having free energy peaks at T_f that come closer together as box size, L, decreases.

As box size decreases, the height of the peak of specific heat also becomes steadily smaller. This behavior was expected and results from smaller energy fluctuations being observed under confinement, since this reduces conformational entropy by prohibiting the existence of spatially extended conformations (i.e. unfolded conformations for which $U \approx 0$, as again may be seen from the decreasing energy of the unfolded plateau in Fig. 4.3 A and B).

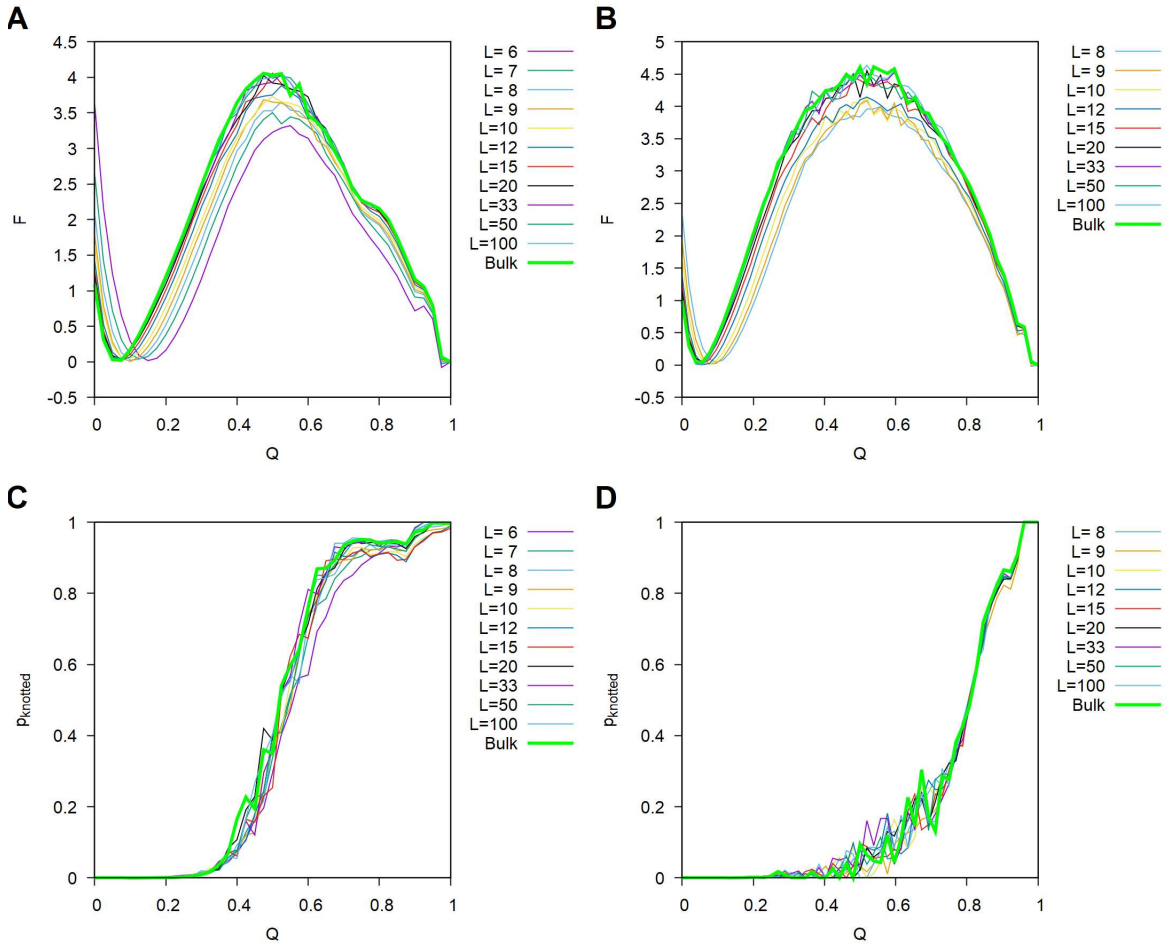

Figure 4.4: The cross-section of the Helmholtz free energy relative to the native state at transition temperature and the cross-section of the conditional probability of the conformation being knotted at melting temperature for the folding of k31 (left column) and k52 (right column) under steric confinement conditions.

Figure 4.4 A and B also shows that the two-state nature of the folding transition is preserved across all box sizes and that a progressive shift of the unfolded state free energy minimum towards higher Q occurs as the size of the confining box is reduced, indicating that the unfolded state gains residual native structure and becomes more compact and stabilized under steric confinement.

Figure 4.4 C shows a decrease in knotting probability for k31 within box $L = 6$. A previous study [17, 18] has identified the origin of this decrease. It is due to the box being too small and the system forming malformed knots. This also impacts on the folding rate which was found not to be optimal at $L = 6$ contrary to what one might expect from the free energy profile, that shows a more stabilized TSE. Knotting and folding are thus clearly separated processes, with knotting occurring in more consolidated conformations and delaying folding because of malformed knotted conformations which are difficult to unknot.

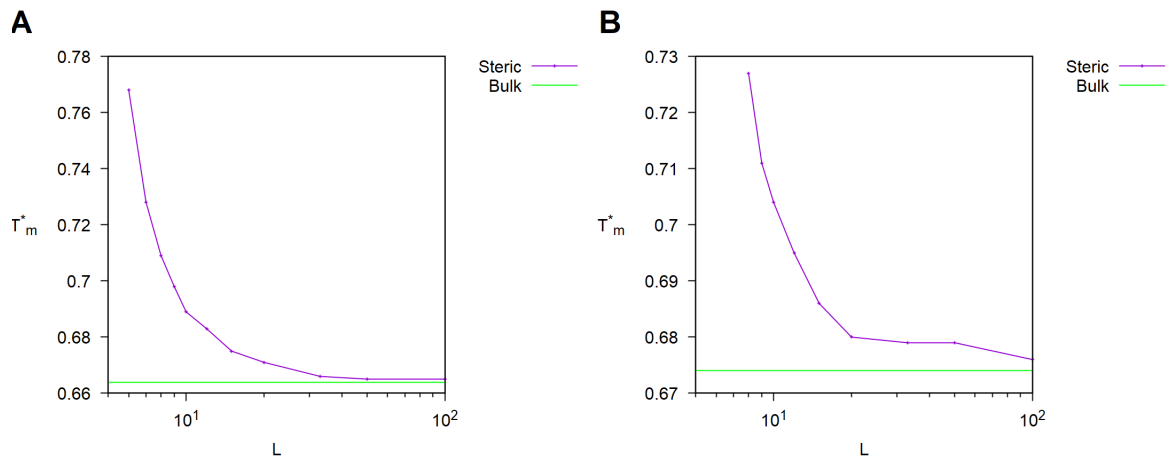

Figure 4.5: Melting temperature as function of the confining box size for the folding of k31 (A) and k52 (B) under steric confinement conditions (note that box size scale is logarithmic).

We can, hence, summarize the main effect of steric confinement on both model proteins, the raising of melting temperature, through Figure 4.5.

From the point of view of chaperonin function, the raising of the melting temperature by steric confinement may appear puzzling. If the chaperonin were to act upon a folding protein solely in this way it would stabilize any misfolded conformation that might enter its cavity and this would certainly decrease the probability of the protein acquiring its native state at the end of the chaperonin' cycle, i.e. reduce the effectiveness of its cycle. Consequently, steric confinement cannot be the sole role of the chaperonin. We know, of course, that at the start of its cycle the chapeonins' inner walls are hydrophobic and thus that confinement is not simply steric at that stage of the cycle. In the next section we present the effects of hydrophobic confinement upon the two model proteins.

4.3 Folding under hydrophobic confinement

The addition of an hydrophobic interaction between the model protein and the walls of the confining box considerably changes the folding behavior of both model proteins.

4.3.1 Folding k31 ctc01 and k52 ctc01 under hydrophobic confinement

In section 3.1.3 we suggested that the base hydrophobic decoration scenario, for each model protein, was the one in which the residues with highest number of native contacts (the contact core) are all made equally hydrophobic and all other residues are considered neutral. We named this hydrophobic decoration scenario ctc01.

Let's thus begin discussing the effects of hydrophobic confinement by presenting the results for both model proteins with decoration ctc01.

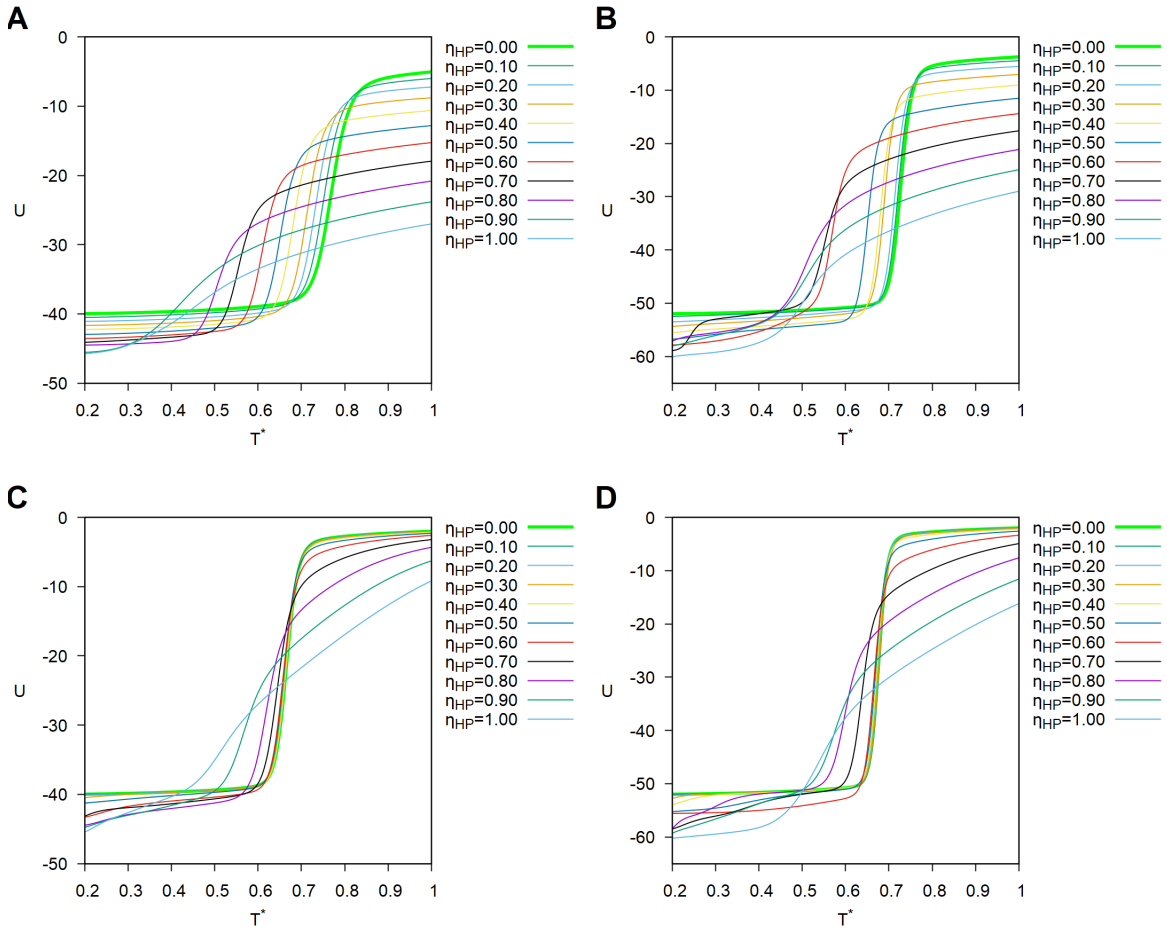

Figure 4.6: Internal energy for the folding under hydrophobic confinement of k31 ctc01 in a box of size $L = 6$ (A) and $L = 100$ (C) and for the folding of k52 ctc01 in a box of size $L = 8$ (B) and $L = 100$ (D) (note that $\eta_{HP} = 0$ is steric confinement).

Figure 4.6 shows the internal energy as a function of temperature for interaction strengths varying in the interval $0 \leqslant \eta_{HP} \leqslant 1$ and confining box sizes ranging from $L = 6$ for k31 in Fig. 4.6 A and $L = 8$ for k52 in Fig. 4.6 B to $L = 100$ for both in Figures 4.6 C and D.

From this figure it becomes clear that the temperature of the folding transition steadily decreases as the intensity of the hydrophobic interaction increases and that this decrease is more pronounced, and begins to occur at weaker interaction strengths, for smaller confining boxes. Larger boxes require stronger interactions before the typical steric confinement behavior begins to be affected.

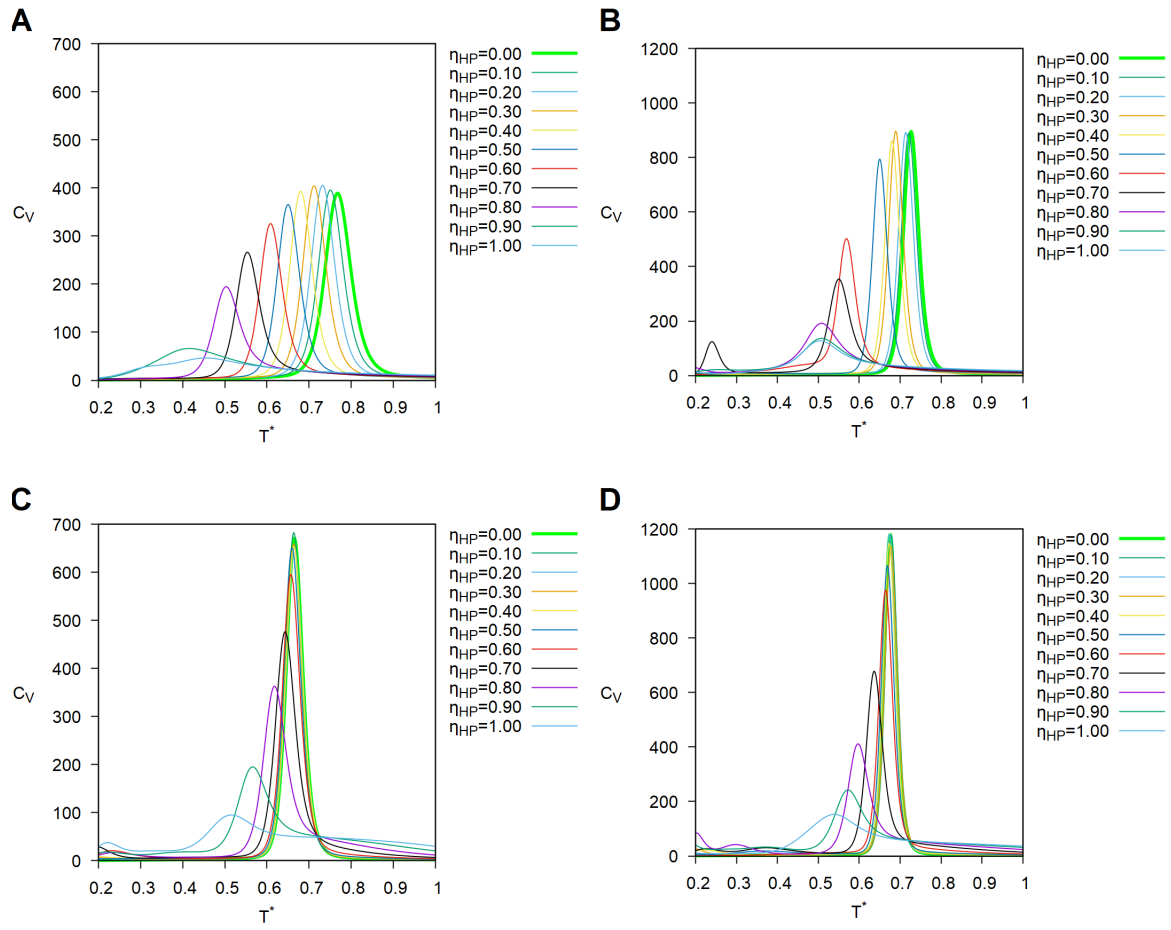

Figure 4.7: Specific heat for the folding under hydrophobic confinement of k31 ctc01 in a box of size $L = 6$ (A) and $L = 100$ (C) and for the folding of k52 ctc01 in a box of size $L = 8$ (B) and $L = 100$ (D) (note that $\eta_{HP} = 0$ is steric confinement).

The same effect can be seen in Fig. 4.7 which displays the behavior of specific heat.

It is interesting to note the appearance of small secondary peaks in the specific heat for small boxes at high interaction strengths and low temperatures. These signal the onset of intermediate states in the folding/unfolding transition process.

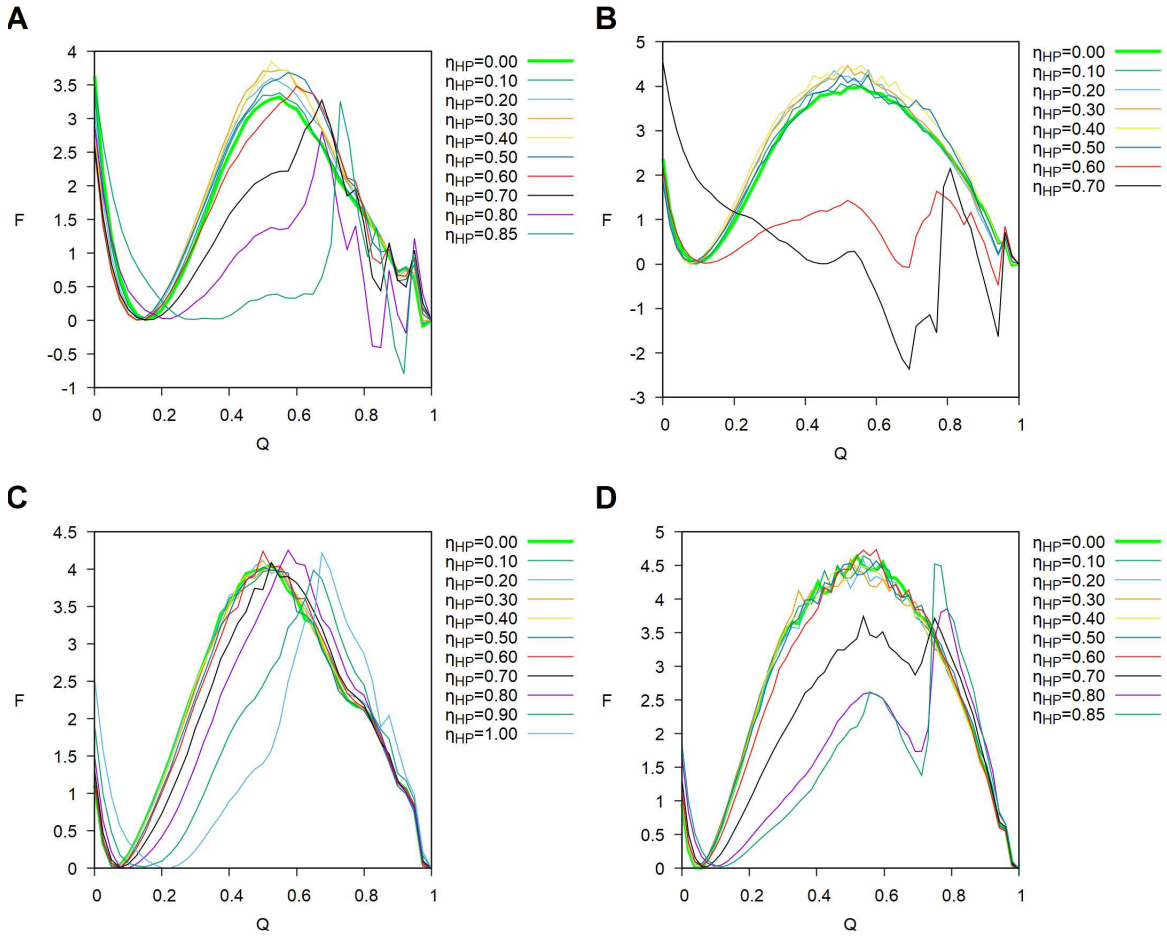

Figure 4.8: The cross-section of the Helmholtz free energy relative to the native state at transition temperature for the folding under hydrophobic confinement of k31 ctc01 in a box of size $L = 6$ (A) and $L = 100$ (C) and for the folding of k52 ctc01 in a box of size $L = 8$ (B) and $L = 100$ (D) (note that $\eta_{HP} = 0$ is steric confinement).

Figure 4.8 reveals a clear breakdown of the two-state transition particularly for smaller boxes and stronger wall interaction strengths.

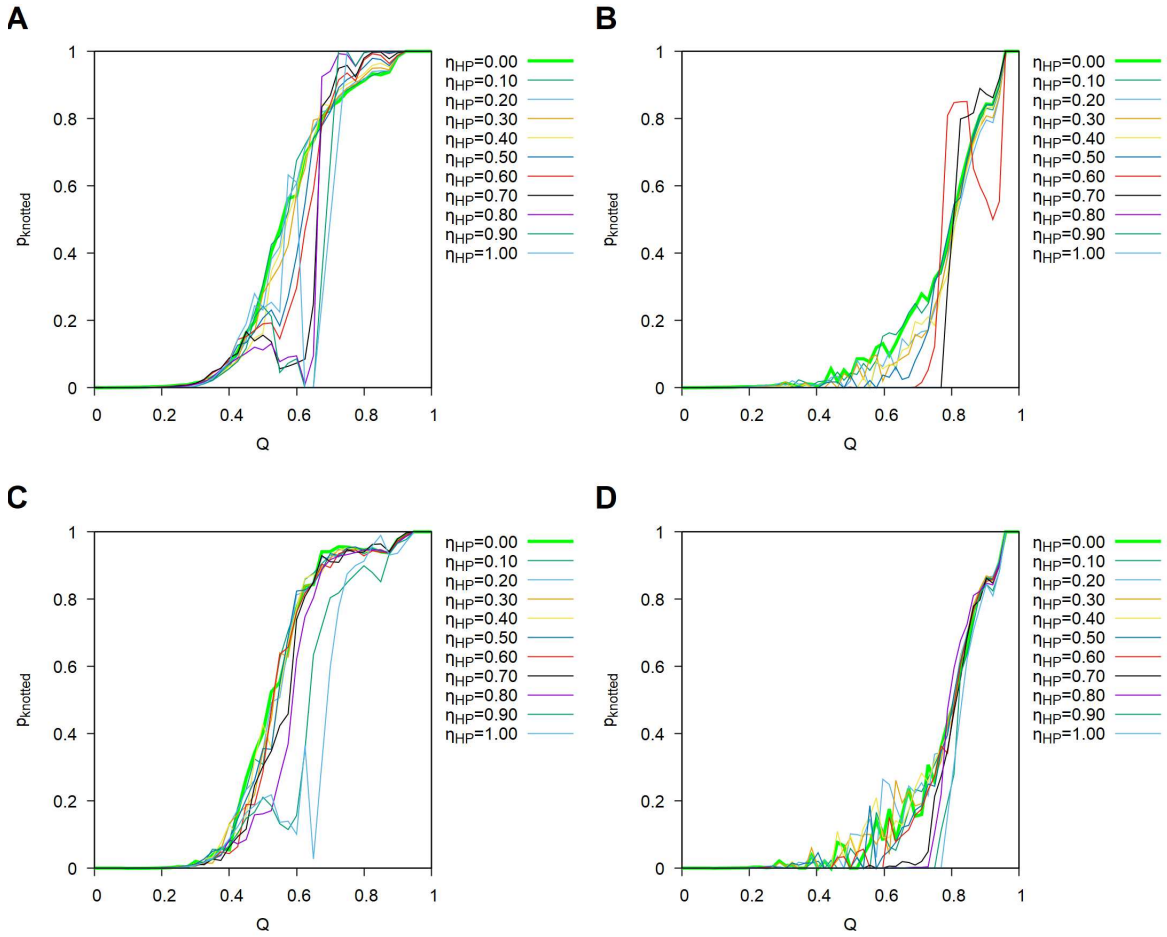

Figure 4.9: Cross-section of the conditional probability of the conformation being knotted at melting temperature for the folding under hydrophobic confinement of k31 ctc01 in a box of size $L = 6$ (A) and $L = 100$ (C) and for the folding of k52 ctc01 in a box of size $L = 8$ (B) and $L = 100$ (D) (note that $\eta_{HP} = 0$ is steric confinement).

The knotting probability, Fig. 4.9, is also profoundly disrupted by the hydrophobic interaction with the box's wall, a higher percentage of native contacts being required to achieve the same knotting probability at higher interaction strengths.

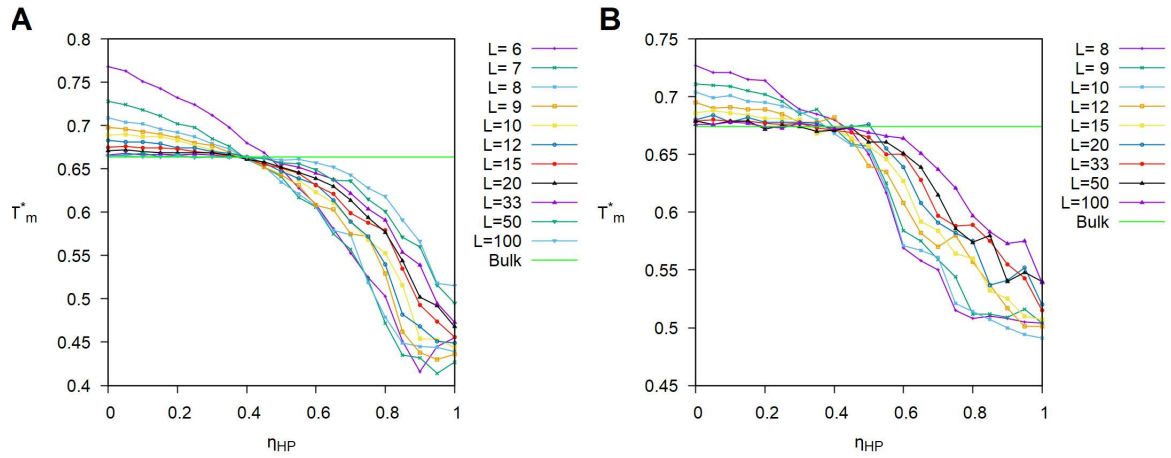

Figure 4.10: The melting temperature as a function of hydrophobic wall interaction strength for the folding under hydrophobic confinement of k31 ctc01 (A) and k52 ctc01 (B) (note that $\eta_{HP} = 0$ is steric confinement).

The effect of hydrophobic confinement upon the melting temperature of the model proteins is summarized in Figure 4.10.

It is important to note that while melting temperature remains higher than physiologic temperature ($T^*_{physiologic} = 0.6$, see section 3.2.6) in bulk and steric confinement conditions, under hydrophobic confinement of strength comparable to native contact strength the melting temperature drops to below physiologic temperature regardless of box size.

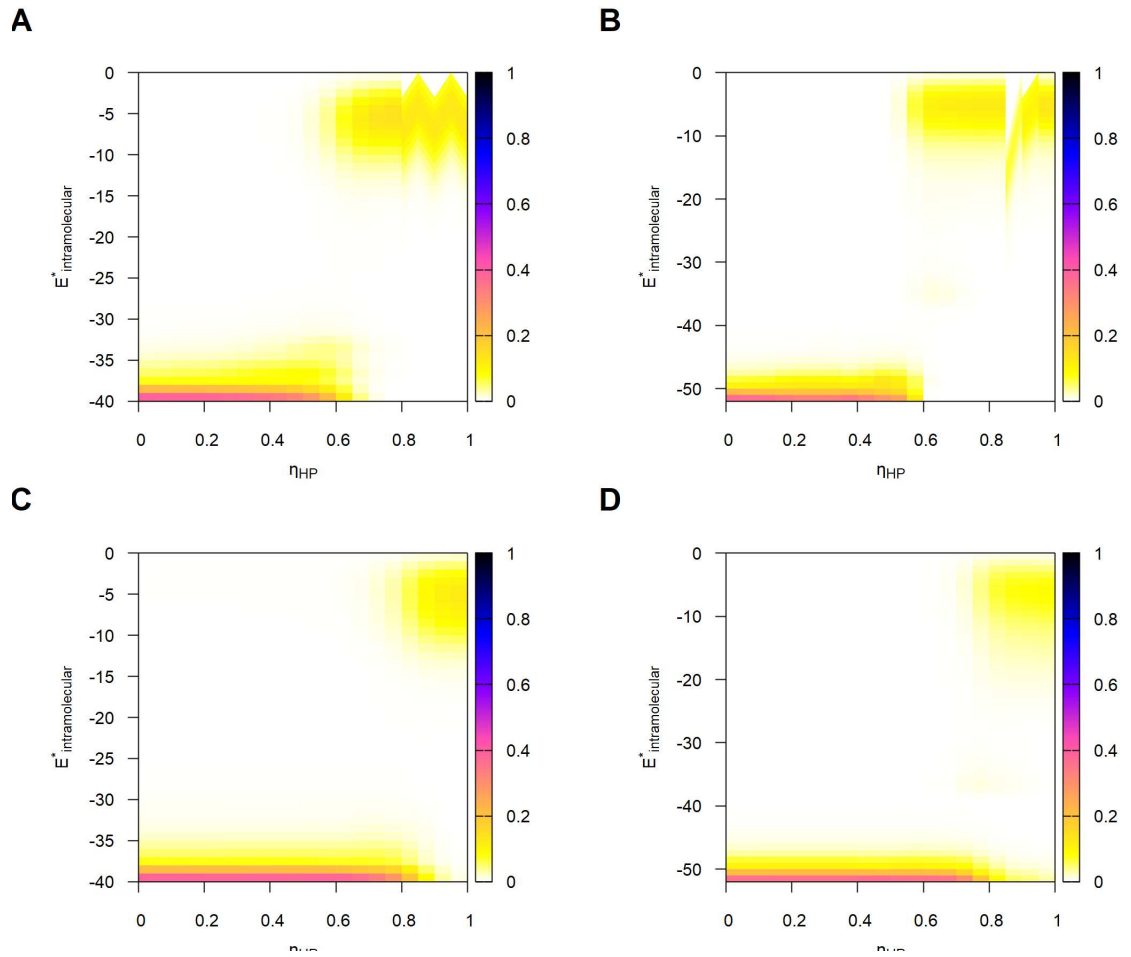

Figure 4.11: The probability distribution of intramolecular energy at physiologic temperature as a function of hydrophobic wall interaction strength for the folding under hydrophobic confinement of k31 ctc01 in a box of size $L = 6$ (A) and $L = 100$ (C) and for the folding of k52 ctc01 in a box of size $L = 8$ (B) and $L = 100$ (D) (note that $\eta_{HP} = 0$ is steric confinement).

This change in melting temperature considerably changes the probability distribution of intramolecular energy at physiologic temperature as Fig. 4.11 shows. These plots gather the slices at $T^* = 0.6$ of the relevant intramolecular energy *probablograms* (see Fig 3.13).

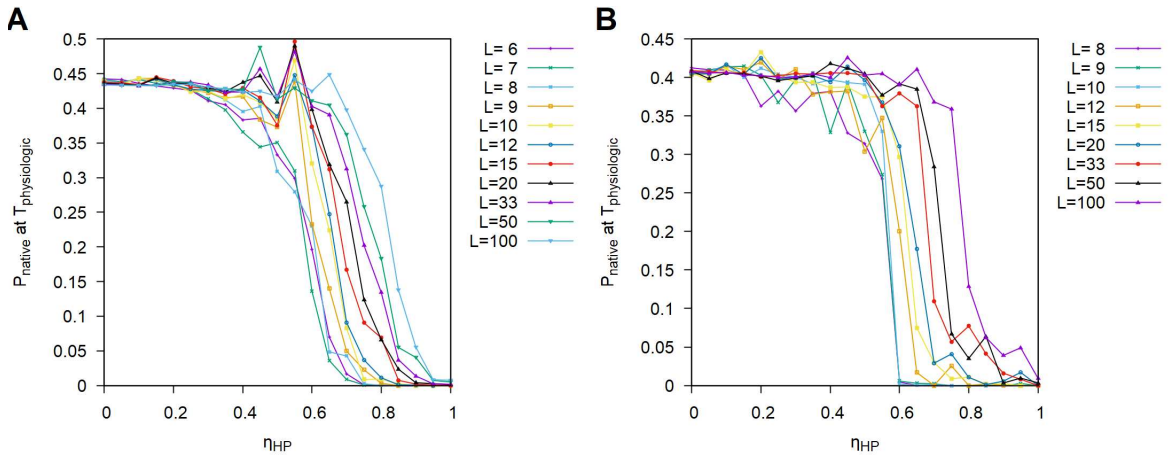

Figure 4.12: The probability of having native conformation at physiologic temperature as a function of hydrophobic wall interaction strength for the folding under hydrophobic confinement of k31 ctc01 (A) and k52 ctc01 (B) (note that $\eta_{HP} = 0$ is steric confinement).

The effect of hydrophobic confinement upon the probability of model proteins k31 ctc01 and k52 ctc01 being in the native state is finally summarized in Figure 4.12.

If the 6 second duration of the chaperonin's cycle is sufficiently long for protein macrostate change to be sufficiently slow for the protein molecule to always be very close to thermal equilibrium throughout the entire cycle (i.e. if the chaperone assisted protein folding process is quasi-static) then the system's thermodynamic properties will, throughout the entire cycle, always be very close to those of thermal equilibrium that we have been describing. The probability distribution of intramolecular energy at physiologic temperature will thus evolve throughout the cycle according to the evolution of the strength of the hydrophobic interaction with the chaperonin's inner walls.

Since the chaperonin's cycle begins with its inner walls hydrophobic and ends with them hydrophilic, this final condition being coarse grained in our model as neutral i.e. $\eta_{HP} = 0$, the chaperonin's cycle is, in our model, represented by a sweep of η_{HP} from a value close to 1 to a value close to 0. Regardless of box size we thus see that at the early part of the cycle both model proteins become unfolded, as its melting temperature is tuned to a value below physiologic temperature, and that, as the cycle progresses, and melting temperature progressively increases, the probability of the protein system acquiring the native state also progressively increases. By first unfolding the protein system, the chaperone is eliminating any trace of its initial conformation and hence of any misfolded features it might initially have had and that might have hindered folding into the native conformation, as assumed early in chapter 2.

From Figure 4.12 we thus find that at the end of the chaperonin's cycle both model proteins have a probability of being in the native state of approximately 40%. Since this value is higher than the 20% (see end of chapter 2) required to explain the fast folding rates observed for also trefoil knotted proteins YibK and YbeA, we conclude that hydrophobic confinement effects may be a possible explanation for the observed folding rate increase.

By adjusting the melting temperature of the protein system, the chaperone effectively provides an annealing heat treatment at constant temperature in each cycle. The folding process thus assisted can be qualified as an iterative annealing process as has been suggested [33, 34, 35] (see section 1.3).

4.3.2 Hydrophobic decorations of k31 and k52 under hydrophobic confinement

Thus far we have assumed that the protein residues with higher number of native contacts were hydrophobic (i.e. decoration ctc01). Nature is bound to be more diverse. Hence we explored other hydrophobic decorations in which the average number of native contacts of the hydrophobic residues is progressively lowered (ctc02 to ctc14 for k31 and ctc02 to ctc17 for k52, see section 3.1.3).

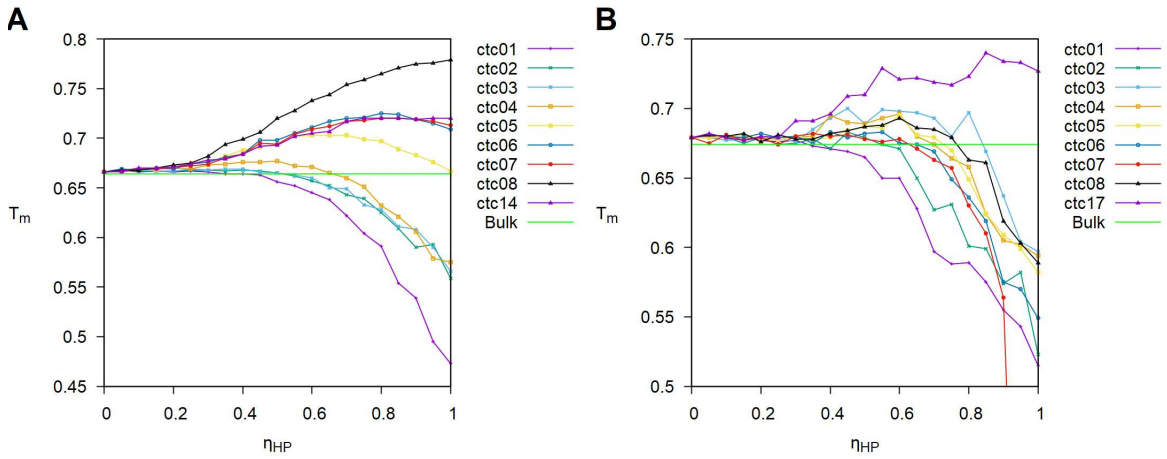

Figure 4.13: The melting temperature as a function of hydrophobic wall interaction strength for the folding under hydrophobic confinement in a box of size $L = 33$ of several hydrophobic decorations of k31 (A) and of k52 (B) (note that $\eta_{HP} = 0$ is steric confinement).

Figure 4.13 shows that, as the average number of native contacts of the hydrophobic residues is progressively lowered, the chaperonin eventually loses its ability to lower the melting temperature of the protein system, in fact, sometimes increasing it above its value under steric confinement.

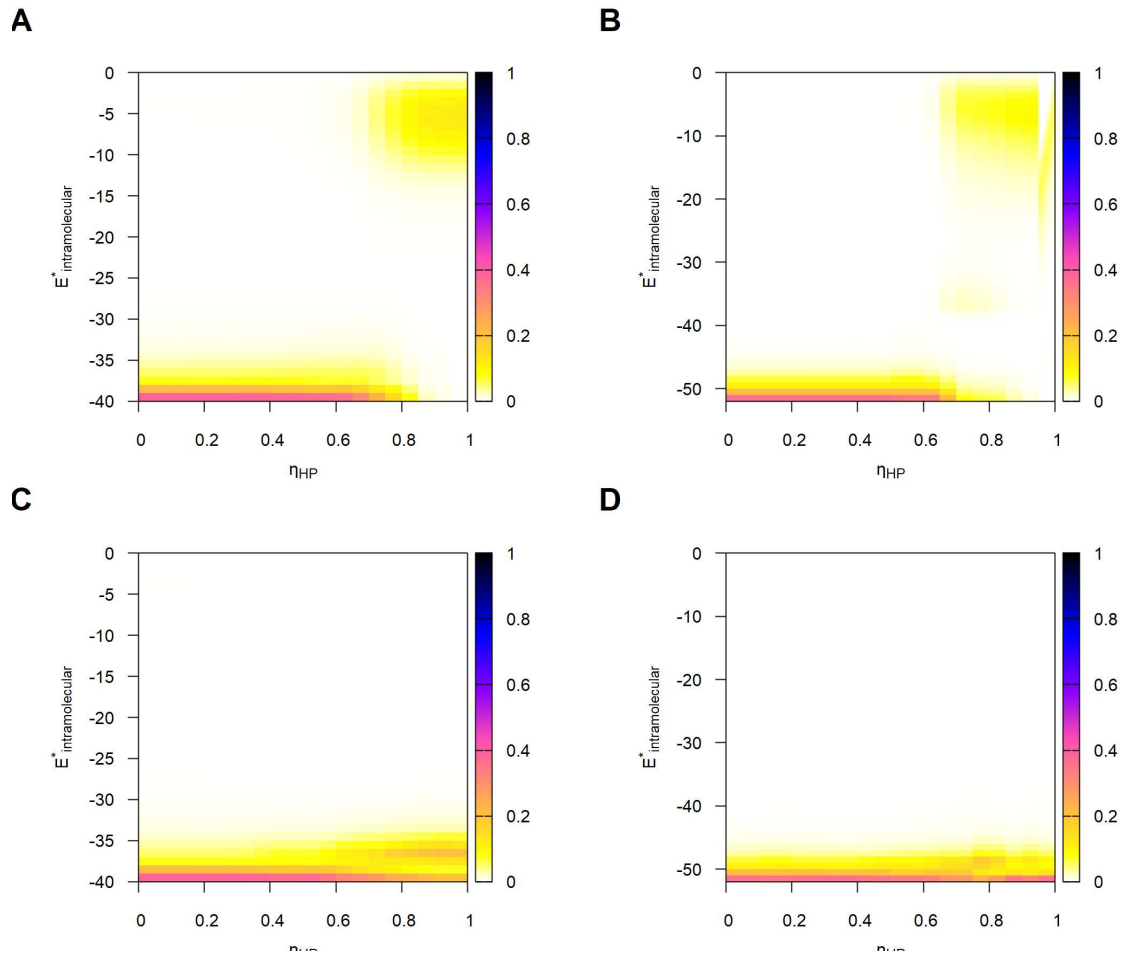

Figure 4.14: The probability distribution of intramolecular energy at physiologic temperature as a function of hydrophobic wall interaction strength for the folding under hydrophobic confinement in a box of size $L = 33$ of k31 ctc01 (A), k52 ctc01 (B), k31 ctc14 (C) and k52 ctc17 (D) (note that $\eta_{HP} = 0$ is steric confinement).

This, in turn, means that the probability distribution of intramolecular energy at physiologic temperature ceases to reach the unfolded ensemble as visible in Fig. 4.14. The chaperonin thus loses its ability to unfold the protein at the beginning of its cycle which means that proteins traped in misfolded conformations may become less effectively refolded. The chaperonin, however, continues to provide protection from crowding and aggregation and thus becomes a passive Anfinsen's cage (see section 1.3).

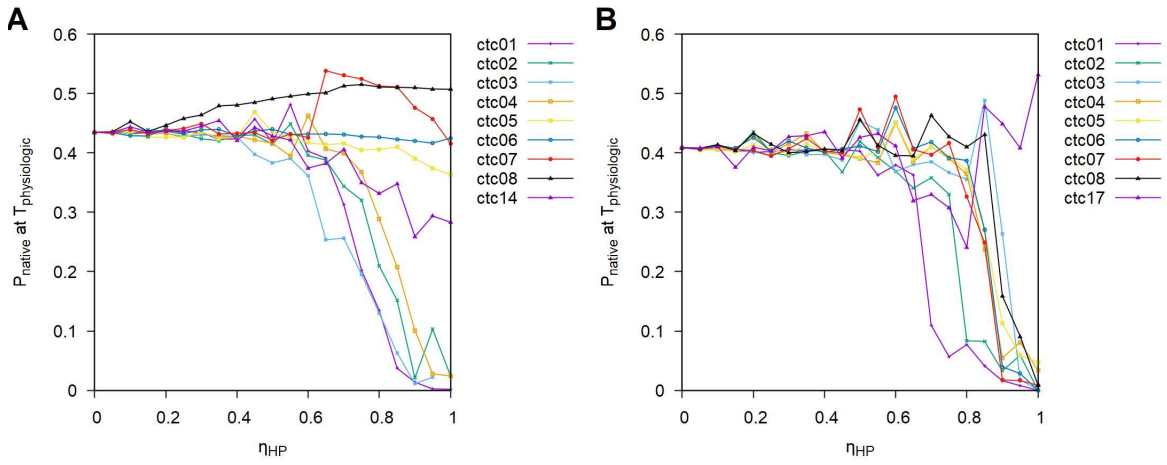

Figure 4.15: The probability of having native conformation at physiologic temperature as a function of hydrophobic wall interaction strength for the folding under hydrophobic confinement in a box of size $L = 33$ of several hydrophobic decorations of k31 (A) and k52 (B) (note that $\eta_{HP} = 0$ is steric confinement).

Figure 4.15 shows that the probability of the protein having its native conformation at physiologic temperature at the end of the chaperonin cycle remains identical to the probability determined for ctc01 decoration (approximately 40%), provided the molecule can achieve thermal equilibrium (i.e. does not become traped in a misfolded conformation, which now, due to the higher than physiologic temperature melting temperature, can no longer be undone).

4.4 Conclusions

The present work investigated the role that molecular chaperones may have on the folding process of knotted proteins. Modeling the GroEL-GroES chaperonin complex's cycle as a confining box that progressively changes the hydropathic nature of its inner walls from hydrophobic to hydrophilic we simulated the effects of such confinement on two knotted lattice proteins, one embedding a trefoil (3_1) knot, the other a three-twist (5_2) knot.

The aim of the work was to determine if these confinement effects might explain the folding rates observed for proteins YibK and YbeA when their folding process is assisted by the GroEL-GroES chaperonin complex. Through the kinetic theory for chaperone assisted protein folding presented in chapter 2 we transformed the problem from explaining folding rates to explaining the probability of the protein having its native conformation at the end of the chaperonin cycle, concluding at the end of chapter 2 that a higher than 20% probability would be able to explain the observed rates.

Assuming the chaperonin cycle to be a quasi-static process, we approached the estimation of this probability through simulation of thermal equilibrium states using the Metropolis Monte Carlo enhanced with Replica Exchange method with trial conformations generated using the kink-jump move set. Conformation knottiness was determined using the Koniaris-Muthukumar-Taylor method. Analysis of the simulation generated data was performed using the Weighted Histogram Analysis Method (WHAM).

We found that when the protein residues with highest numbers of native contacts are assumed hydrophobic, the temperature of the folding transition steadily decreases as the intensity of the hydrophobic

interaction increases. This enables the chaperonin to unfold any potentially misfolded initial conformations of the protein at the beginning of its cycle. As the cycle advances and hydrophobic interaction with the internal surface of the chaperonin weakens, the temperature of the folding transition steadily increases and this, together with protection from misfolding and aggregation, ensures a high probability (approximately 40% for both lattice protein systems) of the protein having its native conformation at the end of the chaperonin cycle. For this kind of hydrophobic decoration the chaperonin constitutes an active cage, providing an annealing heat treatment to the protein in each of its cycles. Folding thus occurs through iterative annealing.

When hydrophobic nature is assigned to residues with lower numbers of native contacts we found that the chaperonin eventually loses its ability to unfold the protein at the beginning of its cycle and thus becomes a passive Anfinsen's cage. The probability of the protein having its native conformation at the end of the chaperonin cycle remains the same (approximately 40% for both lattice protein systems), provided the protein does not become blocked in a misfolded conformation and can reach thermal equilibrium.

The values found for the probability of the protein having its native conformation at the end of the chaperonin cycle can thus explain the observed protein folding rates.

Future work could involve clarifying the role that intermediate states play in the folding and knotting processes and replicating this investigation using off-lattice models of real proteins.

In particular, regarding intermediate states, some studied model sequences exhibit a population of intermediate states for several values of the hydrophobic parameter and box sizes. Our preliminary analysis of one such intermediate state revealed that its knotting probability is considerably higher than that of an intermediate with the same fraction of native contacts populated under steric confinement. It is therefore likely that the chaperonin plays an active role in the knotting process by facilitating the population of productive knoted intermediates that fold fast to the native state. Future work could involve a thorough analysis of these intermediate states to clearly establish their role in folding.

Appendix A

Replica Exchange implementation

```
/*****************************************************************************
 *                                                                           *
 *   Implementation of Replica Exchange (a.k.a. Parallel Tempering) method   *
 *                                                                           *
 *   Improved version by: João N. C. Especial                                *
 *                                                                           *
 *   Date: Jun 20th, 2018                                                    *
 *                                                                           *
 *****************************************************************************/

void runREM(gsl_rng * rng)
{
    int NodeIndex[SIZE];
    int NodeSelect1, NodeSelect2;
    int i, j, k, REM_Trial, iErr, iTmp;
    double delta_E, delta_beta, Delta;
    int buf_in_x[N], buf_in_y[N], buf_in_z[N];
    int buf_out_x[N], buf_out_y[N], buf_out_z[N];
    MPI_Status mpi_status;

    for (i = 0; i < SIZE; i++)
    {
        NodeIndex[i] = i;
    }

    iErr = MPI_Gather(&E, 1, MPI_DOUBLE, EList, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
    iErr = MPI_Gather(&E_box, 1, MPI_DOUBLE, EList_box, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
    /* Gathers distinct messages from each task in the group to a single destination task.
     * This routine is the reverse operation of MPI_Scatter */
    /* iErr = MPI_GATHER(sendbuf, sendcnt, sendtype, recvbuf, recvcount, recvtype, root, comm) */

    REM_Trial = 1;

    // Start procedure for replica exchange in rank 0 task
    if(Rank == 0)
    {
        for(i = 0; i < REM_Trial; i++)
        {
            NodeSelect1 = (int)(SIZE * gsl_rng_uniform(rng));
            NodeSelect2 = (int)(SIZE * gsl_rng_uniform(rng));

            if (NodeSelect1 != NodeSelect2)
            {
                delta_beta = (1.0 / TList[NodeSelect2]) - (1.0 / TList[NodeSelect1]);
                delta_E = (EList[NodeSelect2] + EList_box[NodeSelect2])
                          - (EList[NodeSelect1] + EList_box[NodeSelect1]);
                Delta = -delta_beta * delta_E;

                if ((delta_all >= 0) || (gsl_rng_uniform(rng) < exp(-Delta)))
                {
                    iTmp = NodeIndex[NodeSelect1];
                    NodeIndex[NodeSelect1] = NodeIndex[NodeSelect2];
                    NodeIndex[NodeSelect2] = iTmp;
```

```
            }
        }
    }
}
// End procedure for replica exchange in rank 0 task

iErr = MPI_Bcast(NodeIndex, SIZE, MPI_INT, 0, MPI_COMM_WORLD);
/* Broadcast (sends) a message from the process with rank "root"
 * to all other processes in the group */
/* iErr = MPI_BCAST(buffer, count, datatype, root, comm) */

if(Rank !=  NodeIndex[Rank]) // If the replica's conformation is to be exchanged
{
    for (j = 0; j < N; j++) // Store current conformation in the output buffer
    {
        buf_out_x[j] = x[j];
        buf_out_y[j] = y[j];
        buf_out_z[j] = z[j];
    }

    // Lower rank replica sends its conformation first and then receives new conformation
    if(Rank < NodeIndex[Rank])
    {
        iErr = MPI_Send(buf_out_x, N, MPI_INT, NodeIndex[Rank], 0, MPI_COMM_WORLD);
        iErr = MPI_Send(buf_out_y, N, MPI_INT, NodeIndex[Rank], 1, MPI_COMM_WORLD);
        iErr = MPI_Send(buf_out_z, N, MPI_INT, NodeIndex[Rank], 2, MPI_COMM_WORLD);
        iErr = MPI_Recv(buf_in_x, N, MPI_INT, NodeIndex[Rank], 0, MPI_COMM_WORLD, &mpi_status);
        iErr = MPI_Recv(buf_in_y, N, MPI_INT, NodeIndex[Rank], 1, MPI_COMM_WORLD, &mpi_status);
        iErr = MPI_Recv(buf_in_z, N, MPI_INT, NodeIndex[Rank], 2, MPI_COMM_WORLD, &mpi_status);
    }

    // Higher rank replica first receives new conformation and then sends its conformation
    if(Rank > NodeIndex[Rank])
    {
        iErr = MPI_Recv(buf_in_x, N, MPI_INT, NodeIndex[Rank], 0, MPI_COMM_WORLD, &mpi_status);
        iErr = MPI_Recv(buf_in_y, N, MPI_INT, NodeIndex[Rank], 1, MPI_COMM_WORLD, &mpi_status);
        iErr = MPI_Recv(buf_in_z, N, MPI_INT, NodeIndex[Rank], 2, MPI_COMM_WORLD, &mpi_status);
        iErr = MPI_Send(buf_out_x, N, MPI_INT, NodeIndex[Rank], 0, MPI_COMM_WORLD);
        iErr = MPI_Send(buf_out_y, N, MPI_INT, NodeIndex[Rank], 1, MPI_COMM_WORLD);
        iErr = MPI_Send(buf_out_z, N, MPI_INT, NodeIndex[Rank], 2, MPI_COMM_WORLD);
    }

    for (j = 0; j < N; j++) // Load new conformation from the input buffer
    {
        x[j] = buf_in_x[j];
        y[j] = buf_in_y[j];
        z[j] = buf_in_z[j];
    }
}

// Reset all replica states in accordance to their current conformation,
// to prevent numeric error accumulation
for (i = 0; i < L; i++)
{
    for (j = 0; j < L; j++)
    {
        for (k = 0; k < L; k++)
        {
            lattice[i][j][k] = 0;
            lbead[i][j][k] = -2;
        }
    }
}
for (i = 0; i < N; i++)
{
    if (lattice[x[i]][y[i]][z[i]] != 0)
        printf("Bad initial condition - 2\n");
    lattice[x[i]][y[i]][z[i]] = sq[js][i];
    lbead[x[i]][y[i]][z[i]] = i;
}
E = energy(lbead, x, y, z);
```

```
    E_box = energy_box(lbead, x, y, z);
    call = contact_map(x, y, z);
}
```

Appendix B

Knot detection implementation

```
/*****************************************************************************
 *                                                                           *
 *   Implementation of the Koniaris-Muthukumar-Taylor (KMT) knot detection method   *
 *                                                                           *
 *   Author: João N. C. Especial                                             *
 *                                                                           *
 *   Date:   Mar 25th, 2018                                                  *
 *                                                                           *
 *****************************************************************************/

// lknotp: Returns 1 if lattice conformation embeds a knot and 0 if it does not

int lknotp(int *x_arg, int *y_arg, int *z_arg, int n)
{
    int x_lat[n], y_lat[n], z_lat[n], x12, y12, z12, x21, y21, z21,
        x32, y32, z32, x43, y43, z43, x12X32, y12X32, z12X32,
        i, j, jp, jp2, change_flag;

    double x_cnt[n], y_cnt[n], z_cnt[n];

    // Preserve the argument conformation by
    // copying it to a working data structure

    for (i = 0; i < n; i++)
    {
        x_lat[i] = x_arg[i];
        y_lat[i] = y_arg[i];
        z_lat[i] = z_arg[i];
    }

    // Remove U-turns

    do
    {
        change_flag = 0;

        for (i = 0; i < n-3; i++)
        {
            x21 = x_lat[i+1] - x_lat[i];
            y21 = y_lat[i+1] - y_lat[i];
            z21 = z_lat[i+1] - z_lat[i];

            x43 = x_lat[i+3] - x_lat[i+2];
            y43 = y_lat[i+3] - y_lat[i+2];
            z43 = z_lat[i+3] - z_lat[i+2];

            if (x21 + x43 == 0 && y21 + y43 == 0 && z21 + z43 == 0)
            {
                // Delete beads i+1 and i+2

                for (j = i+1; j < n-2; j++)
                {
                    jp2 = j + 2;
```

```
                    x_lat[j] = x_lat[jp2];
                    y_lat[j] = y_lat[jp2];
                    z_lat[j] = z_lat[jp2];
                }
                n -= 2;

                change_flag = 1;

                i--;
            }
        }
    } while (change_flag);

    // Delete inner beads from all straight lines of beads

    for (i = 1; i < n-1; i++)
    {
        x12 = x_lat[i-1] - x_lat[i];
        y12 = y_lat[i-1] - y_lat[i];
        z12 = z_lat[i-1] - z_lat[i];

        x32 = x_lat[i+1] - x_lat[i];
        y32 = y_lat[i+1] - y_lat[i];
        z32 = z_lat[i+1] - z_lat[i];

        x12X32 = y12 * z32 - z12 * y32;
        y12X32 = z12 * x32 - x12 * z32;
        z12X32 = x12 * y32 - y12 * x32;

        if (x12X32 == 0 && y12X32 == 0 && z12X32 == 0)
        {
            // Delete bead i

            for (j = i; j < n-1; j++)
            {
                jp = j + 1;

                x_lat[j] = x_lat[jp];
                y_lat[j] = y_lat[jp];
                z_lat[j] = z_lat[jp];
            }
            n--;

            change_flag = 1;

            i--;
        }
    }

    // Convert conformation to off-lattice

    for (i = 0; i < n; i++)
    {
        x_cnt[i] = (double)x_lat[i];
        y_cnt[i] = (double)y_lat[i];
        z_cnt[i] = (double)z_lat[i];
    }

    return knotp(x_cnt, y_cnt, z_cnt, n);
}

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

// knotp: Returns 1 if conformation embeds a knot and 0 if it does not

int knotp(double *x_arg, double *y_arg, double *z_arg, int n)
{
    int i, im, ip, j, jp, change_flag;
    double x_cnt[n], y_cnt[n], z_cnt[n],
           xc, yc, zc, xci, yci, zci;
```

```
// Preserve the argument conformation by
// copying it to a working data structure

for (i = 0; i < n; i++)
{
    x_cnt[i] = x_arg[i];
    y_cnt[i] = y_arg[i];
    z_cnt[i] = z_arg[i];
}

// Apply Koniaris-Muthukumar-Taylor (KMT) algorithm

do
{
    change_flag = 0;

    for (i = 1; i < n-1; i++)
    {
        // Koniaris Muthukumar (KM) step - Try bead deletion

        if (clear1p(i, x_cnt, y_cnt, z_cnt, n))
        {
            // Delete bead i

            for (j = i; j < n-1; j++)
            {
                jp = j + 1;

                x_cnt[j] = x_cnt[jp];
                y_cnt[j] = y_cnt[jp];
                z_cnt[j] = z_cnt[jp];
            }
            n--;

            change_flag = 1;

            i--;
            continue;
        }

        // Cannot delete bead i, hence,
        // Taylor (T) step - Try moving it

        im = i - 1;
        ip = i + 1;

        xc = (x_cnt[im] + x_cnt[i] + x_cnt[ip]) / 3.0;
        yc = (y_cnt[im] + y_cnt[i] + y_cnt[ip]) / 3.0;
        zc = (z_cnt[im] + z_cnt[i] + z_cnt[ip]) / 3.0;

        xci = xc - x_cnt[i];
        yci = yc - y_cnt[i];
        zci = zc - z_cnt[i];

        if (xci * xci + yci * yci + zci * zci < 0.000001)
        {
            // No point in moving bead i

            continue;
        }

        if (clear2p(i, xc, yc, zc, x_cnt, y_cnt, z_cnt, n))
        {
            // Move bead i

            x_cnt[i] = xc;
            y_cnt[i] = yc;
            z_cnt[i] = zc;

            change_flag = 1;
```

```
            }
        }
    } while (change_flag);

    if (n > 2)
        return 1;
    else
        return 0;
}

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

int clear1p(int i, double *x_cnt, double *y_cnt, double *z_cnt, int n)
{
    int im, ip, j, jp;

    im = i - 1;
    ip = i + 1;

    for (j = 0; j < n-1; j++)
    {
        jp = j + 1;

        if (jp == im || jp == i || j == i || j == ip)
            continue;

        if (crossp(x_cnt[im], y_cnt[im], z_cnt[im],
                   x_cnt[i],  y_cnt[i],  z_cnt[i],
                   x_cnt[ip], y_cnt[ip], z_cnt[ip],
                   x_cnt[j],  y_cnt[j],  z_cnt[j],
                   x_cnt[jp], y_cnt[jp], z_cnt[jp]))
        {
            return 0;
        }
    }

    return 1;
}

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

int clear2p(int i, double xc,     double yc,     double zc,
                   double *x_cnt, double *y_cnt, double *z_cnt, int n)
{
    int im, ip, j, jp;

    im = i - 1;
    ip = i + 1;

    for (j = 0; j < n-1; j++)
    {
        jp = j + 1;

        if (jp == im || jp == i || j == i || j == ip)
            continue;

        if (crossp(x_cnt[im], y_cnt[im], z_cnt[im],
                   x_cnt[i],  y_cnt[i],  z_cnt[i],
                   xc,    yc,    zc,
                   x_cnt[j],  y_cnt[j],  z_cnt[j],
                   x_cnt[jp], y_cnt[jp], z_cnt[jp]) ||
            crossp(xc,    yc,    zc,
                   x_cnt[i],  y_cnt[i],  z_cnt[i],
                   x_cnt[ip], y_cnt[ip], z_cnt[ip],
                   x_cnt[j],  y_cnt[j],  z_cnt[j],
                   x_cnt[jp], y_cnt[jp], z_cnt[jp]))
        {
            return 0;
        }
    }
```

```
    return 1;
}

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

int crossp(double x1, double y1, double z1,
           double x2, double y2, double z2,
           double x3, double y3, double z3,
           double x4, double y4, double z4,
           double x5, double y5, double z5)
{
    double x12, y12, z12, x32, y32, z32, x12X32, y12X32, z12X32,
           x42, y42, z42, x52, y52, z52, x45, y45, z45,
           det, det1, det2, det3, a14, a24, a34, a15, a25, a35, a1, a2; //, a3;

    x12 = x1 - x2;
    y12 = y1 - y2;
    z12 = z1 - z2;

    x32 = x3 - x2;
    y32 = y3 - y2;
    z32 = z3 - z2;

    x12X32 = y12 * z32 - z12 * y32;
    y12X32 = z12 * x32 - x12 * z32;
    z12X32 = x12 * y32 - y12 * x32;

    if (x12X32 == 0.0 && y12X32 == 0.0 && z12X32 == 0.0)
    {
        return 0;
    }

    x42 = x4 - x2;
    y42 = y4 - y2;
    z42 = z4 - z2;

    det = x12 * y32 * z12X32 + x32 * y12X32 * z12 + x12X32 * y12 * z32
        - x12 * y12X32 * z32 - x32 * y12 * z12X32 - x12X32 * y32 * z12;

    det1 = x42 * y32 * z12X32 + x32 * y12X32 * z42 + x12X32 * y42 * z32
         - x42 * y12X32 * z32 - x32 * y42 * z12X32 - x12X32 * y32 * z42;

    det2 = x12 * y42 * z12X32 + x42 * y12X32 * z12 + x12X32 * y12 * z42
         - x12 * y12X32 * z42 - x42 * y12 * z12X32 - x12X32 * y42 * z12;

    det3 = x12 * y32 * z42 + x32 * y42 * z12 + x42 * y12 * z32
         - x12 * y42 * z32 - x32 * y12 * z42 - x42 * y32 * z12;

    a14 = det1 / det;
    a24 = det2 / det;
    a34 = det3 / det;

    if (a34 == 0.0 &&
        a14 >= 0.0 && a24 >= 0.0 && a14 + a24 <= 1.0)
    {
        return 1;
    }

    x52 = x5 - x2;
    y52 = y5 - y2;
    z52 = z5 - z2;

    det1 = x52 * y32 * z12X32 + x32 * y12X32 * z52 + x12X32 * y52 * z32
         - x52 * y12X32 * z32 - x32 * y52 * z12X32 - x12X32 * y32 * z52;

    det2 = x12 * y52 * z12X32 + x52 * y12X32 * z12 + x12X32 * y12 * z52
         - x12 * y12X32 * z52 - x52 * y12 * z12X32 - x12X32 * y52 * z12;

    det3 = x12 * y32 * z52 + x32 * y52 * z12 + x52 * y12 * z32
         - x12 * y52 * z32 - x32 * y12 * z52 - x52 * y32 * z12;
```

```
    a15 = det1 / det;
    a25 = det2 / det;
    a35 = det3 / det;

    if (a35 == 0.0 &&
        a15 >= 0.0 && a25 >= 0.0 && a15 + a25 <= 1.0)
    {
        return 1;
    }

    if (a34 * a35 >= 0.0)
    {
        return 0;
    }

    x45 = x4 - x5;
    y45 = y4 - y5;
    z45 = z4 - z5;

    det = x12 * y32 * z45 + x32 * y45 * z12 + x45 * y12 * z32
        - x12 * y45 * z32 - x32 * y12 * z45 - x45 * y32 * z12;

    det1 = x42 * y32 * z45 + x32 * y45 * z42 + x45 * y42 * z32
         - x42 * y45 * z32 - x32 * y42 * z45 - x45 * y32 * z42;

    det2 = x12 * y42 * z45 + x42 * y45 * z12 + x45 * y12 * z42
         - x12 * y45 * z42 - x42 * y12 * z45 - x45 * y42 * z12;

    // det3 = x12 * y32 * z42 + x32 * y42 * z12 + x42 * y12 * z32
    //      - x12 * y42 * z32 - x32 * y12 * z42 - x42 * y32 * z12;

    a1 = det1 / det;
    a2 = det2 / det;
    // a3 = det3 / det;

    if (a1 >= 0.0 && a2 >= 0.0 && a1 + a2 <= 1.0)
    {
        return 1;
    }

    return 0;
}
```

Appendix C

WHAM implementation

```
/*****************************************************************************************
 *                                                                                       *
 *   Implementation of the Weighted Histogram Analysis Method (WHAM) data analysis method   *
 *                                                                                       *
 *   Author: João N. C. Especial                                                         *
 *                                                                                       *
 *   Date:   Aug 7th, 2018                                                               *
 *                                                                                       *
 *****************************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <assert.h>

#define  k_B  1.0             // Boltzmann constant when using adimensional H* and T*


int main(int argc, char * argv[])
{
    FILE * in_file, * Omega_H_conv, * F_T_conv, * Omega_H, * omega_HQ, * omega_HW, * omega_HR,
        * omega_HL, * WHAM_T, * P_QT, * L_QT, * F_QT, * F_QT_relnat, * tm, * L_QatTm, * tf,
        * F_QatTf;
    int i, j, k, jQ, flag, num_replicas, num_residues, Lknot, Lknot_min, Lknot_max, num_H_bins,
        num_Q_bins, num_W_bins, num_R_bins, num_L_bins;
    long * N_T, ** N_HT, * N_H, *** N_HQT, ** N_HQ, *** N_HWT, ** N_HW, *** N_HRT, ** N_HR,
        *** N_HLT, ** N_HL, *** N_HQL, ** N_QT, lsum, iter, max_iterations;
    long long mcs, num_equilibration_mcs;
    double * T, eta_hp, E_intra, E_intra_min, E_intra_max, * E_intra_bin, E_E_intra, * Z_QT,
        p_QT, E_inter, E_inter_min, E_inter_max, E_inter_bin_width, * E_inter_bin, E_E_inter,
        E_total, E_total_min, E_total_max, E_total_bin_width, Rg, Rg_min, Rg_max, Rg_bin_width,
        * Rg_bin, E_Rg, E_Lknot, E_Lknot_Q, dsum, * H, * Omega, ** Omega_HQ, ** Omega_HW,
        ** Omega_HR, ** Omega_HL, *** Omega_HQL, * beta, * f, * f_prev, delta_f, delta_f_max,
        tolerance, Tp_resolution, Tp, betap, Z, U, Cv, Cv_max, Tm, betam, S, F, F0, F_1st_min,
        Tf, betaf;
    char s[512];

    // Validate number of arguments

    if (argc != 4)
    {
        printf("Usage: ./wham wham_parameters_file temperature_grid_file protein_data_file\n");
        exit(-1);
    }

    // Read wham parameters

    if ((in_file = fopen(argv[1], "r")) == NULL)
    {
        printf("Unable to open WHAM parameters file. Exiting.\n");
        exit(-1);
    }
```

```
fscanf(in_file, "%lld", &num_equilibration_mcs);      // Number of initial mcs required for
                                                     // equilibration
fscanf(in_file, "%ld",  &max_iterations);             // Maximum number of iterations
fscanf(in_file, "%lf",  &tolerance);                  // Convergence tolerance
fscanf(in_file, "%lf",  &Tp_resolution);              // Resolution of temperature plots

fclose(in_file);

// Read temperature grid

if ((in_file = fopen(argv[2], "r")) == NULL)
{
    printf("Unable to open temperature grid file. Exiting.\n");
    exit(-1);
}

fscanf(in_file, "%d", &num_replicas);

T = malloc(num_replicas * sizeof(double));

for (k = 0; k < num_replicas; k++)
    fscanf(in_file, "%lf", &T[k]);

fclose(in_file);

// Read wall interaction strength

if ((in_file = fopen(argv[3], "r")) == NULL)
{
    printf("Unable to open protein data file. Exiting.\n");
    exit(-1);
}

fscanf(in_file, "%d", &num_residues);

for (k = 0; k < 7; k++)
    fgets(s, 512, in_file); // Skip conformation stuff

eta_hp = 0.0;
for (k = 0; k < num_residues; k++)
{
    fscanf(in_file, "%lf", &dsum);
    if (dsum > eta_hp)
        eta_hp = dsum;
}

fclose(in_file);

printf("eta_hp = %f\n", eta_hp);
fflush(stdout);

// Determine maxima, minima and numbers of samples

printf("Scanning Files...\n");
fflush(stdout);

E_intra_min = 0.0;
E_intra_max = -1000000.0;
E_inter_min = 0.0;
E_inter_max = -1000000.0;
E_total_min = 0.0;
E_total_max = -1000000.0;
Rg_min = 100000.0;
Rg_max = 0.0;
Lknot_min = 2;
Lknot_max = -1;

N_T = malloc(num_replicas * sizeof(long));
for (k = 0; k < num_replicas; k++)
    N_T[k] = 0;
```

```
for (k = 0; k < num_replicas; k++)
{
    sprintf(s, "Length-N=%d-T=%5.3lf-ID=%d.dat", num_residues, T[k], k);
    printf("Scanning file \"%s\"\n", s);
    fflush(stdout);

    if ((in_file = fopen(s, "r")) == NULL)
    {
        printf("File \"%s\" not found. Exiting.\n", s);
        exit(-1);
    }

    while (!feof(in_file))
    {
        if (fscanf(in_file, "%lf %lf %lf %lf %d %lld", &E_intra, &E_inter, &E_total,
                                                       &Rg, &Lknot, &mcs) > 0)
        {
            fgets(s, 512, in_file); // Skip equilibration stuff

            if(mcs > num_equilibration_mcs)
            {
                if (E_intra > E_intra_max)
                    E_intra_max = E_intra;
                if (E_intra < E_intra_min)
                    E_intra_min = E_intra;
                if (E_inter > E_inter_max)
                    E_inter_max = E_inter;
                if (E_inter < E_inter_min)
                    E_inter_min = E_inter;
                if (E_total > E_total_max)
                    E_total_max = E_total;
                if (E_total < E_total_min)
                    E_total_min = E_total;
                if (Rg > Rg_max)
                    Rg_max = Rg;
                if (Rg < Rg_min)
                    Rg_min = Rg;
                if (Lknot > Lknot_max)
                    Lknot_max = Lknot;
                if (Lknot < Lknot_min)
                    Lknot_min = Lknot;

                N_T[k]++;
            }
        }
    }

    fclose(in_file);
}

printf("E_intra_min = %f\n", E_intra_min);
printf("E_intra_max = %f\n", E_intra_max);
printf("E_inter_min = %f\n", E_inter_min);
printf("E_inter_max = %f\n", E_inter_max);
printf("E_total_min = %f\n", E_total_min);
printf("E_total_max = %f\n", E_total_max);
printf("Rg_min = %f\n", Rg_min);
printf("Rg_max = %f\n", Rg_max);
printf("Lknot_min = %d\n", Lknot_min);
printf("Lknot_max = %d\n", Lknot_max);
flag = 0;
for (k = 1; k < num_replicas; k++)
    if (N_T[k] != N_T[0])
        flag = 1;
if (flag)
    for (k = 0; k < num_replicas; k++)
        printf("Number of samples in file %d: %ld\n", k, N_T[k]);
else
    printf("Number of samples in all files: %ld\n", N_T[0]);
fflush(stdout);
```

```
// Set numbers of histogram bins

num_Q_bins = (int)floor(E_intra_max - E_intra_min + 1.5);
printf("num_Q_bins = %d\n", num_Q_bins);
if (eta_hp != 0.0)
    num_W_bins = (int)floor((E_inter_max - E_inter_min) / eta_hp + 1.5);
else
    num_W_bins = 1;
printf("num_W_bins = %d\n", num_W_bins);
num_H_bins = num_Q_bins;
printf("num_H_bins = %d\n", num_H_bins);
num_R_bins = (3 * num_Q_bins) / 2;
printf("num_R_bins = %d\n", num_R_bins);
num_L_bins = 2;
fflush(stdout);

// Set histogram bin widths

E_total_bin_width = (E_total_max - E_total_min) / (num_H_bins - 1);
if (num_W_bins != 1)
    E_inter_bin_width = (E_inter_max - E_inter_min) / (num_W_bins - 1);
else
    E_inter_bin_width = 1.0;
Rg_bin_width = (Rg_max - Rg_min) / (num_R_bins - 1);

printf("E_total_bin_width = %f\n", E_total_bin_width);
printf("E_inter_bin_width = %f\n", E_inter_bin_width);
printf("Rg_bin_width = %f\n", Rg_bin_width);
fflush(stdout);

// Compute central values of histogram bins

H           = malloc(num_H_bins * sizeof(double));
E_intra_bin = malloc(num_Q_bins * sizeof(double));
E_inter_bin = malloc(num_W_bins * sizeof(double));
Rg_bin      = malloc(num_R_bins * sizeof(double));

for (i = 0; i < num_H_bins; i++)
    H[i] = E_total_min + i * E_total_bin_width;

for (j = 0; j < num_Q_bins; j++)
    E_intra_bin[j] = E_intra_min + j;

for (j = 0; j < num_W_bins; j++)
    E_inter_bin[j] = E_inter_min + j * E_inter_bin_width;

for (j = 0; j < num_R_bins; j++)
    Rg_bin[j] = Rg_min + j * Rg_bin_width;

// Allocate memory to histogram arrays

N_HT  = malloc(num_H_bins * sizeof(long *));
N_H   = malloc(num_H_bins * sizeof(long));
N_HQT = malloc(num_H_bins * sizeof(long **));
N_HQ  = malloc(num_H_bins * sizeof(long *));
N_HWT = malloc(num_H_bins * sizeof(long **));
N_HW  = malloc(num_H_bins * sizeof(long *));
N_HRT = malloc(num_H_bins * sizeof(long **));
N_HR  = malloc(num_H_bins * sizeof(long *));
N_HLT = malloc(num_H_bins * sizeof(long **));
N_HL  = malloc(num_H_bins * sizeof(long *));
N_HQL = malloc(num_H_bins * sizeof(long **));
N_QT  = malloc(num_Q_bins * sizeof(long *));

for (i = 0; i < num_H_bins; i++)
{
    N_HT[i]  = malloc(num_replicas * sizeof(long));
    N_HQT[i] = malloc(num_Q_bins * sizeof(long *));
    N_HQ[i]  = malloc(num_Q_bins * sizeof(long));
    N_HWT[i] = malloc(num_W_bins * sizeof(long *));
    N_HW[i]  = malloc(num_W_bins * sizeof(long));
```

```
    N_HRT[i] = malloc(num_R_bins * sizeof(long *));
    N_HR[i]  = malloc(num_R_bins * sizeof(long));
    N_HLT[i] = malloc(num_L_bins * sizeof(long *));
    N_HL[i]  = malloc(num_L_bins * sizeof(long));
    N_HQL[i] = malloc(num_Q_bins * sizeof(long *));

    for (j = 0; j < num_Q_bins; j++)
        N_HQT[i][j] = malloc(num_replicas * sizeof(long));
    for (j = 0; j < num_W_bins; j++)
        N_HWT[i][j] = malloc(num_replicas * sizeof(long));
    for (j = 0; j < num_R_bins; j++)
        N_HRT[i][j] = malloc(num_replicas * sizeof(long));
    for (j = 0; j < num_L_bins; j++)
        N_HLT[i][j] = malloc(num_replicas * sizeof(long));
    for (j = 0; j < num_Q_bins; j++)
        N_HQL[i][j] = malloc(num_L_bins * sizeof(long));
}

for (j = 0; j < num_Q_bins; j++)
    N_QT[j] = malloc(num_replicas * sizeof(long));

// Initialize histogram arrays

for (i = 0; i < num_H_bins; i++)
{
    for (k = 0; k < num_replicas; k++)
    {
        N_HT[i][k] = 0;
        for (j = 0; j < num_Q_bins; j++)
            N_HQT[i][j][k] = 0;
        for (j = 0; j < num_W_bins; j++)
            N_HWT[i][j][k] = 0;
        for (j = 0; j < num_R_bins; j++)
            N_HRT[i][j][k] = 0;
        for (j = 0; j < num_L_bins; j++)
            N_HLT[i][j][k] = 0;
    }
    N_H[i] = 0;
    for (j = 0; j < num_Q_bins; j++)
        N_HQ[i][j] = 0;
    for (j = 0; j < num_W_bins; j++)
        N_HW[i][j] = 0;
    for (j = 0; j < num_R_bins; j++)
        N_HR[i][j] = 0;
    for (j = 0; j < num_L_bins; j++)
        N_HL[i][j] = 0;
    for (jQ = 0; jQ < num_Q_bins; jQ++)
        for (j = 0; j < num_L_bins; j++)
            N_HQL[i][jQ][j] = 0;
}

for (j = 0; j < num_Q_bins; j++)
    for (k = 0; k < num_replicas; k++)
        N_QT[j][k] = 0;

// Build histograms

printf("Building Histograms...\n");
fflush(stdout);

for (k = 0; k < num_replicas; k++)
{
    sprintf(s, "Length-N=%d-T=%5.3lf-ID=%d.dat", num_residues, T[k], k);
    printf("Processing file \"%s\"\n", s);
    fflush(stdout);

    if ((in_file = fopen(s, "r")) == NULL)
    {
        printf("File \"%s\" not found. Exiting.\n", s);
        exit(-1);
    }
```

```
    while (!feof(in_file))
    {
        if (fscanf(in_file, "%lf %lf %lf %lf %d %lld", &E_intra, &E_inter, &E_total,
                                                       &Rg, &Lknot, &mcs) > 0)
        {
            fgets(s, 512, in_file); // Skip equilibration stuff

            if(mcs > num_equilibration_mcs)
            {
                // H
                i = (int)floor((E_total - E_total_min) / E_total_bin_width + 0.5);
                N_H[i]++;
                N_HT[i][k]++;
                // Q
                j = (int)floor(E_intra - E_intra_min + 0.5);
                N_HQ[i][j]++;
                N_HQT[i][j][k]++;
                N_QT[j][k]++;
                jQ = j;
                // W
                j = (int)floor((E_inter - E_inter_min) / E_inter_bin_width + 0.5);
                N_HW[i][j]++;
                N_HWT[i][j][k]++;
                // R
                j = (int)floor((Rg - Rg_min) / Rg_bin_width + 0.5);
                N_HR[i][j]++;
                N_HRT[i][j][k]++;
                // L
                j = (int)floor(Lknot - Lknot_min + 0.5);
                N_HL[i][j]++;
                N_HLT[i][j][k]++;
                N_HQL[i][jQ][j]++;
            }
        }
    }

    fclose(in_file);
}

// Validate histograms

for (k = 0; k < num_replicas; k++)
{
    lsum = 0;
    for (i = 0; i < num_H_bins; i++)
        lsum += N_HT[i][k];
    assert(lsum == N_T[k]);
}

for (i = 0; i < num_H_bins; i++)
{
    lsum = 0;
    for (k = 0; k < num_replicas; k++)
        lsum += N_HT[i][k];
    assert(lsum == N_H[i]);

    lsum = 0;
    for (j = 0; j < num_Q_bins; j++)
        lsum += N_HQ[i][j];
    assert(lsum == N_H[i]);

    lsum = 0;
    for (j = 0; j < num_W_bins; j++)
        lsum += N_HW[i][j];
    assert(lsum == N_H[i]);

    lsum = 0;
    for (j = 0; j < num_R_bins; j++)
        lsum += N_HR[i][j];
    assert(lsum == N_H[i]);
```

```
    lsum = 0;
    for (j = 0; j < num_L_bins; j++)
        lsum += N_HL[i][j];
    assert(lsum == N_H[i]);

    lsum = 0;
    for (jQ = 0; jQ < num_Q_bins; jQ++)
        for (j = 0; j < num_L_bins; j++)
            lsum += N_HQL[i][jQ][j];
    assert(lsum == N_H[i]);

    for (k = 0; k < num_replicas; k++)
    {
        lsum = 0;
        for (j = 0; j < num_Q_bins; j++)
            lsum += N_HQT[i][j][k];
        assert(lsum == N_HT[i][k]);

        lsum = 0;
        for (j = 0; j < num_W_bins; j++)
            lsum += N_HWT[i][j][k];
        assert(lsum == N_HT[i][k]);

        lsum = 0;
        for (j = 0; j < num_R_bins; j++)
            lsum += N_HRT[i][j][k];
        assert(lsum == N_HT[i][k]);

        lsum = 0;
        for (j = 0; j < num_L_bins; j++)
            lsum += N_HLT[i][j][k];
        assert(lsum == N_HT[i][k]);
    }
}

for (k = 0; k < num_replicas; k++)
{
    lsum = 0;
    for (j = 0; j < num_Q_bins; j++)
        lsum += N_QT[j][k];
    assert(lsum == N_T[k]);
}

// Do WHAM analysis

printf("WHAM analysis running...\n");
fflush(stdout);

beta   = malloc(num_replicas * sizeof(double));
Omega  = malloc(num_H_bins * sizeof(double));
f      = malloc(num_replicas * sizeof(double));
f_prev = malloc(num_replicas * sizeof(double));

// Compute inverse temperature parameters

for (k = 0; k < num_replicas; k++)
    beta[k] = 1.0 / (k_B * T[k]);

// Initialize number of states array

for (i = 0; i < num_H_bins; i++)
    Omega[i] = 0.0;

// Set initial reduced free energy values

for (k = 0; k < num_replicas; k++)
    f[k] = 0.0;

Omega_H_conv = fopen("Omega_H_conv.dat", "w");
F_T_conv     = fopen("F_T_conv.dat", "w");
```

```
for (iter = 0; iter < max_iterations; iter++)
{
    // Compute number of states in each energy bin

    for (i = 0; i < num_H_bins; i++)
    {
        if (N_H[i] > 0)
        {
            dsum = 0.0;
            for (k = 0; k < num_replicas; k++)
            {
                dsum += N_T[k] * exp(f[k] - beta[k] * H[i]);
            }

            Omega[i] = N_H[i] / dsum;
        }

        fprintf(Omega_H_conv, "%f\t%e\n", H[i], Omega[i]);
    }

    // Compute reduced free energy for each replica

    for (k = 0; k < num_replicas; k++)
    {
        dsum = 0.0;
        for (i = 0; i < num_H_bins; i++)
        {
            if (N_H[i] > 0)
            {
                dsum += Omega[i] * exp(-beta[k] * H[i]);
            }
        }

        f_prev[k] = f[k];
        f[k] = -log(dsum);

        fprintf(F_T_conv, "%f\t%f\n", T[k], f[k]);
    }

    // Test for self-consistency convergence.

    delta_f_max = 0.0;
    for (k = 0; k < num_replicas; k++)
    {
        delta_f = fabs((f[k] - f_prev[k]) / f_prev[k]);
        if (delta_f > delta_f_max)
            delta_f_max = delta_f;
    }

    if (delta_f_max < tolerance)
        break;

    fprintf(Omega_H_conv, "\n");
    fprintf(F_T_conv, "\n");
}

if (iter < max_iterations)
    printf("Done: %ld iterations (Tolerance = 1E%.f).\n",
            iter + 1, log(tolerance) / log(10.0));
else
{
    printf("Free energy calculation has not converged in %ld iterations. Exiting.\n",
            iter);
    exit(-1);
}

fclose(Omega_H_conv);
fclose(F_T_conv);

// Project the number of states vector along the property dimensions
```

```
Omega_HQ  = malloc(num_H_bins * sizeof(double *));
Omega_HW  = malloc(num_H_bins * sizeof(double *));
Omega_HR  = malloc(num_H_bins * sizeof(double *));
Omega_HL  = malloc(num_H_bins * sizeof(double *));
Omega_HQL = malloc(num_H_bins * sizeof(double **));

for (i = 0; i < num_H_bins; i++)
{
    Omega_HQ[i]  = malloc(num_Q_bins * sizeof(double));
    Omega_HW[i]  = malloc(num_W_bins * sizeof(double));
    Omega_HR[i]  = malloc(num_R_bins * sizeof(double));
    Omega_HL[i]  = malloc(num_L_bins * sizeof(double));
    Omega_HQL[i] = malloc(num_Q_bins * sizeof(double *));
}

for (i = 0; i < num_H_bins; i++)
    for (j = 0; j < num_Q_bins; j++)
        Omega_HQL[i][j] = malloc(num_L_bins * sizeof(double));

for (i = 0; i < num_H_bins; i++)
{
    for (j = 0; j < num_Q_bins; j++)
        Omega_HQ[i][j] = Omega[i] * N_HQ[i][j] / N_H[i];
    for (j = 0; j < num_W_bins; j++)
        Omega_HW[i][j] = Omega[i] * N_HW[i][j] / N_H[i];
    for (j = 0; j < num_R_bins; j++)
        Omega_HR[i][j] = Omega[i] * N_HR[i][j] / N_H[i];
    for (j = 0; j < num_L_bins; j++)
        Omega_HL[i][j] = Omega[i] * N_HL[i][j] / N_H[i];
    for (jQ = 0; jQ < num_Q_bins; jQ++)
        for (j = 0; j < num_L_bins; j++)
            Omega_HQL[i][jQ][j] = Omega[i] * N_HQL[i][jQ][j] / N_H[i];
}

// Create output files

Omega_H  = fopen("Omega_H.dat", "w");
for (i = 0; i < num_H_bins; i++)
    fprintf(Omega_H, "%f\t%e\n", H[i], Omega[i]);
fclose(Omega_H);

omega_HQ  = fopen("Omega_HQ.dat", "w");
for (i = 0; i < num_H_bins; i++)
{
    for (j = 0; j < num_Q_bins; j++)
        fprintf(omega_HQ, "%f\t%f\t%e\n", H[i], E_intra_bin[j], Omega_HQ[i][j]);
    fprintf(omega_HQ, "\n");
}
fclose(omega_HQ);

omega_HW  = fopen("Omega_HW.dat", "w");
for (i = 0; i < num_H_bins; i++)
{
    for (j = 0; j < num_W_bins; j++)
        fprintf(omega_HW, "%f\t%f\t%e\n", H[i], E_inter_bin[j], Omega_HW[i][j]);
    fprintf(omega_HW, "\n");
}
fclose(omega_HW);

omega_HR  = fopen("Omega_HR.dat", "w");
for (i = 0; i < num_H_bins; i++)
{
    for (j = 0; j < num_R_bins; j++)
        fprintf(omega_HR, "%f\t%f\t%e\n", H[i], Rg_bin[j], Omega_HR[i][j]);
    fprintf(omega_HR, "\n");
}
fclose(omega_HR);

omega_HL  = fopen("Omega_HL.dat", "w");
for (i = 0; i < num_H_bins; i++)
```

```
{
    for (j = 0; j < num_L_bins; j++)
        fprintf(omega_HL, "%f\t%d\t%e\n", H[i], j, Omega_HL[i][j]);
    fprintf(omega_HL, "\n");
}
fclose(omega_HL);

WHAM_T      = fopen("WHAM_T.dat", "w");
P_QT        = fopen("P_QT.dat", "w");
L_QT        = fopen("L_QT.dat", "w");
F_QT        = fopen("F_QT.dat", "w");
F_QT_relnat = fopen("F_QT_relnat.dat", "w");

Cv_max = 0.0;
Z_QT  = malloc(num_Q_bins * sizeof(double));

for (Tp = T[0]; Tp < (T[num_replicas -1] + Tp_resolution / 2.0); Tp += Tp_resolution)
{
    betap = 1.0 / (k_B * Tp);

    dsum = 0.0;
    for (i = 0; i < num_H_bins; i++)
    {
        if (N_H[i] > 0)
        {
            dsum += Omega[i] * exp(-betap * H[i]);
        }
    }
    Z = dsum;

    F = -k_B * Tp *log(Z);

    dsum = 0.0;
    for (i = 0; i < num_H_bins; i++)
    {
        if (N_H[i] > 0)
        {
            dsum += H[i] * Omega[i] * exp(-betap * H[i]);
        }
    }
    U = dsum / Z;

    S = k_B * (log(Z) + betap * U);

    dsum = 0.0;
    for (i = 0; i < num_H_bins; i++)
    {
        if (N_H[i] > 0)
        {
            dsum += H[i] * H[i] * Omega[i] * exp(-betap * H[i]);
        }
    }
    Cv = (dsum / Z - U * U) / (k_B * Tp * Tp);

    if (Cv > Cv_max)
    {
        Tm = Tp;
        Cv_max = Cv;
    }

    E_E_intra = 0.0;
    for (j = 0; j < num_Q_bins; j++)
    {
        dsum = 0.0;
        for (i = 0; i < num_H_bins; i++)
        {
            if (N_H[i] > 0)
            {
                dsum += Omega_HQ[i][j] * exp(-betap * H[i]);
            }
        }
```

```
    Z_QT[j] = dsum;

    p_QT = Z_QT[j] / Z;

    fprintf(P_QT, "%f\t%f\t%f\n", Tp, E_intra_bin[j], p_QT);

    E_E_intra += E_intra_bin[j] * Z_QT[j];
}
E_E_intra /= Z;

fprintf(P_QT, "\n");

E_E_inter = 0.0;
for (j = 0; j < num_W_bins; j++)
{
    dsum = 0.0;
    for (i = 0; i < num_H_bins; i++)
    {
        if (N_H[i] > 0)
        {
            dsum += Omega_HW[i][j] * exp(-betap * H[i]);
        }
    }
    E_E_inter += E_inter_bin[j] * dsum;
}
E_E_inter /= Z;

E_Rg = 0.0;
for (j = 0; j < num_R_bins; j++)
{
    dsum = 0.0;
    for (i = 0; i < num_H_bins; i++)
    {
        if (N_H[i] > 0)
        {
            dsum += Omega_HR[i][j] * exp(-betap * H[i]);
        }
    }
    E_Rg += Rg_bin[j] * dsum;
}
E_Rg /= Z;

dsum = 0.0;
for (i = 0; i < num_H_bins; i++)
{
    if (N_H[i] > 0)
    {
        dsum += Omega_HL[i][1] * exp(-betap * H[i]);
    }
}
E_Lknot = dsum / Z;

fprintf(WHAM_T, "%f\t%f\t%f\t%f\t%f\t%f\t%f\t%e\t%f\t%f\t%f\t%f\t%f\t%f\n",
    Tp, E_E_intra, E_E_inter, E_E_intra + E_E_inter, U, Cv, Z, -log(Z),
    F, S, E_Rg, E_Lknot);

for (jQ = 0; jQ < num_Q_bins; jQ++)
{
    dsum = 0.0;
    for (i = 0; i < num_H_bins; i++)
    {
        if (N_H[i] > 0)
        {
            dsum += Omega_HQL[i][jQ][1] * exp(-betap * H[i]);
        }
    }
    E_Lknot_Q = dsum / Z_QT[jQ];

    // Since the temperature grid may not be uniform,
    // determining which k bin contains Tp requires this loop
    for (k = 0; k < num_replicas - 1; k++)
```

```
            if (Tp >= T[k] && Tp <= T[k + 1])
                break;

        // Require at least two samples in the bin for the expected value
        // to be regarded as representative
        if (N_QT[jQ][k] >= 2)
            fprintf(L_QT, "%f\t%f\t%f\n", Tp, E_intra_bin[jQ], E_Lknot_Q);
        // Otherwise report as missing data
        else
            fprintf(L_QT, "%f\t%f\tNIL\n", Tp, E_intra_bin[jQ]);
    }
    fprintf(L_QT, "\n");

    for (j = 0; j < num_Q_bins; j++)
    {
        dsum = 0.0;
        for (i = 0; i < num_H_bins; i++)
        {
            if (N_H[i] > 0)
            {
                dsum += Omega_HQ[i][j] * exp(-betap * H[i]);
            }
        }
        F = -k_B * Tp * log(dsum);
        if (j == 0)
            F0 = F;

        fprintf(F_QT, "%f\t%f\t%f\n", Tp, E_intra_bin[j] / E_intra_min, F);
        fprintf(F_QT_relnat, "%f\t%f\t%f\n", Tp, E_intra_bin[j] / E_intra_min, F - F0);
    }
    fprintf(F_QT, "\n");
    fprintf(F_QT_relnat, "\n");
}

fprintf(WHAM_T, "\n\n");
    for (k = 0; k < num_replicas; k++)
        fprintf(WHAM_T, "%f\t%f\n", T[k], f[k]);

fclose(WHAM_T);
fclose(P_QT);
fclose(L_QT);
fclose(F_QT);
fclose(F_QT_relnat);

tm  = fopen("Tm.dat", "w");
fprintf(tm, "%5.3f\n", Tm);
fclose(tm);

betam = 1.0 / (k_B * Tm);

L_QatTm  = fopen("L_QatTm.dat", "w");

for (jQ = 0; jQ < num_Q_bins; jQ++)
{
    dsum = 0.0;
    for (i = 0; i < num_H_bins; i++)
    {
        if (N_H[i] > 0)
        {
            dsum += Omega_HQ[i][jQ] * exp(-betam * H[i]);
        }
    }
    Z_QT[jQ] = dsum;

    dsum = 0.0;
    for (i = 0; i < num_H_bins; i++)
    {
        if (N_H[i] > 0)
        {
            dsum += Omega_HQL[i][jQ][1] * exp(-betam * H[i]);
        }
```

```
    }
    E_Lknot_Q = dsum / Z_QT[jQ];

    // Since the temperature grid may not be uniform,
    // determining which k bin contains Tm requires this loop
    for (k = 0; k < num_replicas - 1; k++)
        if (Tm >= T[k] && Tm <= T[k + 1])
            break;

    // Require at least two samples in the bin for the expected value
    // to be regarded as representative
    if (N_QT[jQ][k] >= 2)
        fprintf(L_QatTm, "%f\t%f\n", E_intra_bin[jQ], E_Lknot_Q);
    // Otherwise report as missing data
    else
        fprintf(L_QatTm, "%f\tNIL\n", E_intra_bin[jQ]);
}

fclose(L_QatTm);

for (Tf = T[0]; Tf < (T[num_replicas -1] + Tp_resolution / 2.0); Tf += Tp_resolution)
{
    betaf = 1.0 / (k_B * Tf);

    dsum = 0.0;
    for (i = 0; i < num_H_bins; i++)
    {
        if (N_H[i] > 0)
        {
            dsum += Omega_HQ[i][0] * exp(-betaf * H[i]);
        }
    }
    F0 = -k_B * Tf * log(dsum);

    F_1st_min = 100000.0;

    for (j = num_Q_bins - 1; j > 0 ; j--)
    {
        dsum = 0.0;
        for (i = 0; i < num_H_bins; i++)
        {
            if (N_H[i] > 0)
            {
                dsum += Omega_HQ[i][j] * exp(-betaf * H[i]);
            }
        }
        F = -k_B * Tf * log(dsum);

        if ((F - F0) < F_1st_min)
            F_1st_min = F - F0;
        else
            break;
    }
    if (F_1st_min < 0.0)
        break;
}

Tf -= Tp_resolution;
betaf = 1.0 / (k_B * Tf);

tf  = fopen("Tf.dat", "w");
fprintf(tf, "%5.3f\n", Tf);
fclose(tf);

F_QatTf  = fopen("F_QatTf.dat", "w");

for (j = 0; j < num_Q_bins; j++)
{
    dsum = 0.0;
    for (i = 0; i < num_H_bins; i++)
    {
```

```
            if (N_H[i] > 0)
            {
                dsum += Omega_HQ[i][j] * exp(-betaf * H[i]);
            }
        }
        F = -k_B * Tf * log(dsum);
        if (j == 0)
            F0 = F;

        fprintf(F_QatTf, "%f\t%f\n", E_intra_bin[j] / E_intra_min, F - F0);
    }

    fclose(F_QatTf);

    return 0;
}
```

Bibliography

[1] K. Huang. *Lectures on Statistical Physics and Protein Folding*. World Scientific, Singapore, 2005.

[2] J. M. Berg, J. L. Tymoczko, G. J. Gatto, Jr., and L. Stryer. *Biochemistry*. W. H. Freeman and Company, New York, 2015.

[3] M. Hayer-Hartl, A. Bracher, and F. U. Hartl. The groel-groes chaperonin machine: A nano-cage for protein folding. *Trends Biochem. Sci.*, 41(1):62–76, 2016.

[4] J. Kyte and R. F. Doolittle. A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.*, 157:105–132, 1982.

[5] K. A. Dill. Dominant forces in protein folding. *Biochemistry*, 29:7133–7155, 1990.

[6] M. L. Anson and A. E. Mirsky. The effect of denaturation on the viscosity of protein systems. *J. Gen. Physiol.*, 15(3):341–350, 1932.

[7] C. B. Anfinsen, E. Haber, M. Sela, and F. H. White. The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *Proc. Natl. Acad. Sci. USA*, 23:121–282, 1968.

[8] C. B. Anfinsen. Principles that govern folding of protein chains. *Science*, 181(4096):223–230, 1973.

[9] C. Levinthal. Are there pathways for protein folding? *J. Chim. Physico-Chim. Biol*, 65:44–45, 1968.

[10] C. Levinthal. Mossbauer spectroscopy in biological systems. In P. Debrunner, J. Tsibris, and E. Munck, editors, *Proceedings of a meeting held at Allerton house, Monticello, Illinois*, pages 22–24, Urbana, Illinois, 1969. University of Illinois Press.

[11] M. L. Mansfield. Are there knots in proteins? *Nat. Struct. Biol.*, 1(4):213–214, 1994.

[12] J. S. Richardson. β-sheet topology and the relatedness of proteins. *Nature*, 268:495–500, 1977.

[13] K. Koniaris and M. Muthukumar. Knottedness in ring polymers. *Physical Review Letters*, 66:2211–2214, 1991.

[14] W. R. Taylor. A deeply knotted protein structure and how it might fold. *Nature*, 406:916–919, 2000.

[15] M. Jamroz, W. Niemyska, E. J. Rawdon, A. Stasiak, K. C. Millett, P. Sulkowski, and J. I. Sulkowska. Knotprot: a database of proteins with knots and slipknots. *Nucleic Acids Research*, 43:D306–D314, 2015.

[16] D. Bölinger, J. I. Sulkowska, H.-P. Hsu, L. A. Mirny, M. Kardar, J. N. Onuchic, and P. Virnau. A stevedore's protein knot. *PLoS Comput. Biol.*, 6(4):e1000731, 2010.

[17] P. F. N. Faísca, R. D. M. Travasso, T. Charters, A. Nunes, and M. Cieplak. The folding of knotted proteins: insights from lattice simulations. *Physical Biology*, 7:016009, 2010.

[18] M. A. Soler, A. Nunes, and P. F. N. Faísca. Effects of knot type in the folding of topologically complex lattice proteins. *The Journal of Chemical Physics*, 141:025101, 2014.

[19] S. a Beccara, T. Škrbic, R. Covino, C. Micheletti, and P. Faccioli. Folding pathways of a knotted protein with a realistic atomistic force field. *PLoS Comput. Biol.*, 9:e1003002, 2013.

[20] J. K. Noel, J. N. Onuchic, and J. I. Sulkowska. Knotting a protein in explicit solvent. *The Journal of Physical Chemistry Letters*, 4:3570–3573, 2013.

[21] P. F. N. Faísca. Knotted proteins: A tangled tale of structural biology. *Computational and Structural Biotechnology Journal*, 13:459–468, 2015.

[22] F. I. Andersson, D. G. Pina, A. L. Mallam, G. Blaser, and S. E. Jackson. Untangling the folding mechanism of the 5_2-knotted protein uch-l3. *FEBS Journal*, 276:2625–2635, 2009.

[23] F. Ziegler, N. C. H. Lim, S. S. Mandal, B. Pelz, W.-P. Ng, M. Schlierf, S. E. Jackson, and M. Rief. Knotting and unknotting of a protein in single molecule experiments. *Proceedings of the National Academy of Sciences*, 113:7533–7538, 2016.

[24] Y. Zhao, P. Dabrowski-Tumanski, S. Niewieczerzal, and J. I. Sulkowska. The exclusive effects of chaperonin on the behavior of proteins with 5_2 knot. *PLoS Comput. Biol.*, 14(3):e1005970, 2018.

[25] M. A. Soler and P. F. N. Faísca. Effects of knots on protein folding properties. *PLoS ONE*, 8:e74755, 2013.

[26] A. L. Mallam and S. E. Jackson. Knot formation in newly translated proteins is spontaneous and accelerated by chaperonins. *Nat. Chem. Biol.*, 8:147–153, 2012.

[27] N. C. H. Lim and S. E. Jackson. Mechanistic insights into the folding of knotted proteins in vitro and in vivo. *Journal of Molecular Biology*, 427:248–258, 2015.

[28] K. A. Sharp. Analysis of the size dependence of macromolecular crowding shows that smaller is better. *Proc. Natl. Acad. Sci. USA*, 112:7990–7995, 2015.

[29] F. U. Hartl, A. Bracher, and M. Hayer-Hartl. Molecular chaperones in protein folding and proteostasis. *Nature*, 475:324–332, 2011.

[30] Z. H. Xu, A. L. Horwich, and P. B. Sigler. The crystal structure of the asymmetric groel-groes-$(adp)_7$ chaperonin complex. *Nature*, 388:741–750, 1997.

[31] R. J. Ellis. Molecular chaperones: Inside and outside the anfinsen cage. *Current Biology*, 11:R1038–R1040, 2001.

[32] A. C. Apetri and A. L. Horwich. Chaperonin chamber accelerates protein folding through passive action of preventing aggregation. *Proc. Natl. Acad. Sci. USA*, 105(45):17351–17355, 2008.

[33] M. J. Todd, G. H. Lorimer, and D. Thirumalai. Chaperonin-facilitated protein folding: Optimization of rate and yield by an iterative annealing mechanism. *Proc. Natl. Acad. Sci. USA*, 93:4030–4035, 1996.

[34] G. Stan, D. Thirumalai, G. H. Lorimer, and Brooks B. R. Annealing function of groel: structural and bioinformatic analysis. *Biophysical Chemistry*, 100:453–467, 2003.

[35] D. Thirumalai and C. Hyeon. Signalling networks and dynamics of allosteric transitions in bacterial chaperonin groel: implications for iterative annealing of misfolded proteins. *Phil. Trans. R. Soc. B*, 373:20170182, 2018.

[36] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953.

[37] Y. Sugita and Y. Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chem. Phys. Lett.*, 314:141–151, 1999.

[38] J. D. Chodera, W. C. Swope, J. W. Pitera, C. Seok, and K. A. Dill. Use of the weighted histogram analysis method for the analysis of simulated and parallel tempering simulations. *J. Chem. Theory and Computation*, 3:26–41, 2007.

[39] W. J. C. Orr. Statistical treatment of polymer solutions at infinite dilution. *Trans. Faraday Soc.*, 43:12–27, 1947.

[40] P.-G. de Gennes. *Scaling concepts in polymer physics*. Cornell University Press, Ithaca, U.S.A., 1979.

[41] D. W. Sumners and S. G. Whittington. Knots in self-avoiding walks. *J. Phys. A: Math. Gen.*, 21:1689–1694, 1988.

[42] H. Taketomi, Y. Ueda, and N. Gō. Studies on protein folding, unfolding and fluctuations by computer simulation. i. the effect of specific amino acid sequence represented by specific inter-unit interactions. *Int. J. Pept. Protein Res.*, 7:445–459, 1975.

[43] J. D. Bryngelson and P. G. Wolynes. Spin glasses and the statistical mechanics of protein folding. *Proc. Natl. Acad. Sci. U.S.A.*, 84:7524–7528, 1987.

[44] H. S. Chan. Protein folding: Matching speed and locality. *Nature*, 392:761–763, 1998.

[45] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[46] R. H. Swendsen and J.-S. Wang. Replica monte carlo simulation of spin-glasses. *Phys. Rev. Lett.*, 57:2607–2609, 1986.

[47] K. Hukushima and K. Nemoto. Exchange monte carlo method and application to spin glass simulations. *J. Phys. Soc. Japan*, 65:1604–1608, 1996.

[48] P. H. Verdier and W. H. Stockmayer. Monte carlo calculations on the dynamics of polymers in dilute solution. *J. Chem. Phys.*, 36:227–235, 1962.

[49] M. Lax and C. Brender. Monte carlo study of lattice polymer dynamics. *J. Chem. Phys.*, 67:1785–1787, 1977.

[50] S. Miyazawa and R. L. Jernigan. Estimation of effective interresidue contact energies from protein crystal structures: Quasi-chemical approximation. *Macromolecules*, 18:534–552, 1985.

[51] S. Miyazawa and R. L. Jernigan. Residue-residue potentials with a favorable contact pair term and an unfavorable high packing density term, for simulation and threading. *J. Mol. Biol.*, 256:623–644, 1996.

[52] A. M. Ferrenberg and R. H. Swendsen. Optimized monte carlo data analysis. *Phys. Rev. Lett.*, 63:1195–1198, 1989.

[53] S. Kumar, D. Bouzida, R. H. Swendsen, P. A. Kollman, and J. M. Rosenberg. The weighted histogram analysis method for free-energy calculations on biomolecules. i. the method. *J. Comput. Chem.*, 13:1011–1021, 1992.

[54] Glen Cowan. *Statistical Data Analysis*. Clarendon Press, Oxford, 1998.

[55] R. P. Brent. An algorithm with guaranteed convergence for finding a zero of a function. *Computer Journal*, 14:422–425, 1971.

[56] J. C. P. Bus and T. J. Dekker. Two efficient algorithms with guaranteed convergence for finding a zero of a function. *ACM Transactions of Mathematical Software*, 1(4):330–345, 1975.

[57] Dusan Bratko, Troy Cellmer, John M. Prausnitz, and Harvey W. Blanch. Effects of single-point sequence alterations on the aggregation propensity of a model protein. *J. Am. Chem. Soc.*, 128:1683–1691, 2006.