ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΥΓΕΙΑΣ
ΤΜΗΜΑ ΜΟΡΙΑΚΗΣ ΒΙΟΛΟΓΙΑΣ ΚΑΙ ΓΕΝΕΤΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ακρίβεια Προσομοιώσεων Αναδίπλωσης για οριακά σταθερά πεπτίδια: Εφαρμογή στο πεπτίδιο HP21 της πρωτεΐνης villin»

Αθανάσιος Μπαλτζής (ΑΕΜ: 1119)

Επίβλεψη:

Δρ. Νικόλαος Μ. Γλυκός

Επίκουρος Καθηγητής Υπολογιστικής και Δομικής Βιολογίας

Τμήμα Μοριακής Βιολογίας και Γενετικής

Δημοκρίτειο Πανεπιστήμιο Θράκης

Αλεξανδρούπολη, Ιούλιος 2015

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Νικόλαο Γλυκό, ο οποίος έπαιξε καταλυτικό ρόλο στην επίτευξη της διπλωματικής αυτής εργασίας αφενός με την καθοδήγηση και τις συμβουλές του και αφετέρου με τη μεταλαμπάδευση γνώσεων και ιδεών.

Επιπλέον, θα ήθελα να αναφερθώ στην οικογένεια μου, η οποία στάθηκε αρωγός καθ' όλη τη διάρκεια των σπουδών μου ενθαρρύνοντας και στηρίζοντας έμπρακτα κάθε μου προσπάθεια.

Τέλος, ευχαριστώ τους φίλους μου για τη διαρκή συμπαράσταση καθώς και τους συναδέλφους μου στο εργαστήριο για το ευχάριστο εργασιακό περιβάλλον.

Περιεχόμενα

Περίληψη

Οι προσομοιώσεις μοριακής δυναμικής είναι μια ευρέως χρησιμοποιούμενη μέθοδος για την κατανόηση της διαδικασίας αναδίπλωσης των πρωτεϊνών. Στην εργασία αυτή εξετάζουμε την ικανότητα της μεθόδου αυτής να αναπαράγει τα πειραματικά ευρήματα για οριακά σταθερά πεπτίδια. Συγκεκριμένα, πραγματοποιήσαμε μία προσομοίωση αναδίπλωσης μεγάλης διάρκειας (15 μs) με το δυναμικό πεδίο AMBER99SB-STAR-ILDN για το πεπτίδιο HP21, το οποίο προέρχεται από την υποεπικράτεια HP36 της πρωτεΐνης villin και, όπως έχει αποδειχθεί με ανάλυση χημικών μετατοπίσεων από πειράματα NMR, υιοθετεί μία δομή παρόμοια με την φυσική του διαμόρφωση παρά την ασταθή του φύση. Τα αποτελέσματα από την ανάλυση της προσομοίωσης επισημαίνουν την οριακή σταθερότητα του HP21, καθώς μόνο ένα μικρό ποσοστό του τροχιακού αντιστοιχεί σε γεγονότα αναδίπλωσης. Η σταθερότερη ομάδα διαμορφώσεων αντιπροσωπεύει μια δομή που μοιάζει με τα αντίστοιχα τμήματα των πειραματικά προσδιορισμένων δομών της HP36 και της επικράτειας HP. Η στατιστικά σημαντική συσχέτιση μεταξύ των πειραματικών δευτεροταγών χημικών μετατοπίσεων και των αντίστοιχων της προσομοίωσης για τα άτομα $^{13}C^{\alpha}$, $^{13}C^{\beta}$, ^{13}CO επιβεβαιώνει τον παραπάνω ισχυρισμό. Τέλος, η πολύ καλή συμφωνία μεταξύ πειράματος και προσομοίωσης γίνεται φανερή και μέσα από τη σύγκλιση των τιμών της ροπής δευτεροταγούς δομής.

Abstract

Molecular dynamics simulations are a widely used method for the understanding of protein folding process. In this project, we examine the ability of this method to reproduce the experimental findings for marginally stable peptides. In more detail, we performed a 15 μs long folding simulation with the AMBER99SB-STAR-ILDN forcefield for the HP21 peptide, which is derived from the HP36 villin headpiece subdomain and, as it is proven by the analysis of NMR-derived chemical shifts, adopts a native-like structure in spite of its unstable nature. The results from the analysis of simulation point out the marginal stability of HP21, as only a small percentage of the whole trajectory corresponds to folding events. The most stable conformer represents a structure quite similar to the corresponding fragments of experimental structures of HP36 and villin headpiece domain HP. The statistically significant correlation between the experimental and the simulation-derived $\Delta\delta^{13}C^{\alpha}$, $\Delta\delta^{13}C^{\beta}$ and $\Delta\delta^{13}CO$ secondary shifts confirms the aforementioned claim. Finally, the very good agreement between experiment and simulation is indicated by the convergence of secondary structure propensity scores.

Κεφάλαιο 1: Εισαγωγή

1.1 Αναδίπλωση των πρωτεϊνών

Οι πρωτεΐνες αποτελούν αναμφίβολα μία κατηγορία βιομορίων υψίστης σημασίας για την εύρυθμη λειτουργία των κυττάρων και, κατά συνέπεια, των οργανισμών, καθώς συνεισφέρουν δομικά και λειτουργικά σε ένα πλήθος βιολογικών διεργασιών. Δομικός λίθος των πρωτεϊνών είναι τα αμινοξέα, τα οποία ενώνονται μεταξύ τους μέσω πεπτιδικών δεσμών σχηματίζοντας πολυπεπτιδικές αλυσίδες (πρωτοταγής δομή), οι οποίες με τη σειρά τους αναδιπλώνονται στο χώρο παράγοντας τρισδιάστατες δομές (τριτοταγής δομή). Με αυτήν την διαδικασία που ονομάζεται πρωτεϊνική αναδίπλωση η κάθε πρωτεΐνη αποκτά τη φυσική της διαμόρφωση (native state), γεγονός που την καθιστά ικανή να επιτελέσει το βιολογικό της ρόλο [1].

Όπως γίνεται αντιληπτό, η κατανόηση των μηχανισμών που οδηγούν στην πρωτεϊνική αναδίπλωση, ή με άλλα λόγια ο τρόπος με τον οποίο η πρωτοταγής δομή μιας πρωτεΐνης καθορίζει την τριτοταγή της δομή, κρίνεται αναγκαία για την πλήρη αποσαφήνιση της λειτουργίας των πρωτεϊνών. Ωστόσο, η εντατική έρευνα στο συγκεκριμένο πεδίο εδώ και 50 περίπου χρόνια δεν έχει αποδώσει μία καθολικά αποδεκτή ερμηνεία, με αποτέλεσμα το περίφημο “πρόβλημα της αναδίπλωσης των πρωτεϊνών” (protein folding problem) να παραμένει ως ένα από τα βασικότερα άλυτα προβλήματα της Μοριακής Βιολογίας.

1.2 Ιστορική αναδρομή και μοντέλα αναδίπλωσης πρωτεϊνών

Οι έννοιες της αναδίπλωσης και αποδιάταξης των πολυπεπτιδικών αλυσίδων ήταν ήδη γνωστές στην επιστημονική κοινότητα πριν περίπου 80 χρόνια [2]. Παρολ' αυτά σημαντική ερευνητική δραστηριότητα σημειώθηκε μετά τη δημοσίευση των ευρημάτων του Christian P. Anfinsen [3,4] και του Cyrus Levinthal [5,6].

Πιο αναλυτικά, ο πρώτος, βασισμένος σε πειράματα αποδιάταξης και επαναδιάταξης του ενζύμου ριβονουκλεάση, διατύπωσε το 1973 την "θερμοδυναμική υπόθεση", σύμφωνα με την οποία η φυσική διαμόρφωση μιας πρωτεΐνης σε φυσιολογικές συνθήκες διαλύματος είναι αυτή για την οποία ελαχιστοποιείται η τιμή της ελεύθερης ενέργειας Gibbs ολόκληρου του συστήματος. Κατά συνέπεια, η φυσική δομή μιας πρωτεΐνης καθορίζεται από το σύνολο των διατομικών αλληλεπιδράσεων, δηλαδή από την αλληλουχία των αμινοξέων της [3].

Ωστόσο, πέραν της θερμοδυναμικής παραμέτρου, θα πρέπει επίσης να ληφθεί υπόψη και η κινητική παράμετρος. Ο Levinthal στην προσπάθεια του να ερμηνεύσει τους παράγοντες που καθορίζουν την ταχύτητα της πρωτεϊνικής αναδίπλωσης διαπίστωσε ότι ο σχηματισμός της σταθερότερης θερμοδυναμικά στερεοδιαμόρφωσης μιας πρωτεΐνης είναι αδύνατο να συμβαίνει μέσα από τυχαίες μετατοπίσεις (παράδοξο του Levinthal). Εάν υποθέσουμε πως μια πρωτεΐνη αποτελείται από 100 αμινοξέα και κάθε αμινοξύ μπορεί να υιοθετήσει δύο διαφορετικές στερεοδιαμορφώσεις, τότε συνολικά υπάρχουν 10^{30} διαμορφώσεις από τις οποίες θα πρέπει να μεταβεί η πρωτεΐνη ώστε να βρει τη φυσική της δομή. Ο χρόνος που θα χρειαζόταν για να πραγματοποιηθούν όλες οι πιθανές μεταβάσεις είναι περίπου 10^{10} χρόνια! Επομένως, η διαδικασία αναδίπλωσης των πρωτεϊνών απαιτεί την ύπαρξη μιας αλληλουχίας καθορισμένων μονοπατιών, τα οποία διευκολύνουν την εύρεση της διαμόρφωσης εκείνης που αντιστοιχεί στο ολικό ενεργειακό ελάχιστο σε λίγα μόνο δευτερόλεπτα [5,6].

Με βάση τα δύο παραπάνω θεμελιώδη ευρήματα, διενεργήθηκαν πολλές πειραματικές μελέτες με στόχο την εύρεση ενδιάμεσων καταστάσεων στη διαδικασία της αναδίπλωσης και προτάθηκε ένα πλήθος μοντέλων που επιχειρούν να ερμηνεύσουν τον μηχανισμό αναδίπλωσης των πρωτεϊνών με διαφορετικούς τρόπους, δίχως, όμως, κάποιο από αυτά να έχει καθολική ισχύ [7,8,9,10].

Το μοντέλο "*nucleation/growth*" [11] προτείνει έναν μηχανισμό αναδίπλωσης τριών σταδίων. Αρχικά, σχηματίζονται τοπικά δευτεροταγείς δομές, όπως α-έλικες και β-πτυχώτα φύλλα, οι οποίες σε επόμενο στάδιο σταθεροποιούνται για τη δημιουργία ενός συμπαγούς πυρήνα. Τα υπόλοιπα μη αναδιπλωμένα τμήματα της πολυπεπτιδικής αλυσίδας αναδιπλώνονται με κατάλληλο τρόπο γύρω από τον πυρήνα ώστε να προκύψει η φυσική διαμόρφωση. Συνεπώς, ο σχηματισμός του πυρήνα είναι το γεγονός εκείνο που αυξάνει την ταχύτητα της αναδίπλωσης.

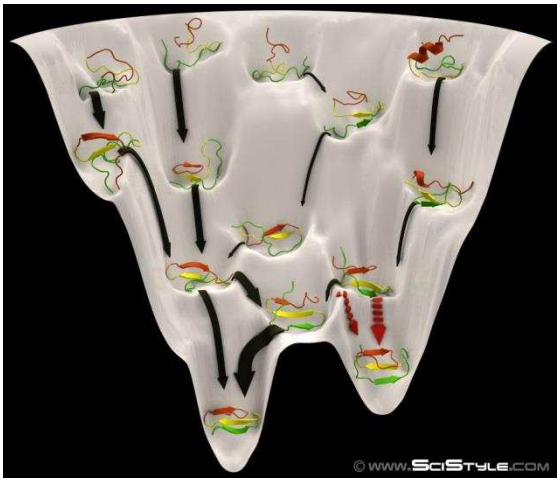
Σύμφωνα με το μοντέλο "*diffusion-collision*" [12,13,14], μικροεπικράτειες, αποτελούμενες είτε από τμήματα δευτεροταγών δομών είτε από υδρόφοβες ομάδες, συγκρούονται επαναληπτικά μεταξύ τους λόγω διαρκούς κίνησης οδηγώντας στη συναρμολόγηση μεγαλύτερων δομικών μονάδων.

Το μοντέλο "*nucleation-condensation*" [15,16] αποτελεί μια προσπάθεια ενοποίησης των μοντέλων "*framework*" [17] και "*hydrophobic collapse*" [18]. Πιο αναλυτικά, προτείνει ότι ο σχηματισμός στοιχείων δευτεροταγούς δομής συμβαίνει ταυτόχρονα, και όχι ανεξάρτητα σύμφωνα με τα δύο προγενέστερα μοντέλα, με τον σχηματισμό δομών λόγω υδρόφοβων αλληλεπιδράσεων γεγονός που σταθεροποιεί τις ενδιάμεσες καταστάσεις και επιταχύνει την διαδικασία της αναδίπλωσης [19].

Τέλος, το μοντέλο "*jigsaw puzzle*" [20] αναφέρεται στην δυνατότητα κάθε πρωτεϊνικού μορίου να ακολουθεί μία μοναδική διαδρομή αναδίπλωσης προς τη φυσική του δομή, χωρίς να είναι απαραίτητη η ύπαρξη ενός κοινού μονοπατιού αναδίπλωσης για όλες τις πρωτεΐνες. Η παραπάνω θεώρηση άνοιξε το δρόμο για την επικράτηση ενός νέου μοντέλου τα τελευταία

χρόνια, το οποίο ονομάστηκε μοντέλο των *ενεργειακών τοπίων (energy landscape)* ή *χωνιών αναδίπλωσης (folding funnels)* [21].

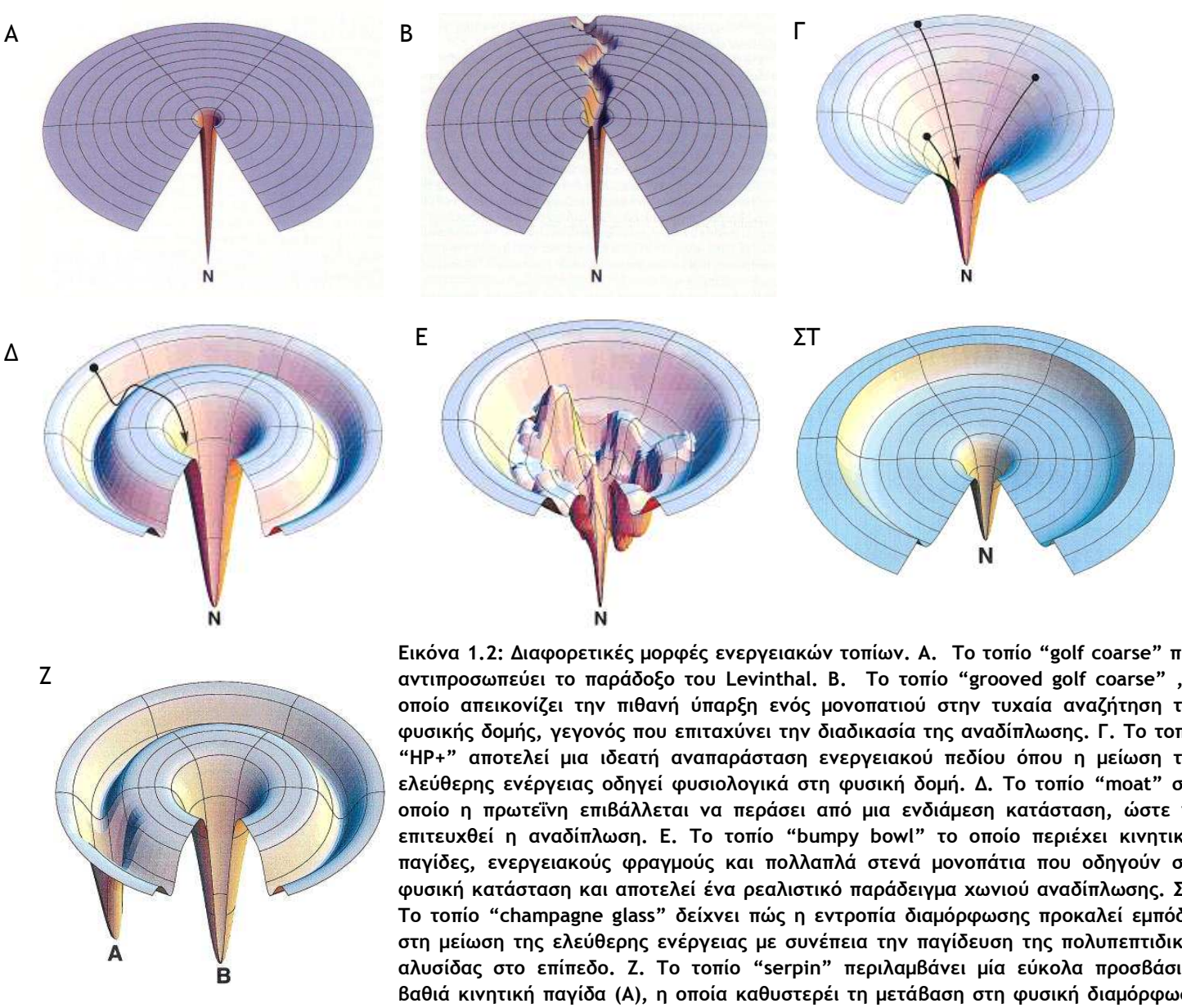
Τα *ενεργειακά τοπία* αποτελούν στατιστικές περιγραφές της ενέργειας μιας πρωτεΐνης. Σε αντίθεση με τα προγενέστερα μοντέλα που βασίζονται στην ιδέα ενός μονοπατιού διαδοχικών γεγονότων τα οποία οδηγούν στην αναδίπλωση, η νέα αυτή θεωρία προτείνει πως η αναδίπλωση είναι μια διαδικασία παράλληλων γεγονότων που μοιάζει με διάχυση και περιλαμβάνει πλήθος μονοπατιών.



Εικόνα 1.1: Σχηματική αναπαράσταση ενός χωνιού αναδίπλωσης

Κάθε *ενεργειακό τοπίο* μπορεί να παρασταθεί γραφικά σε πολυδιάστατα διαγράμματα που παρουσιάζουν τη διακύμανση της ελεύθερης ενέργειας κάθε στερεοδιαμόρφωσης ως συνάρτηση των βαθμών ελευθερίας (**Εικόνα 1.1**). Ο κάθετος άξονας αναπαριστά την 'εσωτερική' ελεύθερη ενέργεια, ενώ οι πλευρικοί άξονες τις συντεταγμένες των διάφορων διαμορφώσεων. Κάθε πιθανή στερεοδιαμόρφωση είναι ένα σημείο του *χωνιού αναδίπλωσης*, η μορφή του οποίου μοιάζει με κορυφές και κοιλάδες. Οι στερεοδιαμορφώσεις με υψηλή ενέργεια (μη ευνοϊκές θερμοδυναμικά) εντοπίζονται στις κορυφές, ενώ εκείνες με χαμηλή ενέργεια (ευνοϊκές θερμοδυναμικά) στις κοιλάδες. Η στερεοδιαμόρφωση που αντιστοιχεί στη φυσική δομή βρίσκεται στη βάση του χωνιού. Επομένως, η κινητική διαδικασία της αναδίπλωσης μπορεί να παρομοιαστεί με την κίνηση μιας μπάλας μέσα στο *ενεργειακό τοπίο*: η πρωτεΐνη "κυλάει" στο *ενεργειακό τοπίο* διαπερνόντας μέσα από κορυφές και κοιλάδες μέχρι να φτάσει στη

βάση του (φυσική δομή) δημιουργώντας ένα τροχιακό. Η **Εικόνα 1.2** αναπαριστά διαφορετικές μορφές *ενεργειακών τοπίων* [22,23].



Εικόνα 1.2: Διαφορετικές μορφές ενεργειακών τοπίων. Α. Το τοπίο "golf coarse" που αντιπροσωπεύει το παράδοξο του Levinthal. Β. Το τοπίο "grooved golf coarse" ,το οποίο απεικονίζει την πιθανή ύπαρξη ενός μονοπατιού στην τυχαία αναζήτηση της φυσικής δομής, γεγονός που επιταχύνει την διαδικασία της αναδίπλωσης. Γ. Το τοπίο "HP+" αποτελεί μια ιδεατή αναπαράσταση ενεργειακού πεδίου όπου η μείωση της ελεύθερης ενέργειας οδηγεί φυσιολογικά στη φυσική δομή. Δ. Το τοπίο "moat" στο οποίο η πρωτεΐνη επιβάλλεται να περάσει από μια ενδιάμεση κατάσταση, ώστε να επιτευχθεί η αναδίπλωση. Ε. Το τοπίο "bumpy bowl" το οποίο περιέχει κινητικές παγίδες, ενεργειακούς φραγμούς και πολλαπλά στενά μονοπάτια που οδηγούν στη φυσική κατάσταση και αποτελεί ένα ρεαλιστικό παράδειγμα χωνιού αναδίπλωσης. ΣΤ. Το τοπίο "champagne glass" δείχνει πώς η εντροπία διαμόρφωσης προκαλεί εμπόδια στη μείωση της ελεύθερης ενέργειας με συνέπεια την παγίδευση της πολυπεπτιδικής αλυσίδας στο επίπεδο. Ζ. Το τοπίο "serpin" περιλαμβάνει μία εύκολα προσβάσιμη βαθιά κινητική παγίδα (Α), η οποία καθυστερεί τη μετάβαση στη φυσική διαμόρφωση μέσω του μονοπατιού (Β). (Οι εικόνες αναπαράγονται άνευ αδείας [22,23])

1.3 Μελέτη της αναδίπλωσης των πρωτεϊνών

1.3.1 Πειραματικές προσεγγίσεις

Οι πειραματικές μέθοδοι αποτελούν αναμφισβήτητα τις πιο αξιόπιστες προσεγγίσεις για τη μελέτη της δομής και της αναδίπλωσης των πρωτεϊνικών μορίων. Έως σήμερα οι πειραματικές τεχνικές έχουν συνεισφέρει σε μεγάλο βαθμό στην αποκάλυψη δομικών πληροφοριών για πληθώρα πρωτεϊνών και πεπτιδίων [24,25]. Η κρυσταλλογραφία ακτινών Χ είναι η σημαντικότερη πειραματική μέθοδος και βρίσκει εφαρμογή στον προσδιορισμό της δευτεροταγούς και τριτοταγούς δομής πρωτεϊνών με την προϋπόθεση να μπορούν να σχηματίσουν καλά οργανωμένους κρυστάλλους, οι οποίοι θα επιτρέπουν την περίθλαση ακτινών Χ [1]. Η μέθοδος του πυρηνικού μαγνητικού συντονισμού (NMR) συνιστά μία εξίσου αξιόπιστη και ευρέως χρησιμοποιούμενη τεχνική παρέχοντας πολύτιμα δεδομένα σχετικά με τις αλληλεπιδράσεις μεταξύ των ατόμων των βιομορίων [26]. Τέλος, αξίζει να γίνει αναφορά σε ορισμένες επιπλέον πειραματικές μεθόδους, όπως ο κυκλικός διχρωϊσμός (CD) για την εύρεση στοιχείων δευτεροταγούς δομής [26,27,28], η στοχευόμενη μεταλλαξιγένεση μέσω της πρωτεϊνικής μηχανικής [26], η φασματομετρία μάζας [29], η μικροσκοπία ατομικής μάζας (AFM), η μέθοδος small-angle x-ray scattering (SAXS) [30] καθώς και η φασματοσκοπία FT-IR (Fourier transform infrared) [31].

Επειδή τα αποτελέσματα της μεθόδου NMR, και συγκεκριμένα οι χημικές μετατοπίσεις, χρησιμοποιήθηκαν ευρύτατα στην παρούσα εργασία για τον έλεγχο της ακρίβειας της προσομοίωσης αναδίπλωσης σε σχέση με τα πειραματικά δεδομένα, κρίνεται αναγκαίο να επισημανθούν οι βασικές αρχές που διέπουν την τεχνική αυτή.

Αρχικά, ορισμένοι πυρήνες, όπως ^{1}H, ^{13}C και ^{15}N, που έχουν ιδιοστροφορμή (spin) μπορούν να συμπεριφερθούν σαν μαγνήτες (μαγνητική ροπή). Σύμφωνα με την κβαντομηχανική, το spin ενός πυρήνα είναι κβαντισμένο, δηλαδή δέχεται ορισμένες μόνο τιμές. Στην περίπτωση που ο πυρήνας αυτός

έχει spin διάφορο του μηδενός, μπορεί να αλληλεπιδράσει με ένα εξωτερικό μαγνητικό πεδίο. Με βάση την αλληλεπίδραση αυτή, τόσο το spin όσο και η μαγνητική ροπή μπορούν να προσανατολιστούν είτε προς είτε ενάντια στο εξωτερικό μαγνητικό πεδίο. Ο προσανατολισμός προς το εξωτερικό πεδίο είναι σταθερότερος και, συνεπώς, απαιτείται απορρόφηση ενέργειας με τη μορφή ακτινοβολίας από τον πυρήνα, ώστε να αλλάξει προσανατολισμό. Η ποσότητα της ενέργειας αυτής εξαρτάται από την ένταση του εξωτερικού πεδίου.

Ωστόσο, η συχνότητα ακτινοβόλησης για την οποία συμβαίνει απορρόφηση ενέργειας δεν είναι η ίδια για όλους τους πυρήνες ενός μορίου, γεγονός που οφείλεται στην διαφορετική ένταση εφαρμοζόμενου μαγνητικού πεδίου που δέχεται ο κάθε πυρήνας ως απόρροια του περιβάλλοντος στο οποίο βρίσκεται. Επομένως, διατηρώντας σταθερή τη συχνότητα ακτινοβόλησης και μεταβάλλοντας την ένταση του εξωτερικού μαγνητικού πεδίου, παράγεται ένα φάσμα NMR, το οποίο αποτελείται από πολλές κορυφές απορρόφησης, οι σχετικές θέσεις των οποίων περιέχουν πληροφορίες σχετικά με τη δομή του μορίου.

Ειδικότερα, οι θέσεις των σημάτων απορρόφησης αντικατοπτρίζουν το ηλεκτρονιακό περιβάλλον του κάθε πυρήνα. Τα ηλεκτρόνια λόγω της εφαρμογής του εξωτερικού μαγνητικού πεδίου δημιουργούν με τη σειρά τους δευτερογενή μαγνητικά πεδία. Τα δευτερογενή αυτά μαγνητικά πεδία μπορούν να έχουν είτε ίδιο είτε αντίθετο προσανατολισμό σε σχέση με το εφαρμοζόμενο εξωτερικό μαγνητικό πεδίο. Στην πρώτη περίπτωση ενισχύουν το εφαρμοζόμενο πεδίο και ο πυρήνας θεωρείται αποπροστατευμένος, ενώ στη δεύτερη περίπτωση αντιτίθεται στο εφαρμοζόμενο πεδίο και ο πυρήνας θεωρείται προστατευμένος. Το σήμα απορρόφησης ενός προστατευμένου πυρήνα μετατοπίζεται ανοδικά (upfield), ενώ το αντίστοιχο σήμα ενός αποπροστατευμένου πυρήνα καθοδικά (downfield). Συμπερασματικά, οι μετατοπίσεις αυτές στη θέση απορρόφησης σε ένα φάσμα NMR που προέρχονται από την προστασία ή όχι του πυρήνα από τα ηλεκτρόνια ονομάζονται χημικές μετατοπίσεις (chemical shifts) [32,33].

Οι χημικές μετατοπίσεις αποτελούν εδώ και περίπου 60 χρόνια ένα πολύτιμο εργαλείο στα χέρια των ερευνητών για την αποκρυπτογράφηση δομικής πληροφορίας σχετικά με νέες ενώσεις, και ειδικότερα με πρωτεϊνικά μόρια [34]. Πιο συγκεκριμένα, η χρησιμότητά τους έγκειται στον καθορισμό του είδους και της θέσης δευτεροταγών δομών μέσα στην πολυπεπτιδική αλυσίδα [35,36], στην ταξινόμηση των πρωτεϊνών [37], στην μέτρηση της προσβασιμότητας επιφανειακών περιοχών [38], στην ακριβή μέτρηση των γωνιών της κύριας αλυσίδας (φ, ψ, ω) καθώς και των γωνιών συστροφής των πλευρικών ομάδων (x1, x2) [39], στον καθορισμό της ευελιξίας της κύριας αλυσίδας (RMSD, RMSF) [40] και, τέλος, ακόμη και στην παραγωγή τρισδιάστατης δομής με μεγάλη ακρίβεια [41].

Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι δευτεροταγείς χημικές μετατοπίσεις (Secondary Chemical Shifts – Δδ Shifts), διότι περιέχουν το μεγαλύτερο μέρος της πληροφορίας που αφορά τις μη ομοιοπολικές αλληλεπιδράσεις και τη δυναμική μιας πρωτεΐνης και γι' αυτό το λόγο μπορούν να αποκαλύψουν τη ροπή καθενός καταλοίπου προς ένα συγκεκριμένο είδος δευτεροταγούς δομής (α-έλικες, β-πτυχωτά φύλλα, τυχαίο σπείραμα) [35]. Ο υπολογισμός των τιμών των δευτεροταγών χημικών μετατοπίσεων γίνεται με την αφαίρεση μεταξύ της παρατηρούμενης τιμής της χημικής μετατόπισης (δ_{obs}) και της τιμής τυχαίου σπειράματος (δ_{rc}) για το αντίστοιχο άτομο του αμινοξέως:

$$\Delta\delta = \delta_{obs} - \delta_{rc}$$

Ο όρος τυχαίο σπείραμα (random coil) αναφέρεται στην διαμόρφωση εκείνη που υιοθετεί μια πολυπεπτιδική αλυσίδα όταν η διευθέτηση του κάθε αμινοξικού της καταλοίπου στο χώρο δεν επηρεάζεται από τα γειτονικά κατάλοιπα. Όπως γίνεται αντιληπτό, η ακρίβεια των τιμών των χημικών μετατοπίσεων των τυχαίων σπειραμάτων για κάθε αμινοξύ είναι αναγκαία για την εγκυρότητα της πληροφορίας που αντλείται από τις δευτεροταγείς χημικές μετατοπίσεις. Για το λόγο αυτό, έχουν διενεργηθεί πολλά πειράματα από τα τέλη της δεκαετίας του 1970 [42,43] έως και σήμερα με στόχο τη βελτίωση της ακρίβειας των τιμών αυτών [44,45].

1.3.2 Υπολογιστικές προσεγγίσεις

Οι πειραματικές προσεγγίσεις, παρά τα αξιόπιστα αποτελέσματα που παράγουν, εμφανίζουν μερικά μειονεκτήματα, τα οποία δυσχεραίνουν τη χρήση τους για τη μελέτη της αναδίπλωσης των πρωτεϊνικών δομών. Ορισμένα από αυτά αφορούν την αδυναμίας τους να αποδώσουν πολλαπλές διαμορφώσεις της δομής μια πρωτεΐνης κατά τη διάρκεια της αναδίπλωσης σε ατομικό επίπεδο καθώς επίσης και την αδυναμία εφαρμογής τους για την αποκάλυψη των τρισδιάστατων δομών όλων των κατηγοριών των πρωτεϊνών. Οι υπολογιστικές προσεγγίσεις επιχειρούν να καλύψουν τα παραπάνω κενά δρώντας ως συνδετικός κρίκος μεταξύ θεωρίας και πειραματικών τεχνικών με σκοπό την πρόβλεψη πρωτεϊνικών δομών και την κατανόηση των μηχανισμών που οδηγούν στην αναδίπλωση με μεγαλύτερη ακρίβεια και λεπτομέρεια. Προς αυτήν την κατεύθυνση συμβάλλουν επίσης η ολοένα και αυξανόμενη ταχύτητα και απόδοση των υπολογιστικών συστημάτων, η δημιουργία ελεύθερα προσβάσιμων βάσεων δεδομένων αλλά και υπολογιστικών εργαλείων και αλγορίθμων στα πλαίσια της ραγδαίας ανάπτυξης των επιστημονικών πεδίων της Βιοπληροφορικής και της Υπολογιστικής Βιολογίας.

Σε γενικές γραμμές, η πρόβλεψη της δομής μια πρωτεΐνης από την αμινοξική της αλληλουχία αγνοώντας την διαδικασία της φυσικής αναδίπλωσης βασίζεται σε εμπειρικές μεθόδους οι σημαντικότερες από τις οποίες είναι οι εξής [46]:

- Πρόβλεψη δευτεροταγούς δομής με τη χρήση νευρωνικών δικτύων [47]
- Μοντελοποίηση ομολόγων (Homology modeling), όπου γίνεται πρόβλεψη της τρισδιάστατης δομής μιας πρωτεΐνης από γνωστές δομές ομόλογων πρωτεϊνών με βάση τη στοίχιση των αμινοξικών αλληλουχιών [48]
- Αναγνώριση αναδίπλωσης (Threading ή Fold recognition), όπου η αναζήτηση κοινών μοτίβων αναδίπλωσης ανάμεσα σε

πρωτεΐνες με γνωστή δομή και την προς εξέταση πρωτεΐνη οδηγεί στην δημιουργία του τρισδιάστατου μοντέλου της [49]

- Πρόβλεψη δομών de novo με βάση την θερμοδυναμική υπόθεση [50] π.χ. η μέθοδος Rosetta [51]

Αξιοσημείωτο ρόλο στην πρόοδο των μεθόδων πρόβλεψης δομών κατέχει το κέντρο CASP (Critical Assessment of Structure Prediction) κυρίως μέσω του διαγωνισμού που διοργανώνει κάθε δύο χρόνια με αφετηρία το 1994 [52].

Σε αντίθεση με τις εμπειρικές μεθόδους που αναφέρθηκαν, οι φυσικές μέθοδοι (energy-based) αποτελούν μία εναλλακτική λύση, καθώς επιχειρούν να προβλέψουν μία δομή με τη χρήση συναρτήσεων που περιγράφουν την δυναμική ενέργεια του συστήματος λαμβάνοντας υπόψη την διαδικασία της αναδίπλωσης για την αναζήτηση της φυσικής δομής. Χαρακτηριστικό παράδειγμα της κατηγορίας αυτής είναι οι προσομοιώσεις μοριακής δυναμικής (molecular dynamics simulations), οι οποίες με τη βοήθεια των νόμων της Κλασσικής Μηχανικής και των δυναμικών πεδίων (force fields) μπορούν να αναπαραστήσουν ικανοποιητικά την διαδικασία της πρωτεϊνικής αναδίπλωσης [53]. Επίσης, η μελέτη μοριακών συστημάτων είναι εφικτή και μέσα από τις προσομοιώσεις Monte Carlo, οι οποίες στηρίζονται σε στατιστικές και πιθανολογικές μεθόδους για την εξαγωγή αποτελεσμάτων [54,55].

1.4 Χρήση πεπτιδίων για τη μελέτη της πρωτεϊνικής αναδίπλωσης

Ο όρος πεπτίδιο αναφέρεται σε μια μικρή πολυπεπτιδική αλυσίδα, η οποία δεν ξεπερνά σε μήκος τα 50 αμινοξικά κατάλοιπα. Με δεδομένο ότι η τρισδιάστατη δομή των πρωτεϊνών εμπεριέχει στοιχεία δευτεροταγούς δομής, ορισμένοι ερευνητές έχουν στρέψει το ενδιαφέρον τους στη χρήση πεπτιδίων, και όχι ολόκληρων πρωτεϊνών, ως μοντέλα για τη μελέτη της αναδίπλωσης θεωρώντας πως τοπικές περιοχές μιας πρωτεΐνης έχουν την τάση να σχηματίζουν δομές που μοιάζουν με αυτές της φυσικής τους διαμόρφωσης [56]. Η παραπάνω υπόθεση έχει επιβεβαιωθεί σε αρκετές μελέτες με τη χρήση είτε πειραματικών τεχνικών [57] είτε προσομοιώσεων [58].

Εκτός από τα παραπάνω, η χρήση των πεπτιδίων στη θέση των πρωτεϊνών εμφανίζει και περαιτέρω πρακτικά πλεονεκτήματα. Ένα πεπτίδιο λόγω του μικρού του μεγέθους και του σχετικά περιορισμένου αριθμού ατόμων που περιέχει αποτελεί ένα απλό σύστημα μελέτης, το οποίο δεν απαιτεί ιδιαίτερα αυξημένη υπολογιστική ισχύ ούτε κατανάλωση χρόνου αντίστοιχη με ένα πρωτεϊνικό σύστημα. Επίσης, τα αποτελέσματα χαρακτηρίζονται από χαμηλότερο βαθμό πολυπλοκότητας, γεγονός που ευνοεί την ευκολότερη ανάλυση και έχει ως συνέπεια την εξαγωγή περισσότερο αξιόπιστων συμπερασμάτων.

Τέλος, δεν θα μπορούσαμε να παραβλέψουμε τον ρόλο των πεπτιδίων στην βελτίωση των φυσικών μεθόδων πρόβλεψης της τριτοταγούς δομής και αναπαράστασης της διαδικασίας αναδίπλωσης. Ειδικότερα, όσον αφορά της προσομοιώσεις μοριακής δυναμικής, τα πεπτίδια χρησιμοποιούνται ευρέως για τον έλεγχο της εγκυρότητας των δυναμικών πεδίων (force fields) [59-61] με στόχο την διαρκή βελτίωσή τους, ώστε να ανταποκρίνονται όσο το δυνατόν περισσότερο στη φυσική πραγματικότητα.

1.5 Οριακή σταθερότητα και προέλευση του πεπτιδίου HP21

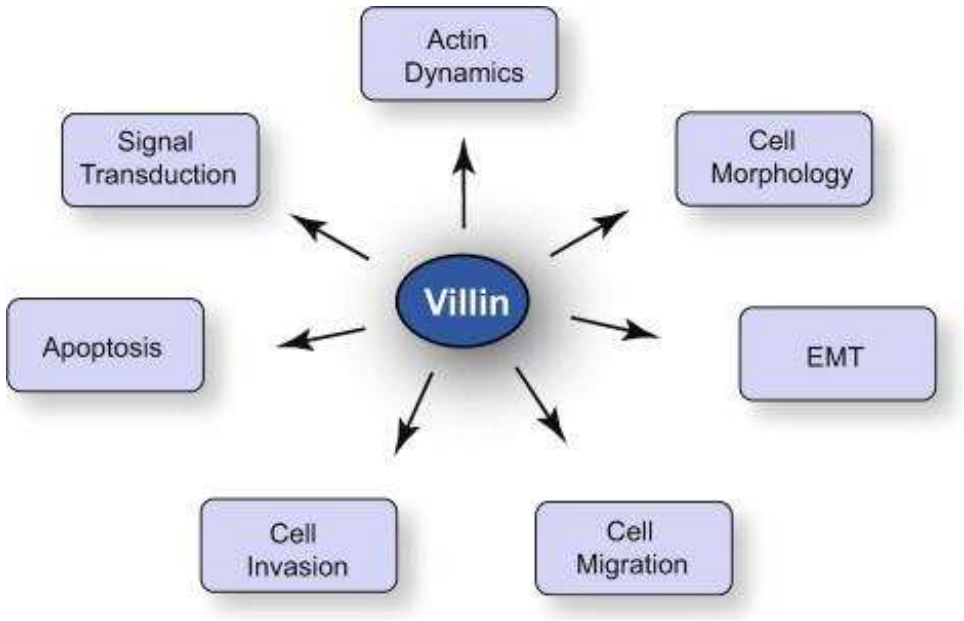
Η οριακή σταθερότητα είναι μία ιδιότητα που χαρακτηρίζει ένα πλήθος πολυπεπτιδικών μορίων, ειδικότερα σφαιρικών πρωτεϊνών, και συνίσταται στην αδυναμία σχηματισμού μιας υψηλά σταθερής φυσικής διαμόρφωσης κατά τη διαδικασία της αναδίπλωσης. Το γεγονός αυτό οφείλεται θερμοδυναμικά σε μειωμένη τιμή της διαφοράς της ελεύθερης ενέργειας αναδίπλωσης στη φυσική δομή $\Delta G_{folding}$ (από -5 έως -10 kcal/mol), το οποίο σημαίνει ότι η τιμή της ενέργειας κατά τη φυσική κατάσταση βρίσκεται πιο κοντά σε τιμές ενέργειας που αντιστοιχούν σε μη αναδιπλωμένες καταστάσεις. Συνεπώς, μια οριακά σταθερή πρωτεΐνη δεν διατηρεί την φυσική της διαμόρφωση αλλά έχει την τάση να αλλάζει διαρκώς μεταξύ μη αναδιπλωμένων διαμορφώσεων.

Οι οριακά σταθερές πρωτεΐνες μελετώνται σε μεγάλη κλίμακα τα τελευταία χρόνια, καθώς παρουσιάζει ιδιαίτερο ενδιαφέρον τόσο ο βιολογικός τους ρόλος μέσα στα κύτταρα όσο και ο τρόπος λειτουργίας τους. Ειδικότερα, ένα μέρος της επιστημονικής κοινότητας έχει καταλήξει στο συμπέρασμα πως η οριακή σταθερότητα ίσως αποτελεί ένα εξελικτικό πλεονέκτημα για την κατηγορία αυτή των βιομορίων σε σχέση με τις περισσότερο σταθερές πρωτεΐνες [62,63]. Οι βασικότεροι λόγοι για τους οποίους μπορεί να συμβαίνει το γεγονός αυτό είναι η αυξημένη λειτουργικότητα των οριακά σταθερών πρωτεϊνών λόγω της ευελιξίας που παρουσιάζουν [64] καθώς επίσης και η ενίσχυση της επιλεκτικότητας στο σχηματισμό αλληλεπιδράσεων με προσδέτες που οφείλεται στη συχνή εναλλαγή διαμορφώσεων [65].

Το πεπτίδιο HP21 που χρησιμοποιήθηκε ως μοντέλο στην παρούσα εργασία εμφανίζει οριακή σταθερότητα και προέρχεται από την πρωτεΐνη villin. Η villin είναι μία ιστοειδική πρωτεΐνη τροποποίησης της ακτίνης με μοριακό βάρος 92.5 kDa που συναντάται στα νημάτια ακτίνης στις μικρολάχνες και τον τερματικό ιστό των επιθηλιακών κυττάρων. Ανήκει σε μία

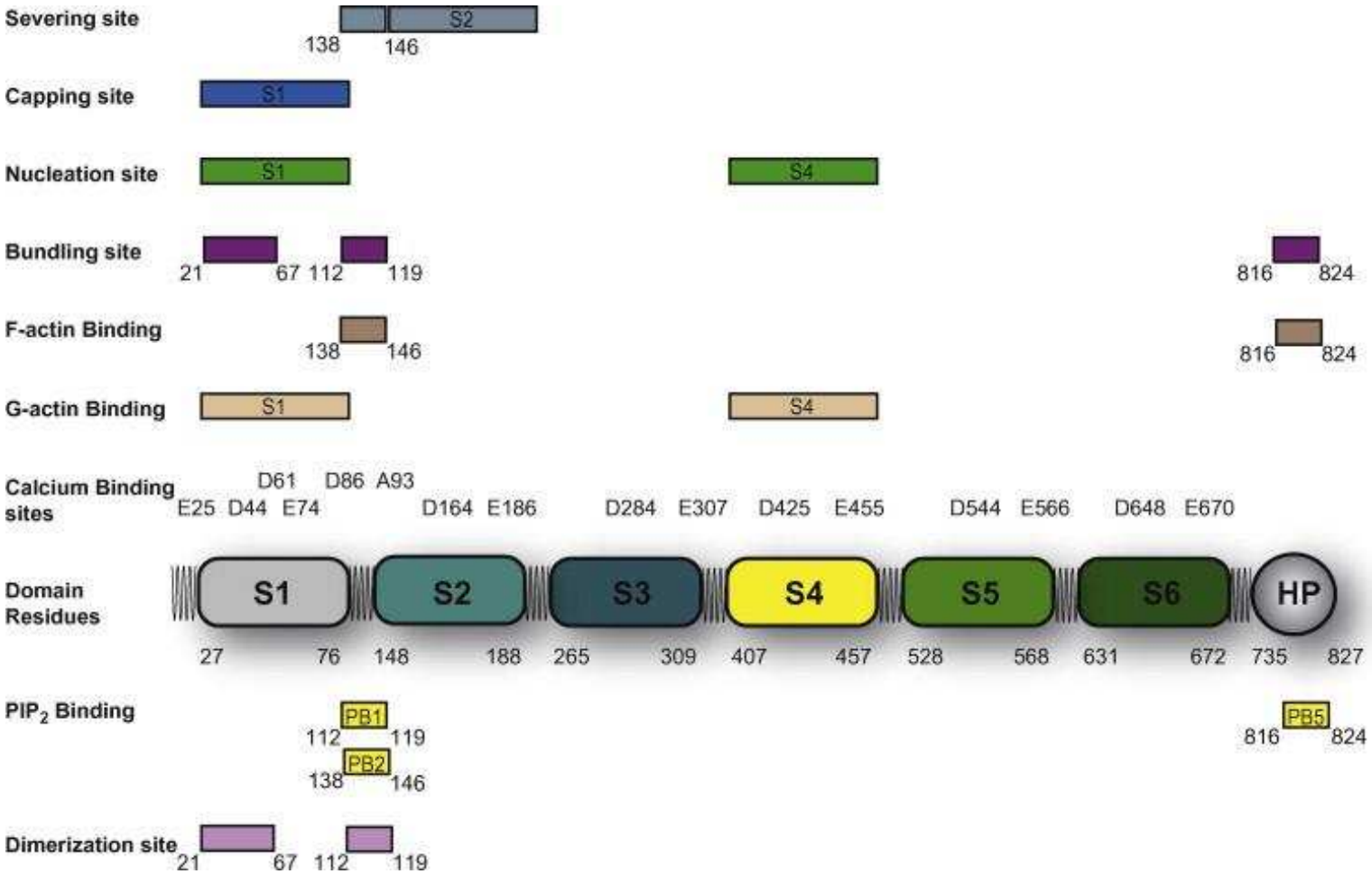
υπεροικογένεια πρωτεϊνών που προσδένονται στην ακτίνη συμπεριλαμβανομένων των gelsolin, severin, fragmin, adseverin/scinderin καθώς και πρωτεϊνών εγκάρσιας σύνδεσης με την ακτίνη όπως οι dematin και supervillin.

Η villin έχει αποδειχθεί πως συμμετέχει ενεργά σε ένα πλήθος βιολογικών διεργασιών εντός των επιθηλιακών κυττάρων κατέχοντας σπουδαίο ρόλο τόσο στη φυσιολογία όσο και στην παθοφυσιολογία των κυττάρων αυτών (**Εικόνα 1.3**) [66]. Συνοπτικά, έχει παρατηρηθεί εμπλοκή της στη δυναμική της ακτίνης ρυθμίζοντας τον πολυμερισμό και αποπολυμερισμό των νηματίων ακτίνης, στη μεταγωγή σημάτων, στον καθορισμό της μορφολογίας του κυττάρου, στη μετάβαση επιθηλιακού σε μεσεγχυματικό ιστό (EMT) κατά την εμβρυογένεση και την επούλωση πληγών, στην μετανάστευση κυττάρων, στην κυτταρική εισβολή καθώς και στην κυτταρική επιβίωση δρώντας ως αντι-αποπτωτικός παράγοντας. Επιπρόσθετα, η εισβολή και η διάδοση παθογόνων μικροοργανισμών του εντέρου κατά μήκος του γαστρεντερικού σωλήνα φαίνεται να εξαρτάται άμεσα από τη λειτουργία της villin.



Εικόνα 1.3: Σχηματική αναπαράσταση των βιολογικών διεργασιών στις οποίες εμπλέκεται η πρωτεΐνη villin. (Η εικόνα αναπαράγεται άνευ αδείας [66])

Αναφορικά με τη δομή της, η villin αποτελείται συνολικά από 827 κατάλοιπα, τα οποία σχηματίζουν 6 ομόλογες επικράτειες (S1-S6), που είναι συντηρημένες μεταξύ των πρωτεϊνών της υπεροικογένειας της villin, καθώς επίσης και μια επικράτεια στο καρβοξυτελικό άκρο γνωστή ως "headpiece" [66]. Μέσα στις επικράτειες αυτές, όπως φαίνεται στην **Εικόνα 1.4**, υπάρχουν θέσεις διμερισμού της πρωτεΐνης, πρόσδεσης λιπιδίων, πρόσδεσης ασβεστίου, πρόσδεσης F-ακτίνης και G-ακτίνης, ομαδοποίησης, εμπυρήνωσης, επικάλυψης και αποκοπής ακτίνης.

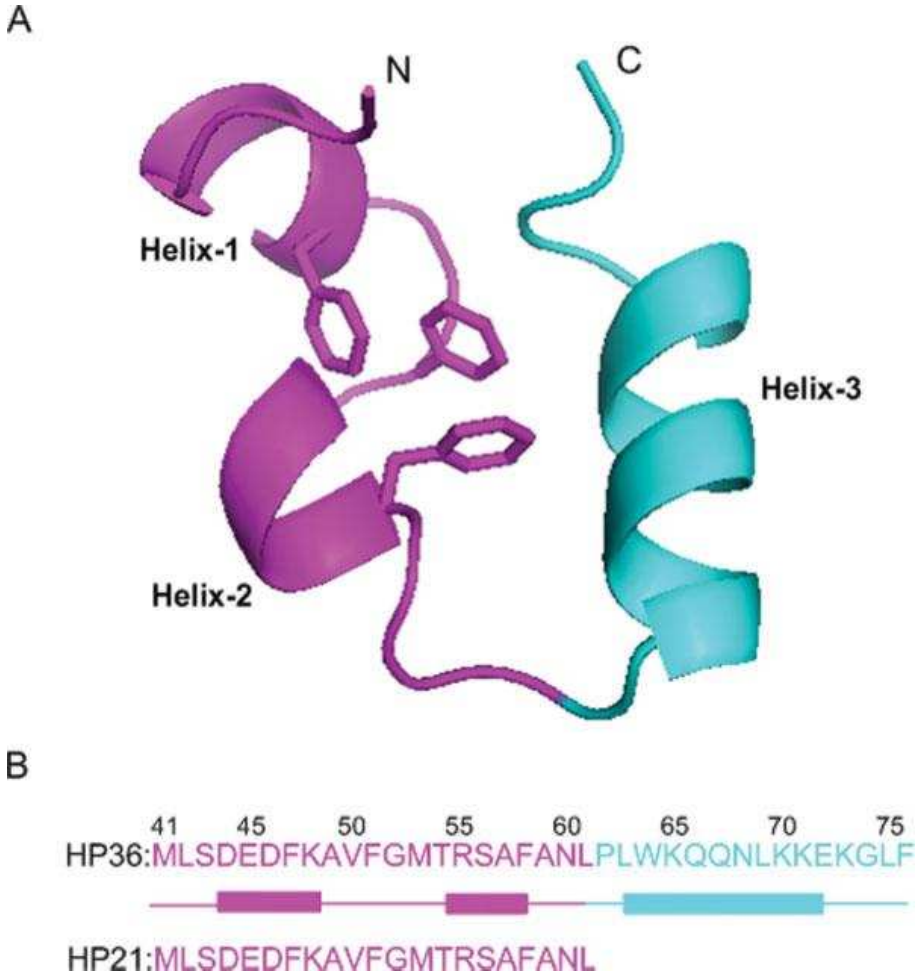


Εικόνα 1.4: Σχηματική αναπαράσταση των επικρατειών και των θέσεων πρόσδεσης της πρωτεΐνης villin. (Η εικόνα αναπαράγεται άνευ αδείας [66])

Η επικράτεια “headpiece” (HP) είναι ένα τροποποιητικό μοτίβο πρόσδεσης F-ακτίνης με συμπαγή δομή αποτελούμενο από 76 αμινοξικά κατάλοιπα. Πειράματα έχουν καταδείξει την πλήρη ικανότητα της επικράτειας αυτής να προσδένει F-ακτίνη ακόμη και όταν εκφράζεται μόνη της in vitro [67]. Όσον αφορά την τρισδιάστατη δομής της, η επικράτεια HP απαρτίζεται από δύο υποεπικράτειες, την αμινοτελική και την καρβοξυτελική, οι οποίες σχηματίζουν έναν στενά πακεταρισμένο υδρόφοβο πυρήνα [68,69]. Η αμινοτελική υποεπικράτεια περιέχει στροφές, μία α-έλικα 4 καταλοίπων και ένα κατάλοιπο ιστιδίνης στο εσωτερικό της δομής από το οποίο καθορίζεται η εξαρτώμενη από το pH αναδίπλωση του πεπτιδίου (πρωτονίωση/απόπρωτονίωση της ιστιδίνης). Αντίθετα, η καρβοξυτελική υποεπικράτεια (HP36) αποτελείται από τρεις α-έλικες και η αναδίπλωσή της δεν εξαρτάται από το pH.

Η υποεπικράτεια HP36 διακρίνεται για την ταχύτητα αναδίπλωσής της (ελάχιστα μs), την απλή τοπολογία της (τρεις α-έλικες) και για το μικρό της μέγεθος (36 κατάλοιπα). Γι’ αυτούς τους λόγους έχει χρησιμοποιηθεί αρκετά συχνά για πειραματικές και υπολογιστικές μελέτες της πρωτεϊνικής αναδίπλωσης, και ιδιαίτερα για τον έλεγχο σχηματισμού τοπικά σταθερών διαμορφώσεων κατά τα αρχικά στάδια της αναδίπλωσης [70].

Για την επίτευξη του παραπάνω στόχου έχει, επίσης, επιχειρηθεί η στρατηγική της “κατάτμησης” του πεπτιδίου HP36 και μελέτης των εκάστοτε τμημάτων γεγονός που έχει οδηγήσει στην αποκάλυψη πλήθους δομικών πληροφοριών σχετικά με την αναδίπλωση του πεπτιδίου αυτού [71,72]. Το πεπτίδιο εκείνο που περιλαμβάνει τα 21 πρώτα κατάλοιπα του HP36, όπου περιέχονται οι δύο πρώτες α-έλικες, είναι γνωστό ως HP21 (**Εικόνα 1.5**). Το HP21 αποτέλεσε το σύστημα μοντέλο της παρούσας εργασίας. Πιο αναλυτικά, πραγματοποιήθηκε μία προσομοίωση μοριακής δυναμικής για το εν λόγω πεπτίδιο και με βάση τα διαθέσιμα πειραματικά δεδομένα [73,74], έγινε έλεγχος της ακρίβειας και αξιοπιστίας των αποτελεσμάτων που συλλέχθηκαν.



Εικόνα 1.5: Φυσική δομή του πεπτιδίου HP36 από τον οργανισμό Gallus gallus σύμφωνα με τον κωδικό καταχώρησης 1VII στην βάση δεδομένων PDB. Α. Τρισδιάστατη δομή των πεπτιδίων HP36 (μωβ και γαλάζιο χρώμα) και HP21 (μωβ χρώμα). Β. Αμινοξική αλληλουχία των πεπτιδίων HP36 (μωβ και γαλάζιο χρώμα) και HP21 (μωβ χρώμα) καθώς και σχηματική αναπαράσταση με ορθογώνια των καταλοίπων που σχηματίζουν τις τρεις α-έλικες. (Η εικόνα αναπαράγεται άνευ άδειας [74])

Κεφάλαιο 2: Προσομοιώσεις Μοριακής Δυναμικής

2.1 Εισαγωγή

Οι υπολογιστικές προσομοιώσεις συνιστούν ένα πολύτιμο εργαλείο για την κατανόηση των ιδιοτήτων διάφορων μοριακών συστημάτων σχετικά με τη δομή και τις μικροσκοπικές αλληλεπιδράσεις που αναπτύσσονται μέσα σ' αυτά. Γι' αυτό το λόγο, γεφυρώνουν το χάσμα τόσο μεταξύ του μικρόκοσμου και του μακρόκοσμου όσο και αυτό μεταξύ της θεωρίας και του πειράματος, καθώς μπορούν να χρησιμοποιηθούν για την επαλήθευση μια θεωρίας είτε μέσα από τη σύγκριση των αποτελεσμάτων που παράγουν με τα πειραματικά δεδομένα είτε αντικαθιστώντας το ίδιο το πείραμα σε περιπτώσεις που παρουσιάζει δυσκολίες η διεξαγωγή του [75].

Οι δύο κυριότερες οικογένειες προσομοιώσεων είναι οι προσομοιώσεις μοριακής δυναμικής (MD simulations) που ασχολούνται με τις δυναμικές ιδιότητες των συστημάτων και οι προσομοιώσεις Monte Carlo (MC simulations) που βασίζονται σε μεθόδους στατιστικής και πιθανοτήτων. Η χρήση τους γίνεται συνήθως ξεχωριστά, αλλά υπάρχουν, ωστόσο, ορισμένες τεχνικές που συνδυάζουν στοιχεία των δύο αυτών κατηγοριών προσομοιώσεων (Langevin Dynamics, Brownian Dynamics) και χρησιμοποιούνται για τη διεξαγωγή περίπλοκων προσομοιώσεων, αφού μειώνουν σημαντικά το υπολογιστικό κόστος [77].

Οι προσομοιώσεις μοριακής δυναμικής υπολογίζουν την συμπεριφορά ενός μοριακού συστήματος σε συνάρτηση με το χρόνο παρέχοντας λεπτομερείς πληροφορίες αναφορικά με τις αλλαγές διαμόρφωσης και τις διακυμάνσεις των βιομορίων. Εν συντομία, η υπολογιστική αυτή μέθοδος βρίσκει εφαρμογή στη μελέτη της πρωτεϊνικής αναδίπλωσης, της πρωτεϊνικής

σταθερότητας, της μοριακής αναγνώρισης καθώς και της μεταφοράς ιόντων σε βιολογικά συστήματα. Επίσης, έχει συμπληρωματικό ρόλο σε τεχνικές σχεδιασμού φαρμάκων και καθορισμού της τρισδιάστατης δομής [76].

Η αφετηρία χρήσης των προσομοιώσεων μοριακής δυναμικής ήταν το 1957 από τους Alder και Wainwright για την υπολογιστική μελέτης της αλληλεπίδρασης μεταξύ σκληρών σφαιρών [78,79]. Περίπου 20 χρόνια αργότερα πραγματοποιήθηκε η πρώτη προσομοίωση μοριακής δυναμικής με ρεαλιστικές συνθήκες με τη χρήση ως μοντέλου του νερού σε υγρή φάση [80]. Τέλος, η πρώτη προσομοίωση μοριακής δυναμικής σε πρωτεΐνη, και συγκεκριμένα στον παγκρεατικό αναστολέα θρυψίνης των βοοειδών (BPTI), συναντάται το 1977 [81]. Στις μέρες μας, η συνεχής βελτίωση των υπολογιστικών συστημάτων έχει επιφέρει ραγδαία αύξηση της χρήσης προσομοιώσεων μοριακής δυναμικής για τη μελέτη μεγάλης ποικιλίας βιολογικών συστημάτων, όπως πρωτεϊνών, λιπιδίων, και νουκλεϊκών οξέων.

2.2 Στατιστική Μηχανική

Τα αποτελέσματα που παράγουν οι προσομοιώσεις μοριακής δυναμικής περιγράφουν την κατάσταση ενός συστήματος σε ατομικό επίπεδο χρησιμοποιώντας μεταβλητές όπως η θέση και η ταχύτητα. Ωστόσο, εφόσον ο στόχος των ερευνητών είναι η μακροσκοπική μελέτη των συστημάτων, κρίνεται αναγκαία η μετατροπή των παρατηρήσιμων δεδομένων σε αντίστοιχα μακροσκοπικά π.χ. πίεση και ενέργεια. Σ' αυτό το σημείο υπεισέρχεται η Στατιστική Μηχανική.

Γενικά, η Στατιστική Μηχανική ανήκει στον κλάδο των φυσικών επιστημών και στοχεύει στην μελέτη μακροσκοπικών συστημάτων με βάση μικροσκοπικά δεδομένα. Με άλλα λόγια, επιχειρεί να προβλέψει την μακροσκοπική συμπεριφορά ενός συστήματος από τις ιδιότητες του κάθε ατόμου που το απαρτίζει ξεχωριστά. Το μέσο με το οποίο επιτυγχάνεται η σύνδεση μικρόκοσμου-μακρόκοσμου είναι ένα πλήθος πολύπλοκων

μαθηματικών εξισώσεων. Επομένως, οι προσομοιώσεις μοριακής δυναμικής παρέχουν τα δεδομένα για την αξιοποίηση των μαθηματικών αυτών συναρτήσεων [76].

2.3 Κλασική Μηχανική και Αλγόριθμοι Ολοκλήρωσης

Ο δεύτερος Νόμος του Νεύτωνα, που περιγράφει τη σχέση της δύναμης με την επιτάχυνση ενός σωματιδίου, καθώς και οι εξισώσεις της κίνησης αποτελούν τη βάση της μεθόδου των προσομοιώσεων μοριακής δυναμικής. Και αυτό διότι γνωρίζοντας τη δύναμη που ασκείται σε κάθε άτομο του συστήματος μπορούμε εύκολα προσδιορίσουμε τις κινητικές παραμέτρους (επιτάχυνση, ταχύτητα) και τη θέση του σε συνάρτηση με το χρόνο παράγοντας ένα τροχιακό του συστήματος αυτού.

Αναλύοντας τα παραπάνω με μαθηματικούς όρους, σύμφωνα με τον δεύτερο Νόμο του Νεύτωνα:

$$F = m\,a \qquad (1.1)$$

όπου F είναι η δύναμη, m η μάζα και a η επιτάχυνση ενός σωματιδίου.

Επίσης, η δύναμη μπορεί να εκφραστεί και ως συνάρτηση της μεταβολής της δυναμικής ενέργειας:

$$F = -\,dV / dr \quad (1.2)$$

όπου V είναι η δυναμική ενέργεια και r η θέση.

Συνδυάζοντας τις σχέσεις (1.1) και (1.2) με τις εξισώσεις της κίνησης προκύπτουν οι εξισώσεις:

$$a = -\,1/m\; dV/dr \qquad \text{και} \qquad -dV/dr = m\; d^2r/dt^2$$

όπου t ο χρόνος.

Η πρώτη συσχετίζει την παράγωγο της δυναμικής ενέργειας με την επιτάχυνση σε συνάρτηση με την θέση, ενώ η δεύτερη την παράγωγο της δυναμικής ενέργειας με τη μεταβολή της θέσης σε συνάρτηση με τον χρόνο.

Οι παραπάνω εξισώσεις αποκαλούνται ντετερμινιστικές ή αιτιοκρατικές, καθώς καθιστούν εφικτή την πρόβλεψη της κατάστασης του συστήματος για οποιαδήποτε χρονική στιγμή, εφόσον είναι γνωστές οι αρχικές θέσεις, η αρχική κατανομή ταχυτήτων και η επιτάχυνση του κάθε ατόμου. Οι αρχικές θέσεις λαμβάνονται από πειραματικές δομές που έχουν προσδιοριστεί με κρυσταλλογραφία ακτινών Χ ή/και φασματοσκοπία NMR. Η κατανομή Maxwell-Boltzmann ή Gaussian χρησιμοποιείται για τον υπολογισμό της αρχικής κατανομής ταχυτήτων:

$$p(v) = (m / 2\pi k_B T)^{1/2} \exp(- mv^2 / 2k_B T)$$

όπου v η ταχύτητα, k_B είναι η σταθερά Boltzmann και T η θερμοκρασία.

Η επιτάχυνση προκύπτει από τον υπολογισμό της δυναμικής ενέργειας με βάση τα δυναμικά πεδία (force fields) που θα αναλυθούν παρακάτω.

Λόγω της πολυπλοκότητας της συνάρτησης που υπολογίζει την δυναμική ενέργεια, η επίλυση των εξισώσεων της κίνησης γίνεται αριθμητικά και όχι αναλυτικά. Με άλλα λόγια, η λύση τέτοιου είδους εξισώσεων δεν μπορεί να βρεθεί επακριβώς (αναλυτικά), επειδή είτε δεν υπάρχει η συγκεκριμένη δυνατότητα (πολύπλοκη εξίσωση) είτε ο χρόνος υπολογισμού που απαιτείται είναι ιδιαίτερα αυξημένος. Επομένως, επιχειρείται μια προσεγγιστική (αριθμητική) μέθοδος επίλυσης, η οποία προβλέπει σε ικανοποιητικό βαθμό τη λύση και μειώνει τον υπολογιστικό χρόνο.

Στην περίπτωσή μας οι εξισώσεις της κίνησης επιλύονται αριθμητικά με τη βοήθεια των αλγορίθμων ολοκλήρωσης. Οι πιο γνωστοί αλγόριθμοι ολοκλήρωσης είναι οι εξής:

- Verlet
- Leap-frog
- Velocity Verlet
- Beeman’s

Γενικά, οι περισσότεροι αλγόριθμοι ολοκλήρωσης βασίζονται σε μια σειρά επεκτάσεων του Taylor για τον προσεγγιστικό υπολογισμό των θέσεων, των ταχυτήτων και των επιταχύνσεων. Η χρησιμότητα των επεκτάσεων του Taylor έγκειται στην μείωση του αριθμού των όρων μιας εξίσωσης καθιστώντας με αυτόν τον τρόπο ευκολότερη την επίλυσής της.

$$r(t+\delta t) = r(t) + v(t)\delta t + a(t)\delta t^2 / 2 + \dots$$

$$v(t+\delta t) = v(t) + a(t)\delta t + b(t)\delta t^2 / 2 + \dots$$

$$a(t+\delta t) = a(t) + b(t)\delta t + \dots$$

Για παράδειγμα ο αλγόριθμος Verlet χρησιμοποιεί τις θέσεις και τις επιταχύνσεις στο χρόνο t και τις θέσεις σε χρόνο t-δt με στόχο τον υπολογισμό των θέσεων για τη χρονική στιγμή t+δt ως εξής:

$$r(t+\delta t) = r(t) + v(t)\delta t + a(t)\delta t^2 / 2$$

$$r(t-\delta t) = r(t) - v(t)\delta t + a(t)\delta t^2 / 2$$

Από την πρόσθεση των δύο παραπάνω εξισώσεων προκύπτει:

$$r(t+\delta t) = 2r(t) - r(t-\delta t) + a(t)\delta t^2$$

Όπως είναι λογικό, οι αλγόριθμοι ολοκλήρωσης παρουσιάζουν ορισμένα σφάλματα όσον αφορά την ακρίβεια των αποτελεσμάτων που παράγουν. Για το λόγο αυτό, πριν την επιλογή ενός αλγορίθμου θα πρέπει να λαμβάνονται υπόψη μερικά κριτήρια, όπως η υπολογιστική αποτελεσματικότητά του, το χρονικό διάστημα που επιτρέπει για ολοκλήρωση καθώς και η δυνατότητα διατήρησης της ενέργειας και της ορμής στο σύστημα, ώστε τα αποτελέσματα να προσεγγίζουν όσο το δυνατόν περισσότερο την φυσική πραγματικότητα [76].

2.4 Δυναμικά Πεδία

Όπως έχει ήδη αναφερθεί, η μελέτη βιολογικών συστημάτων σε ατομικό επίπεδο μέσω των προσομοιώσεων μοριακής δυναμικής απαιτεί γνώση της δυναμική ενέργειας των συστημάτων αυτών. Ωστόσο, εξαιτίας του μεγάλου αριθμού ατόμων και, συνεπώς, του αυξημένου υπολογιστικού κόστους, η χρήση της κβαντομηχανικής θεωρείται απαγορευτική. Γι' αυτόν τον λόγο, η επιστημονική κοινότητα έχει στραφεί στη χρήση των δυναμικών πεδίων (force fields). Τα δυναμικά πεδία αποτελούν εμπειρικές συναρτήσεις υπολογισμού των δυνάμεων και της δυναμικής ενέργειας ενός συστήματος με βάση τις θέσεις των ατόμων και τις αλληλεπιδράσεις μεταξύ τους. Οι αλληλεπιδράσεις αυτές διακρίνονται σε δεσμικές ή εσωτερικές και μη δεσμικές ή εξωτερικές. Οι δεσμικές αλληλεπιδράσεις περιλαμβάνουν το μήκος δεσμού (bond stretch), την γωνία δεσμού (angle bend) και την περιστροφή δίεδρων γωνιών (torsion angle). Από την άλλη πλευρά, στις μη δεσμικές αλληλεπιδράσεις ανήκουν οι αλληλεπιδράσεις van der Waals και οι ηλεκτροστατικές αλληλεπιδράσεις που υπολογίζονται με τις εξισώσεις Lennard-Jones και με τον Νόμο του Coulomb αντίστοιχα. Το άθροισμα της ενέργειας που παράγεται από τις παραπάνω αλληλεπιδράσεις για όλα τα ζεύγη ατόμων του συστήματος δίνει την ολική δυναμική ενέργεια του συστήματος [75,76].

Ανάμεσα στα πιο συχνά χρησιμοποιούμενα δυναμικά πεδία συμπεριλαμβάνονται τα εξής:

- AMBER (Assisted Model Building for Energy Refinement) [82]
- CHARMM (Chemistry at Harvard Macromolecular Mechanics) [83]
- GROMOS (Groningen Molecular Simulation) [84]
- OPLS (Optimized Potentials for Liquid Simulations) [85]

Παρά το γεγονός ότι η κεντρική ιδέα υπολογισμού της δυναμικής ενέργειας είναι σχεδόν η ίδια στα παραπάνω δυναμικά πεδία, υπάρχουν ορισμένες διαφοροποιήσεις στις παραμέτρους και τις εξισώσεις υπολογισμού των

δεσμικών και μη δεσμικών αλληλεπιδράσεων. Για παράδειγμα, η μορφή της συνάρτησης του δυναμικού πεδίου AMBER είναι:

$$V(rN) = \Sigma_{bonds} k_b(l-l_0)^2 + \Sigma_{angles} k_a(\theta-\theta_0)^2 + \Sigma_{dihedrals} V_n[1 + \cos(n\omega - \gamma)] / 2 + \Sigma^{N-1}_{J=1}$$
$$\Sigma^{N}_{i=j+1} [\varepsilon_{i,j} [(r_{0ij} / r_{ij})^{12} - 2(r_{0ij} / r_{ij})^{6}] + q_i q_j / 4\pi \varepsilon_0 r_{ij}]$$

όπου ο πρώτος, ο δεύτερος και ο τρίτος όρος αντιπροσωπεύουν την ενέργεια των τριών δεσμικών αλληλεπιδράσεων αντίστοιχα και ο τέταρτος την ενέργεια των μη δεσμικών αλληλεπιδράσεων

Τα δυναμικά πεδία υφίστανται διαρκώς βελτιστοποιήσεις και διορθώσεις με στόχο να επέλθει συμφωνία σε μεγαλύτερο βαθμό μεταξύ των παραγόμενων αποτελεσμάτων με τα πειραματικά δεδομένα [59-61]. Ειδικότερα, το AMBER από την πρώτη του έκδοση το 1984 έχει βελτιωθεί σημαντικά μέχρι και σήμερα. Η έκδοση ff94 εισήγαγε παραμέτρους κατάλληλες για προσομοιώσεις όλων των ατόμων πρωτεϊνών. Ωστόσο, παρά την ευρεία χρήση του, εμφάνισε ορισμένα μειονεκτήματα, όπως υπερ-σταθεροποίηση των α-ελίκων. Με τις εκδόσεις ff96 και ff99 επιχειρήθηκε τροποποίηση των παραμέτρων υπολογισμού των δίεδρων γωνιών. Η αντικατάσταση των παραμέτρων για τις δίεδρες της κύριας αλυσίδας οδήγησε στη δημιουργία του δυναμικού πεδίου ff99SB με τη χρήση του οποίου επιτεύχθηκε καλύτερη ισορροπία των στοιχείων δευτεροταγούς δομής [86]. Η βελτίωση των παραμέτρων συστροφής των πλευρικών ομάδων καθώς και της ισορροπίας μεταξύ ελικοειδών δομών και σπειραμάτων για ασταθή πεπτίδια είχε ως συνέπεια την παραγωγή του ff99SB-ILDN [87] και του ff99SB-STAR [88] αντίστοιχα. Από τον συνδυασμό των δύο αυτών δυναμικών πεδίων προέκυψε το ff99SB-STAR-ILDN.

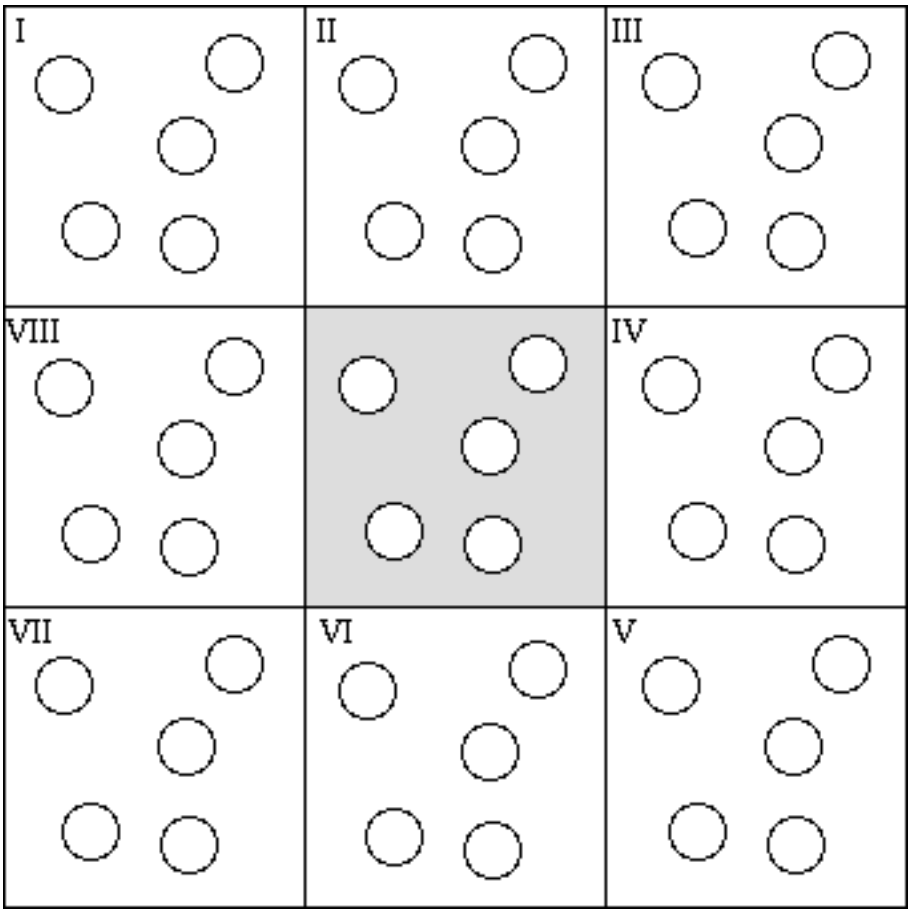
2.5 Διαλύτης και Προσομοιώσεις Μοριακής Δυναμικής

Η χρήση διαλύτη, συνήθως νερού, κατά τη διεξαγωγή προσομοιώσεων μοριακής δυναμικής είναι απαραίτητη, διότι επηρεάζει σε μεγάλο βαθμό τη δομή καθώς και τις δυναμικές και θερμοδυναμικές παραμέτρους ενός βιολογικού συστήματος σε φυσικές συνθήκες. Ο πιο σημαντικός ίσως ρόλος του διαλύτη συνίσταται στον καθορισμό των ηλεκτροστατικών αλληλεπιδράσεων.

Σε γενικές γραμμές, υπάρχουν δύο τρόποι με τους οποίους συμπεριλαμβάνεται σε μια προσομοίωση η επίδραση του διαλύτη. Ο πρώτος αφορά τη χρήση μιας επιπλέον διηλεκτρικής σταθεράς στη συνάρτηση υπολογισμού της δυναμικής ενέργειας, και ειδικότερα στον όρο που αναφέρεται στις ηλεκτροστατικές αλληλεπιδράσεις. Μ' αυτόν τον τρόπο επιτυγχάνεται έμμεσος προσδιορισμός της επίδρασης του διαλύτη στην δυναμική ενέργεια του συστήματος, αφού δεν συμμετέχουν μόρια του στην προσομοίωση. Αντίθετα, με τον δεύτερο τρόπο η προσομοίωση πραγματοποιείται με τη συμμετοχή μορίων διαλύτη. Σ' αυτήν την περίπτωση, ωστόσο, η οριοθέτηση του συστήματος θεωρείται αναγκαία αφενός για την αποφυγή διάχυσης μορίων του διαλύτη και αφετέρου για τη χρήση περιορισμένου αριθμού μορίων διαλύτη, ώστε να είναι εφικτός ο υπολογισμός των μακροσκοπικών ιδιοτήτων του συστήματος [76].

Για την επίτευξη του παραπάνω στόχου, η πιο συχνά χρησιμοποιούμενη μέθοδος είναι ο καθορισμός συνθηκών περιοδικών συνόρων (periodic boundary conditions). Σύμφωνα με τη μέθοδο αυτή, το υπό μελέτη μόριο τοποθετείται μέσα σε ένα κελί στο κέντρο, το οποίο ονομάζεται πρωταρχικό κελί, και γύρω από αυτό υπάρχουν αντίγραφά του προς όλες τις κατευθύνσεις (**Εικόνα 2.1**). Το κάθε άτομο μπορεί να αλληλεπιδράσει με τα γειτονικά άτομα που βρίσκονται είτε στο ίδιο κελί είτε στα περιβάλλοντα κελιά. Επομένως, εάν ένα άτομο εγκαταλείψει το πρωταρχικό κελί, τότε

εισέρχεται το αντίστοιχο άτομο από το αντιπαράλληλο κελί διατηρώντας την περιοδικότητα του συστήματος [75,76].



Εικόνα 2.1: Συνθήκες περιοδικών συνόρων (periodic boundary conditions). Στο κέντρο βρίσκεται το πρωταρχικό κελί και γύρω του αριθμημένα τα περιβάλλοντα κελιά.

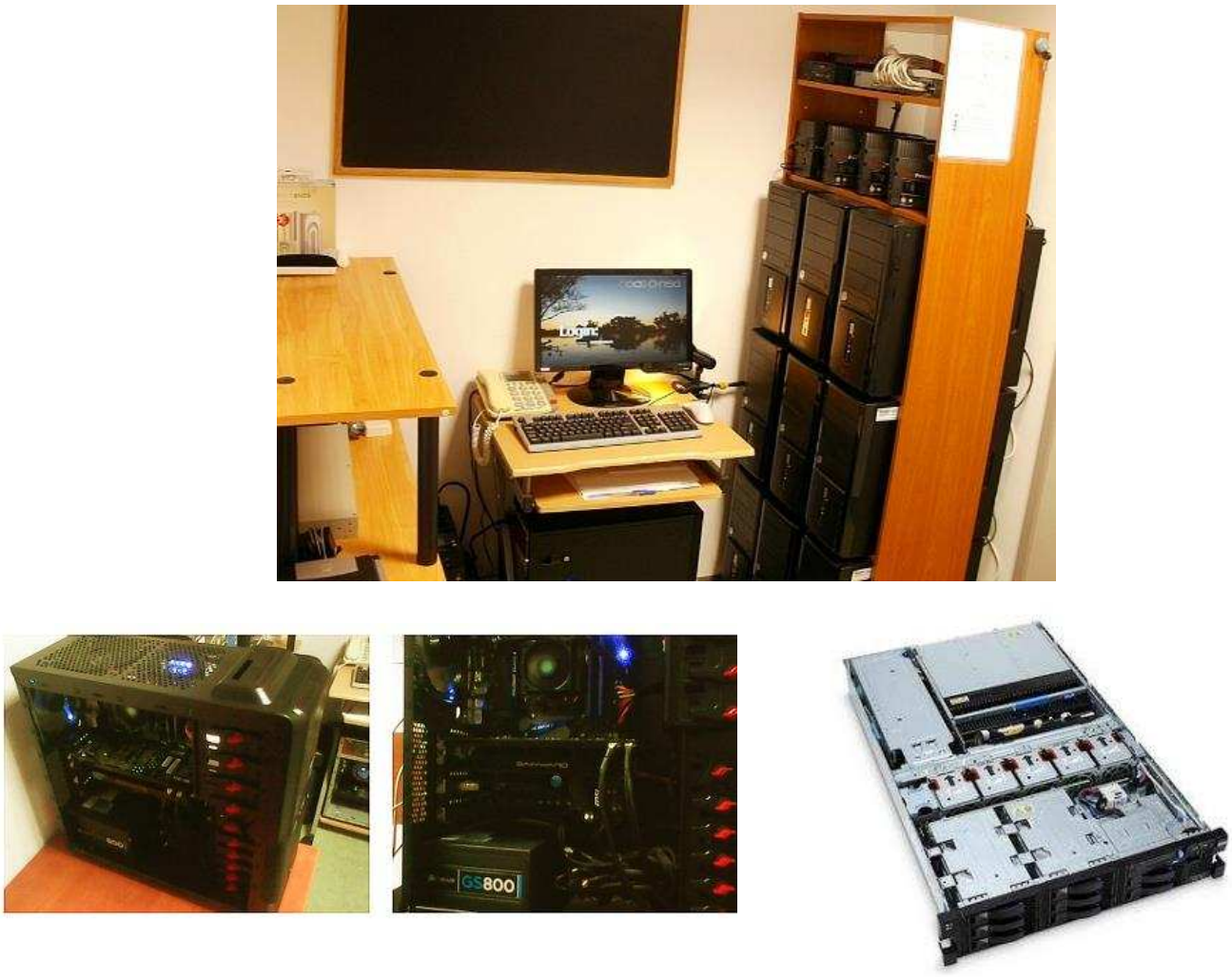
Κεφάλαιο 3: Μέθοδοι

3.1 Εισαγωγή

Η μελέτη αναδίπλωσης του οριακά σταθερού πεπτιδίου HP21 με physics-based μεθόδους πραγματοποιήθηκε με τη διεξαγωγή μιας προσομοίωσης μοριακής δυναμικής με τη βοήθεια του προγράμματος NAMD, το οποίο αποτελεί ένα λογισμικό σχεδιασμένο για την παραγωγή υψηλής απόδοσης προσομοιώσεων μεγάλων βιομοριακών συστημάτων [89]. Το NAMD είναι συμβατό με τα δυναμικά πεδία AMBER και CHARMM. Στην περίπτωσή μας, επιλέχθηκε η χρήση του AMBER ως δυναμικού πεδίου, και ειδικότερα της έκδοσης 99SB-STAR-ILDN.

Κατά γενική ομολογία, η προσομοίωση συστημάτων με σχετικά μεγάλο αριθμό ατόμων είναι μια χρονοβόρα διαδικασία που απαιτεί τη χρήση εξελιγμένων υπολογιστικών συστημάτων. Ένας τρόπος μείωσης του υπολογιστικού κόστους και αύξησης της απόδοσης είναι η παράλληλη σύνδεση υπολογιστών για τη δημιουργία ενός cluster, όπου γίνεται καταμερισμός των εργασιών στους συνδεδεμένους υπολογιστές (κόμβους). Ένα τέτοιο υπολογιστικό συγκρότημα, στο οποίο πραγματοποιήθηκε η προσομοίωση της πτυχιακής αυτής εργασίας, είναι η Norma (**Εικόνα 3.1**) [90]. Η Norma αποτελείται συνολικά από 12 κόμβους συνδεδεμένους σε ένα τοπικό δίκτυο με εγκαταστημένες βιβλιοθήκες και προγράμματα (Beowulf cluster). Συνολικά περιέχει: 96 κεντρικές μονάδες επεξεργασίας (CPU), 114 GB φυσικής μνήμης και 6 GPGPUSs (μονάδες επεξεργασίας γραφικών με δυνατότητα εκτέλεσης εργασιών των CPUs). Οχτώ κόμβοι βασισμένοι σε 4-πύρηνους επεξεργαστές Intel Q6600 Kentsfield 2.4 GHz προσφέρουν 4 GB φυσικής μνήμης ο καθένας καθώς επίσης και από μία μονάδα επεξεργασίας γραφικών (GPU) nvidia GTX-460 οι τέσσερις από αυτούς. Ένας κόμβος

βασισμένος σε επεξεργαστή Intel i7 965 extreme προσφέρει 6 GB φυσικής μνήμης και μία GTX-295. Ένας κόμβος βασισμένος σε 8-πύρηνο επεξεργαστή AMD FX-8150 προσφέρει 4 GB φυσικής μνήμης και μία nvidia GTX-570. Ο κόμβος IBM X3755 server με 48 AMD πυρήνες προσφέρει 64 GB φυσικής μνήμης και 1.8 TB αποθηκευτικό χώρο. Τέλος, ο κεντρικό κόμβος με 4 πυρήνες διαθέτει 8 GB φυσικής μνήμης, 1.5 TB αποθηκευτικό χώρο σε μορφή RAID5 και μια nvidia GTX-260. Όλοι οι κόμβοι συνδέονται μεταξύ τους με μεταγωγέα HP ProCurve 1800-24G Gigabit Ethernet.



Εικόνα 3.1: Το υπολογιστικό συγκρότημα Norma. Επάνω απεικονίζονται ο κεντρικός πυρήνας και οι οχτώ Intel Q6600 Kentsfield κόμβοι. Κάτω αριστερά ο κόμβος με τον 8-πύρηνο επεξεργαστή AMD FX-8150, ενώ κάτω δεξιά ο κόμβος IBM X3755 server.

3.2 Έναρξη προσομοιώσεων με το NAMD

Για την έναρξη μιας προσομοίωσης με δυναμικό πεδίο το AMBER το NAMD χρειάζεται τρία τουλάχιστον αρχεία:

- Ένα αρχείο pdb (Protein Data Bank), το οποίο περιέχει τις συντεταγμένες όλων των ατόμων και των ετερογενών ατόμων του υπό μελέτη συστήματος ή/και τις αντίστοιχες ταχύτητες. Αρχεία pdb είναι διαθέσιμα μέσω της βάσης δεδομένων PDB (http://www.rcsb.org/pdb/home/home.do), αλλά μπορούν επίσης να δημιουργηθούν και από τον ίδιο το χρήστη. Η μορφή ενός τέτοιου αρχείου είναι η εξής:

```
ATOM      1  N   MET A  41       1.177 -10.035  -3.493  1.00  2.04           N
ATOM      2  CA  MET A  41       0.292  -8.839  -3.377  1.00  1.55           C
ATOM      3  C   MET A  41      -0.488  -8.912  -2.063  1.00  1.22           C
ATOM      4  O   MET A  41      -1.039  -9.937  -1.709  1.00  1.32           O
ATOM      5  CB  MET A  41      -0.674  -8.793  -4.565  1.00  1.98           C
ATOM      6  CG  MET A  41      -0.091  -7.889  -5.657  1.00  2.27           C
ATOM      7  SD  MET A  41      -0.153  -8.747  -7.255  1.00  3.04           S
ATOM      8  CE  MET A  41      -0.971  -7.432  -8.193  1.00  3.78           C
ATOM      9  H1  MET A  41       0.835 -10.784  -2.856  1.00  2.30           H
ATOM     10  H2  MET A  41       1.166 -10.381  -4.475  1.00  2.37           H
```

όπου οι στήλες περιέχουν από αριστερά προς δεξιά τον τύπο καταχώρησης, τον αριθμό του ατόμου, το όνομα του ατόμου, το όνομα του καταλοίπου, τον αριθμό του τμήματος, τον αριθμό του καταλοίπου, τις συντεταγμένες x, y και z, την κατοχή, τον παράγοντα θερμοκρασίας και τον τύπο του ατόμου.

- Ένα αρχείο παραμετροποίησης δυναμικού πεδίου AMBER (AMBER format PARM file), το οποίο περιέχει την τοπολογία και τις αναγκαίες παραμέτρους για τον υπολογισμό της δυναμικής ενέργειας του συστήματος. Για την προσομοίωση μας έγινε χρήση του αρχείου παραμετροποίησης του AMBER ff99SB-STAR-ILDN.
- Ένα αρχείο διαμόρφωσης (configuration file), στο οποίο καθορίζονται όλες οι επιλογές του χρήστη σχετικά με τις συνθήκες και τον τρόπο με τον οποίο θα πραγματοποιηθεί η προσομοίωση. Παρακάτω παρατίθεται το αρχείο διαμόρφωσης της προσομοίωσής μας (**Παράρτημα 1**). Οι παράμετροι και οι επιλογές που χρησιμοποιήθηκαν αναλύονται στη συνέχεια.

3.3 Προετοιμασία συστήματος και στάδια προσομοίωσης

Το πρώτο στάδιο για τη διεξαγωγή της προσομοίωσης ήταν η προετοιμασία του συστήματος. Αρχικά, με τη βοήθεια του προγράμματος RIBOSOME [91] δημιουργήθηκε το αρχείο pdb του πεπτιδίου HP21 με αμινοξική αλληλουχία MLSDEDFKAVFGMTRSAFANL σε πλήρως αποδιαταγμένη μορφή (fully extended state). Κατά τη σύνθεση του πεπτιδίου προστέθηκε στο καρβοξυτελικό άκρο ένα κατάλοιπο N-μεθυλαμίνης (NME), η οποία είναι μια μη φορτισμένη οργανική ένωση που χρησιμοποιείται συχνά στα τερματικά άκρα πρωτεϊνών και πεπτιδίων με στόχο την σταθεροποίηση των δομών [92]. Ακολούθησε η προσθήκη των ατόμων υδρογόνου και των ιόντων και η ενυδάτωση του συστήματος με το πρόγραμμα LEAP από τα εργαλεία του AMBER [93]. Η προσομοίωση πραγματοποιήθηκε με τη χρήση συνθηκών οριακών συνόρων (periodic boundary conditions) όπου το μέγεθος της κάθε κυψελίδα ήταν τέτοιο ώστε να εξασφαλίζει έναν ελάχιστο διαχωρισμό μεταξύ των γειτονικών κελιών της τάξεως των 16 Å. Ως δυναμικό πεδίο χρησιμοποιήθηκε το AMBER 99SB-STAR-ILDN και ως μοντέλο νερού το TIP3P.

Επιπρόσθετα, η προσομοίωση επιλέχθηκε να γίνει σε ένα συνεχές εύρος θερμοκρασιών μεταξύ 300 K και 400 K, σύμφωνα με την μέθοδο adaptive tempering του NAMD [94]. Η μέθοδος αυτή επιταχύνει τη διαδικασία δειγματοληψίας διαμορφώσεων για την εύρεση των δομών που αντιστοιχούν στο ολικό ενεργειακό ελάχιστο και λειτουργεί ως εξής: όταν η δυναμική ενέργεια μιας παραγόμενης δομής είναι χαμηλότερη από τη μέση τιμή της δυναμικής ενέργειας έως εκείνο το χρονικό σημείο, τότε μειώνεται η θερμοκρασία. Αντίθετα, όταν η δυναμική ενέργεια μιας παραγόμενης δομής είναι μεγαλύτερη από τη μέση τιμή της δυναμικής ενέργειας, τότε η θερμοκρασία αυξάνεται [89].

Πριν την έναρξη της παραγωγικής φάσης μιας προσομοίωσης απαιτείται να γίνει ελαχιστοποίηση και εξισορρόπηση του συστήματος. Κατά την ελαχιστοποίηση πραγματοποιείται αναζήτηση του ενεργειακού τοπίου του μορίου μέσα από τη συστηματική αλλαγή των θέσεων των ατόμων και τον υπολογισμό της ενέργειας για κάθε θέση με στόχο την εύρεση ενός τοπικού ενεργειακού ελαχίστου. Η εξισορρόπηση αφορά τη μοριακή δυναμική του συστήματος και περιλαμβάνει την επίλυση του Δεύτερου Νόμου του Νεύτωνα για κάθε άτομο του συστήματος υπαγορεύοντας το τροχιακό του.

Το πρωτόκολλο σύμφωνα με το οποίο διεξάχθηκε η προσομοίωση ήταν το εξής: αρχικά έγινε ελαχιστοποίηση της ενέργειας του συστήματος μέσα σε 1000 βήματα και, στη συνέχεια, ακολούθησε μία μικρή φάση θέρμανσης μέχρι τη θερμοκρασία των 300 K με βήμα 20 K για χρονικό διάστημα 32 ps. Έπειτα, το σύστημα εξισορροπήθηκε για 10 ps κάτω από σταθερή πίεση και θερμοκρασία (συνθήκες NpT). Ο έλεγχος της θερμοκρασίας και της πίεσης έγινε με τη χρήση του δυναμικού Nosè-Hoover Langevin και μεθόδους Langevin piston barostat control, ενώ η εφαρμογή του adaptive tempering έγινε με τη χρήση του θερμοστάτη Langevin υπό σταθερή πίεση 1 atm. Για την παραγωγική φάση χρησιμοποιήθηκε ο αλγόριθμος ολοκλήρωσης πολλαπλών χρονικών βημάτων Verlet-I με βήμα 2 fs. Οι μη δεσμικές αλληλεπιδράσεις μικρού εύρους υπολογίζονταν σε κάθε χρονικό βήμα και οι ηλεκτροστατικές αλληλεπιδράσεις μεγάλου εύρους κάθε δύο χρονικά βήματα χρησιμοποιώντας τη μέθοδο PME (Particle Mesh Ewald) [95]. Η μέθοδος αυτή χρησιμοποιείται για τον υπολογισμό των ηλεκτροστατικών αλληλεπιδράσεων

σε ένα σύστημα στο οποίο εφαρμόζονται συνθήκες περιοδικών συνόρων. Το όριο για τον υπολογισμό των αλληλεπιδράσεων van der Waals ήταν στα 9 Å και ο περιορισμός των δεσμών μεταξύ των υδρογόνων και άλλων ατόμων έγινε με τη βοήθεια του προγράμματος SHAKE [96]. Το τροχιακό παράχθηκε με την αποθήκευση των ατομικών συντεταγμένων ολόκληρου του συστήματος κάθε 0.8 ps.

Συνολικά, η προσομοίωση είχε διάρκεια 15 μs, τα οποία αντιστοιχούν περίπου σε 405 ημέρες φυσικού χρόνου, και οδήγησε στην παραγωγή 18835852 διαμορφώσεων (frames).

Κεφάλαιο 4: Αποτελέσματα

4.1 Εισαγωγή

Στο τέταρτο και τελευταίο μέρος της διπλωματικής αυτής εργασίας θα ασχοληθούμε με την ανάλυση του τροχιακού μοριακής δυναμικής του πεπτιδίου HP21 και την παρουσίαση των αποτελεσμάτων καθώς επίσης και με τη σύγκριση των αποτελεσμάτων αυτών με γνωστά πειραματικά δεδομένα.

Η ανάλυση του τροχιακού πραγματοποιήθηκε κατά κύριο λόγο με τη χρήση των προγραμμάτων CARMA [97] και GRCARMA [98] και περιλαμβάνει ένα πλήθος μεθόδων με στόχο την ανάδειξη των δυναμικών και κινητικών ιδιοτήτων των διαφορετικών στερεοδιαμορφώσεων που εντοπίστηκαν. Η παραγωγή των εικόνων, των διαγραμμάτων και των γραφικών παραστάσεων που παρουσιάζονται στη συνέχεια έγινε με τη χρήση των προγραμμάτων VMD [99], RASTER3D [100], CARMA και GRACE [101].

Συνοπτικά, οι μέθοδοι που χρησιμοποιήθηκαν ήταν:

- Ο έλεγχος της σύγκλισης του τροχιακού και της επάρκειας του δείγματος με το πρόγραμμα Good Turing [102]
- Αναλύσεις βασισμένες στον υπολογισμό της ρίζας της μέσης τετραγωνικής απόκλισης (root-mean-square deviation – RMSD)
- Σύγκριση των διαμορφώσεων του τροχιακού με τη φυσική δομή
- Ανάλυση και σύγκριση με βάση την κατανομή της θερμοκρασίας
- Πρόβλεψη της δευτεροταγούς δομής με τη βοήθεια του αλγορίθμου STRIDE [103]
- Ομαδοποίηση (clustering) με βάση δεδομένα που προέρχονται από ανάλυση κύριων συνιστωσών (principal component analysis – PCA)

Το RMSD αποτελεί ένα σημαντικό εργαλείο υπολογισμού της μέσης απόστασης μεταξύ ατόμων διαφορετικών στερεοδιαμορφώσεων και χρησιμοποιείται ευρέως στη Δομική Βιολογία για τη σύγκριση πρωτεϊνικών δομών. Σε γενικές γραμμές, δύο δομές εμφανίζουν ομοιότητα σε σημαντικό βαθμό όταν το RMSD < 2 Å. Η μαθηματική εξίσωση με βάση την οποία υπολογίζεται η τιμή του RMSD είναι η εξής:

$$\mathrm{RMSD}(\mathbf{v},\mathbf{w}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}((v_{ix}-w_{ix})^2+(v_{iy}-w_{iy})^2+(v_{iz}-w_{iz})^2)}$$

όπου v και w είναι δύο διαφορετικά σετ δεδομένων, n το σύνολο των ατόμων και v – w η διαφορά των συντεταγμένων για τα αντίστοιχα άτομα στις τρεις διαστάσεις (x,y,z).

Όσον αφορά τη σύγκριση των αποτελεσμάτων με τα πειραματικά δεδομένα, έγινε χρήση των χημικών μετατοπίσεων από τη μέθοδο του πυρηνικού μαγνητικού συντονισμού (NMR chemical shifts) για το πεπτίδιο HP21 του οργανισμού Gallus gallus, οι οποίες είναι διαθέσιμες από τη δημοσίευση της ομάδας του Raleigh [74]. Πιο αναλυτικά, χρησιμοποιήθηκαν οι τιμές των πειραματικών δευτεροταγών χημικών μετατοπίσεων (secondary chemical shifts), ο υπολογισμός των οποίων έγινε με βάση τις τιμές των πειραματικών χημικών μετατοπίσεων [74] και τις τιμές τυχαίου σπειράματος από την ομάδα του Wishart [44].

Οι τιμές των δευτεροταγών χημικών μετατοπίσεων για τις δομές της προσομοίωσης υπολογίστηκαν με τον συνδυασμό των προγραμμάτων CARMA και SPARTA+ [104] μέσω ενός perl script (**Παράρτημα 2**), το οποίο παράγει αρχεία pdb για τις διαμορφώσεις του τροχιακού, υπολογίζει τις τιμές των δευτεροταγών χημικών μετατοπίσεων για τα αρχεία αυτά και βρίσκει το μέσο όρο και την τυπική απόκλιση των τιμών αυτών για όλα τα άτομα που συμμετέχουν στον υπολογισμό. Ειδικότερα, το SPARTA+ είναι ένα λογισμικό που έχει ως στόχο την πρόβλεψη των χημικών μετατοπίσεων πρωτεϊνικών δομών και η λειτουργία του βασίζεται σε ένα τεχνητό νευρωνικό δίκτυο.

Για τη σύγκριση μεταξύ των πειραματικών δευτεροταγών μετατοπίσεων και των αντίστοιχων από την προσομοίωση έγινε χρήση δύο στατιστικών αναλύσεων: της ανάλυσης reduced x^2 και του γραμμικού συντελεστή συσχέτισης (linear correlation coefficient). Και οι δύο παραπάνω μέθοδοι εξετάζουν τη συμφωνία μεταξύ των τιμών δύο διαφορετικών σετ δεδομένων. Ο στατιστικός δείκτης reduced x^2 υπολογίζεται με βάση τον ακόλουθο τύπο:

$$\chi^2_{\text{red}} = \frac{\chi^2}{\nu} = \frac{1}{\nu}\sum \frac{(O-E)^2}{\sigma^2}$$

όπου Σ είναι το άθροισμα, Ο οι παρατηρούμενες τιμές που αντιστοιχούν στις δευτεροταγείς χημικές μετατοπίσεις των δομών της προσομοίωσης, Ε οι αναμενόμενες τιμές που αντιστοιχούν στις πειραματικές δευτεροταγείς χημικές μετατοπίσεις, σ^2 η διασπορά των παρατηρούμενων τιμών και ν οι βαθμοί ελευθερίας.

- Όταν $x^2_{red} = 1$ τότε τα παρατηρούμενα και τα αναμενόμενα δεδομένα είναι σε συμφωνία με τη διασπορά και υπάρχει υψηλός βαθμός συσχέτισης
- Όταν $x^2_{red} > 1$ τότε είτε δεν υπάρχει πλήρης συσχέτιση μεταξύ των δεδομένων είτε οι τιμές της διασποράς έχουν υποτιμηθεί
- Όταν $x^2_{red} < 1$ τότε έχουμε συσχέτιση σε υπερβολικό βαθμό (over-fitting), γεγονός που οφείλεται είτε σε στατιστικό "θόρυβο" είτε σε υπερτίμηση των τιμών της διασποράς

Ο υπολογισμός του γραμμικού συντελεστή συσχέτισης (Pearson product-moment correlation coefficient – r) γίνεται με τον ακόλουθο τύπο:

$$r = r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

όπου Σ είναι το άθροισμα, x_i η τιμή του πρώτου σετ δεδομένων για τη θέση i, y_i η τιμή του δεύτερου σετ δεδομένων για τη θέση i και x,y ο μέσος όρος όλων των τιμών του πρώτου και δεύτερου σετ αντίστοιχα.

- Όταν r = 1 τότε έχουμε πλήρη θετική συσχέτιση μεταξύ των δύο σετ δεδομένων

- Όταν r = -1 τότε έχουμε πλήρη αρνητική συσχέτιση
- Όταν r = 0 τότε τα δεδομένα δεν συσχετίζονται καθόλου μεταξύ τους

Στο **Παράρτημα 3** βρίσκονται τα δύο perl script μέσω των οποίων πραγματοποιήθηκαν οι υπολογισμοί του reduced x^2 και του γραμμικού συντελεστή συσχέτισης.

Αξίζει να επισημανθεί ότι για τον προσδιορισμό του βαθμού συσχέτισης μεταξύ των πειραματικών δεδομένων και των δεδομένων της προσομοίωσης χρησιμοποιήθηκαν οι τιμές των δευτεροταγών χημικών μετατοπίσεων για τα άτομα $^{13}C^{\alpha}$, $^{13}C^{\beta}$ και ^{13}CO, όπως διατέθηκαν από το πείραμα, εξαιτίας της ευαισθησίας που παρουσιάζουν στις ελικοειδείς διαμορφώσεις [34,74].

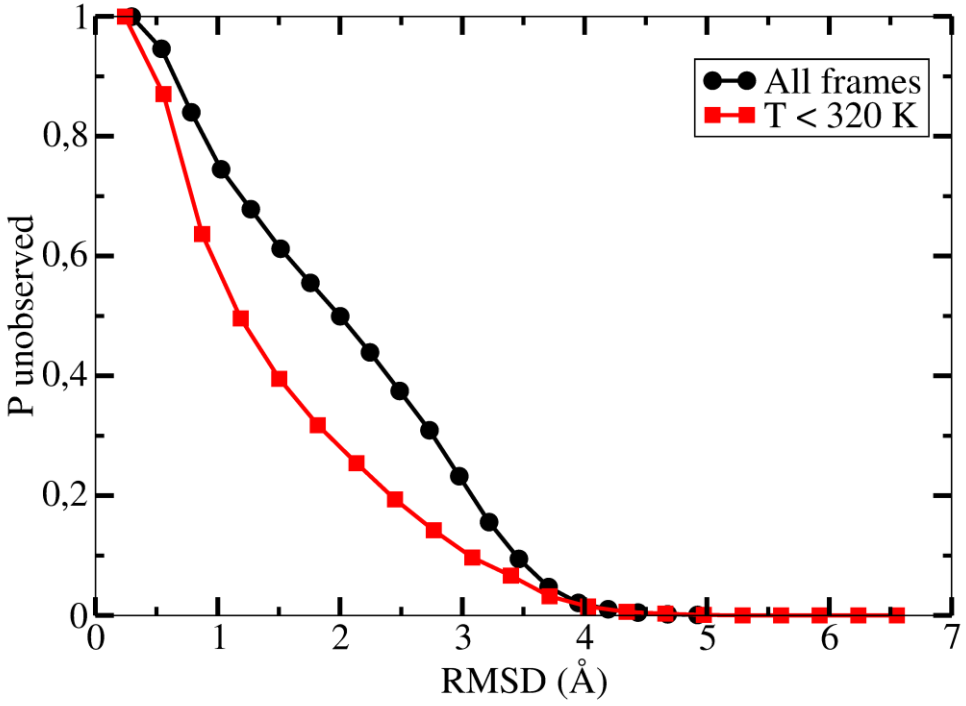
Ωστόσο, εκτός από τις χημικές μετατοπίσεις, επιχειρήσαμε να κάνουμε σύγκριση των δομών της προσομοίωσης με τις πειραματικά προσδιορισμένες δομές του πεπτιδίου HP36 (δομή προερχόμενη από πειράματα NMR με κωδικό καταχώρησης στην PDB 1VII) και της επικράτειας headpiece (HP) (κρυσταλλογραφική δομή με κωδικό καταχώρησης στην PDB 1YU5) του οργανισμού Gallus gallus για τα κατάλοιπα που αντιστοιχούν στο πεπτίδιο HP21.

4.2 Σύγκλιση και έλεγχος επάρκειας δείγματος

Κατά την έναρξη της ανάλυσης του τροχιακού του πεπτιδίου HP21, τέθηκε ως πρώτος στόχος η αξιολόγηση της επάρκειας του δείγματος που παράχθηκε. Γνωρίζοντας ότι το σύστημα μας είναι ιδιαίτερα ευέλικτο λόγω της οριακής σταθερότητας που εμφανίζει το HP21, καταλήξαμε στο συμπέρασμα πως η πλειονότητα των διαμορφώσεων του δείγματος θα αφορούσε την μη αναδιπλωμένη κατάσταση του πεπτιδίου. Συνεπώς, θεωρήθηκε σχεδόν βέβαιο ότι η προσομοίωση, παρά τη μεγάλη διάρκεια της

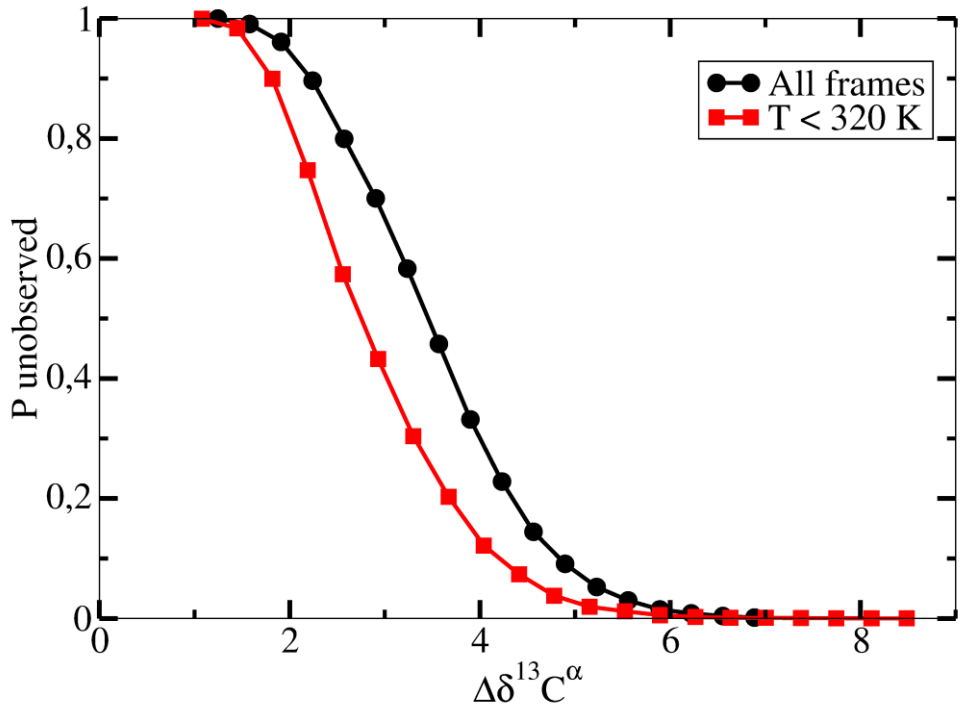
(15 μs) και τη χρήση του adaptive tempering, δεν θα είχε συλλέξει επαρκές δείγμα για τον προσδιορισμό της μη αναδιπλωμένης κατάστασης. Για να επιβεβαιωθεί η παραπάνω θεώρηση επιχειρήθηκε ποσοτικοποίηση της επάρκειας του δείγματος με την πιθανολογική μέθοδο του προγράμματος Good Turing [102]. Πιο συγκεκριμένα, το πρόγραμμα αυτό υπολογίζει την πιθανότητα εμφάνισης διαμορφώσεων διαφορετικών από αυτές που ήδη υπάρχουν στο δείγμα σε περίπτωση που συνεχιζόταν η προσομοίωση. Ο υπολογισμός αυτός πραγματοποιήθηκε με βάση τόσο τη ρίζα της μέσης τετραγωνικής απόκλισης (RMSD) όσο και την απόκλιση μεταξύ των δευτεροταγών χημικών μετατοπίσεων για τα άτομα $^{13}C^{\alpha}$.

Τα αποτελέσματα των αναλύσεων με το Good Turing ήταν τα αναμενόμενα. Όπως φαίνεται στις **Εικόνες 4.1 και 4.2**, η πιθανότητα παρατήρησης σημαντικά διαφορετικών δομών είναι ιδιαίτερα αυξημένη σε περίπτωση συνέχισης της προσομοίωσης, συνεπώς το δείγμα μας δεν μπορεί να περιγράψει με επάρκεια ολόκληρο το ενεργειακό τοπίο αναδίπλωσης του πεπτιδίου HP21. Ειδικότερα, σύμφωνα με το Good Turing, η αναμενόμενη μέγιστη τιμή RMSD στο διπλάσιο χρόνο προσομοίωσης ανέρχεται στα 4.5 Å και η αντίστοιχη τιμή για την απόκλιση των δευτεροταγών χημικών μετατοπίσεων περίπου στα 6 ppm.



Εικόνα 4.1: Πιθανότητα εμφάνισης μη παρατηρούμενων δομών σε συνάρτηση με το RMSD για όλες τις διαμορφώσεις της προσομοίωσης και για τις διαμορφώσεις με θερμοκρασία adaptive tempering μικρότερη από 320 K, με βάση τις προβλέψεις του Good Turing.

Για τις διαμορφώσεις με θερμοκρασία από το adaptive tempering μικρότερη από 320 K που αντιστοιχούν στις σταθερότερες δομές οι γραφικές παραστάσεις (κόκκινες γραμμές) καταδεικνύουν σύγκλιση σε υψηλότερο βαθμό σε σχέση με το δείγμα όλων των διαμορφώσεων της προσομοίωσης, δίχως, όμως, το συγκεκριμένο δείγμα να μπορεί να χαρακτηριστεί επαρκές, αφού οι μέγιστες τιμές RMSD και απόκλισης των δευτεροταγών χημικών μετατοπίσεων εκτιμώνται περίπου στα 5 Å και 5.7 ppm εάν διπλασιαζόταν ο χρόνος της προσομοίωσης.

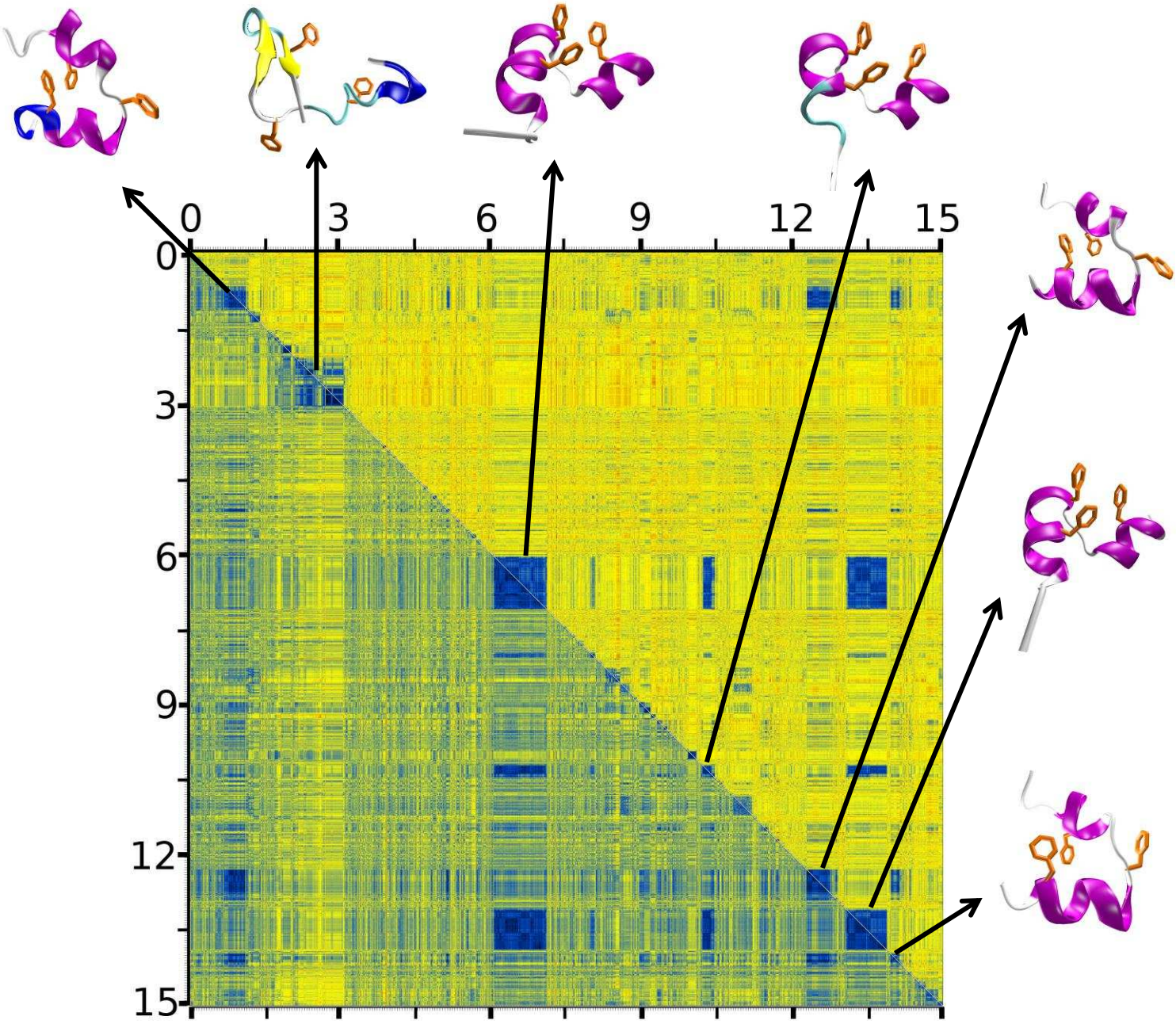


Εικόνα 4.2: Πιθανότητα εμφάνισης μη παρατηρούμενων δομών σε συνάρτηση με την απόκλιση των τιμών των δευτεροταγών χημικών μετατοπίσεων για τα άτομα C^{α} για όλες τις διαμορφώσεις της προσομοίωσης και για τις διαμορφώσεις με θερμοκρασία adaptive tempering μικρότερη από 320 K, με βάση τις προβλέψεις του Good Turing.

4.3 Αναλύσεις RMSD

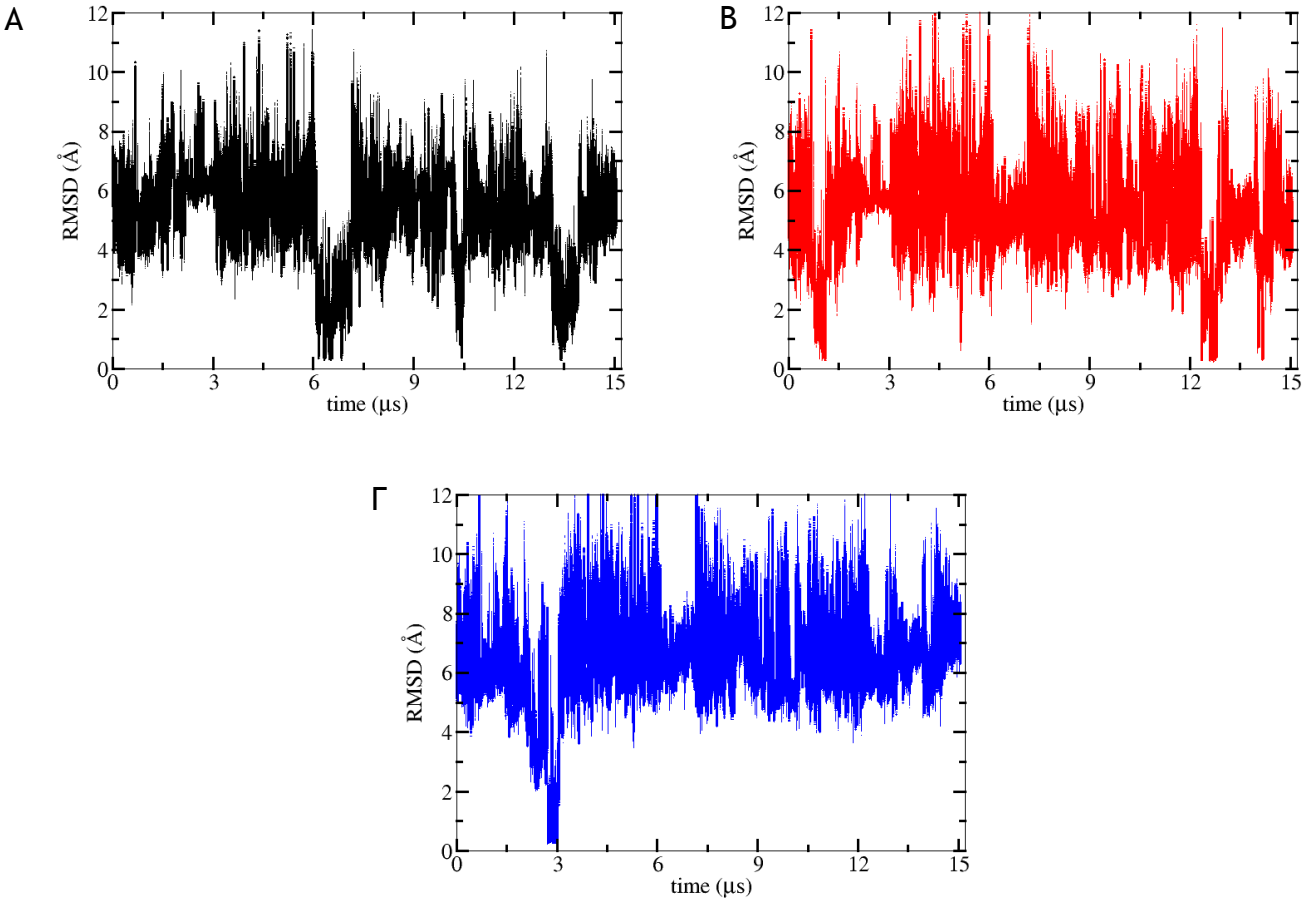
Εφόσον, όπως αποδείχθηκε στην προηγούμενη ενότητα, το δείγμα μας δεν είναι επαρκές ώστε να μελετήσουμε διεξοδικά τα ενδιάμεσα στάδια της πορείας αναδίπλωσης (μη αναδιπλωμένη κατάσταση), στρέψαμε το ενδιαφέρον μας στην εύρεση και απομόνωση γεγονότων αναδίπλωσης που εμφανίζονται κατά τη διάρκεια των 15 μs της προσομοίωσης. Για το λόγο αυτό, δημιουργήθηκε, αρχικά, ένας πίνακας RMSD, στον οποίο γίνεται γραφική αναπαράσταση της σύγκρισης όλων των διαμορφώσεων της προσομοίωσης ανά ζεύγη με βάση τις τιμές RMSD. Οι μπλε αποχρώσεις αντιστοιχούν σε χαμηλές τιμές RMSD (παρόμοιες δομές), οι κίτρινες σε μεσαίες τιμές και οι κόκκινες σε μεγάλες τιμές. Οι μπλε περιοχές που βρίσκονται στο κέντρο της διαγωνίου αντιπροσωπεύουν σταθεροποίηση μιας δομής για χρονικό διάστημα ανάλογο με το μήκος της περιοχής, ενώ οι μπλε περιοχές που βρίσκονται εκτός της διαγωνίου αντιπροσωπεύουν όμοιες δομές, οι οποίες εμφανίστηκαν σε διαφορετικά χρονικά διαστήματα κατά τη διάρκεια της προσομοίωσης.

Ο πίνακας RMSD για το τροχιακό του HP21 παρατίθεται στην **Εικόνα 4.3**. Οι τιμές του RMSD για το κάτω-αριστερά τρίγωνο του πίνακα υπολογιστήκαν με βάση μόνο τα άτομα C^{α}, σε αντίθεση με το πάνω-δεξιά τρίγωνο του πίνακα, όπου ο υπολογισμός περιλάμβανε όλα τα άτομα εκτός από αυτά του υδρογόνου. Το πεπτίδιο παρουσιάζει μία διαρκή αστάθεια, καθώς δεν υιοθετεί μία συνεχή σταθερή διαμόρφωση για μεγάλο χρονικό διάστημα. Παρολ' αυτά, είναι εμφανές πως υπάρχουν τρεις διακριτές αναδιπλωμένες καταστάσεις: η πρώτη συναντάται στα χρονικά διαστήματα 6-7 μs, 10-10.5 μs και 13-14 μs, η δεύτερη μεταξύ 0-1 μs, 12-13 μs και 14-14.5 μs, και, τέλος, η τρίτη μόνο στο διάστημα 2-3 μs. Ενδεικτικά, παραθέτονται οι αντιπροσωπευτικές δομές για τα παραπάνω χρονικά διαστήματα, τα δομικά χαρακτηριστικά των οποίων θα αναλυθούν στις επόμενες ενότητες.



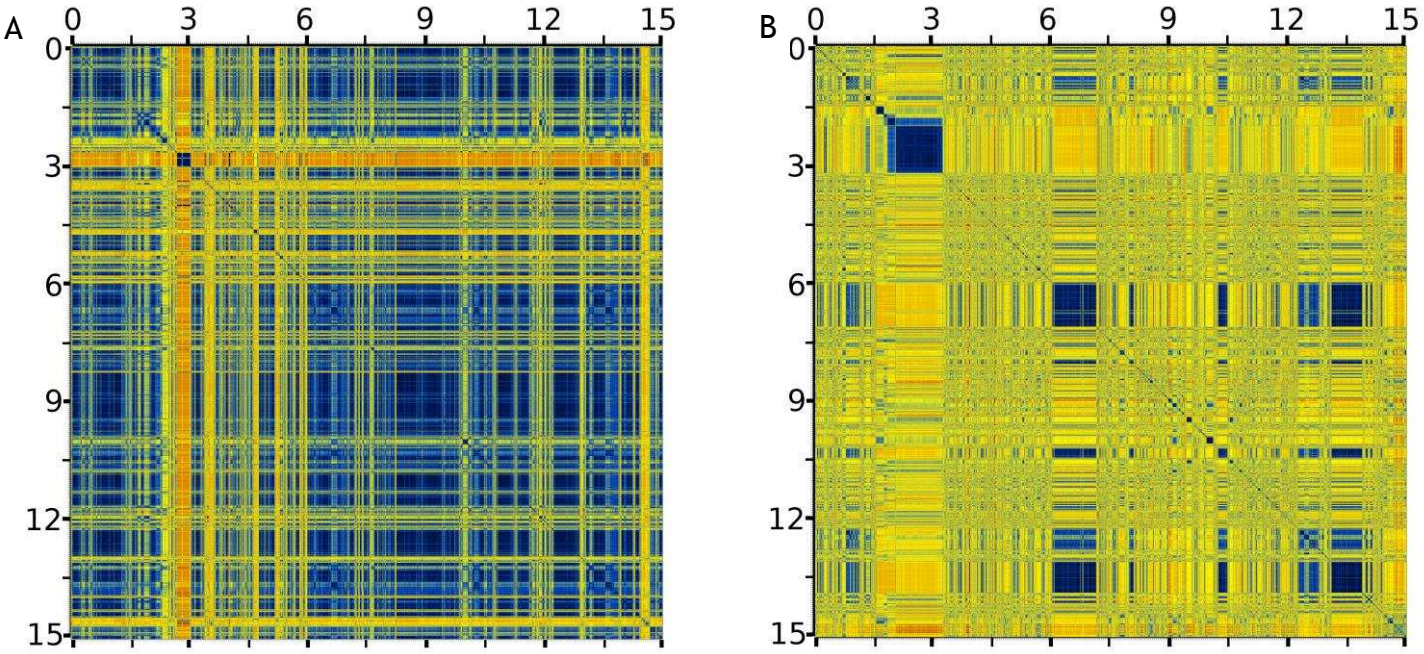
Εικόνα 4.3: Πίνακας RMSD για το τροχιακού του πεπτιδίου HP21. Οι τιμές RMSD για το κάτω τρίγωνο υπολογίστηκαν με βάση μόνο τα άτομα C^{α}, ενώ για το πάνω τρίγωνο με βάση όλα τα άτομα εκτός από αυτά του υδρογόνου. Οι μπλε περιοχές που βρίσκονται στο κέντρο της διαγωνίου αντιπροσωπεύουν σταθεροποίηση μιας δομής για χρονικό διάστημα ανάλογο με το μήκος της περιοχής, ενώ οι μπλε περιοχές που βρίσκονται εκτός της διαγωνίου αντιπροσωπεύουν όμοιες δομές, οι οποίες εμφανίστηκαν σε διαφορετικά χρονικά διαστήματα κατά τη διάρκεια της προσομοίωσης. Γύρω από τον πίνακα υπάρχουν οι αντιπροσωπευτικές δομές για κάθε μεμονωμένο γεγονός αναδίπλωσης.

Τα παραπάνω συμπεράσματα αποτυπώνονται επίσης και στη γραφική παράσταση των τιμών RMSD μεταξύ των αντιπροσωπευτικών διαμορφώσεων των τριών αναδιπλωμένων καταστάσεων και όλων των υπολοίπων διαμορφώσεων του τροχιακού λαμβάνοντας υπόψη στον υπολογισμό μόνο τα άτομα C^{α} (**Εικόνα 4.4**).



Εικόνα 4.4: Γραφική παράσταση των τιμών RMSD μεταξύ των αντιπροσωπευτικών διαμορφώσεων των τριών αναδιπλωμένων καταστάσεων (Α, Β και Γ) και όλων των υπολοίπων διαμορφώσεων του τροχιακού χρησιμοποιώντας μόνο τα άτομα C^{α}.

Τέλος, για να προσδιορίσουμε τη συμπεριφορά αναδίπλωσης των καταλοίπων του HP21 που σχηματίζουν τις δύο α-έλικες στις δομές των HP36 και headpiece (HP), δημιουργήσαμε δύο επιπλέον πίνακες RMSD (**Εικόνα 4.5**). Στους υπολογισμούς για τον πρώτο πίνακα συμπεριλήφθηκαν τα άτομα C^{α} για τα κατάλοιπα 43-49, ενώ στον δεύτερο πίνακα τα άτομα C^{α} για τα κατάλοιπα 54-60. Συμπερασματικά, τα κατάλοιπα της πρώτης έλικας, με εξαίρεση το χρονικά διάστημα 2-3 μs, εμφανίζουν μια σχετική σταθερότητα σε σχέση με τα κατάλοιπα της δεύτερης έλικας, ο πίνακας RMSD για τα οποία μοιάζει με τον αντίστοιχο για όλα τα κατάλοιπα του πεπτιδίου.

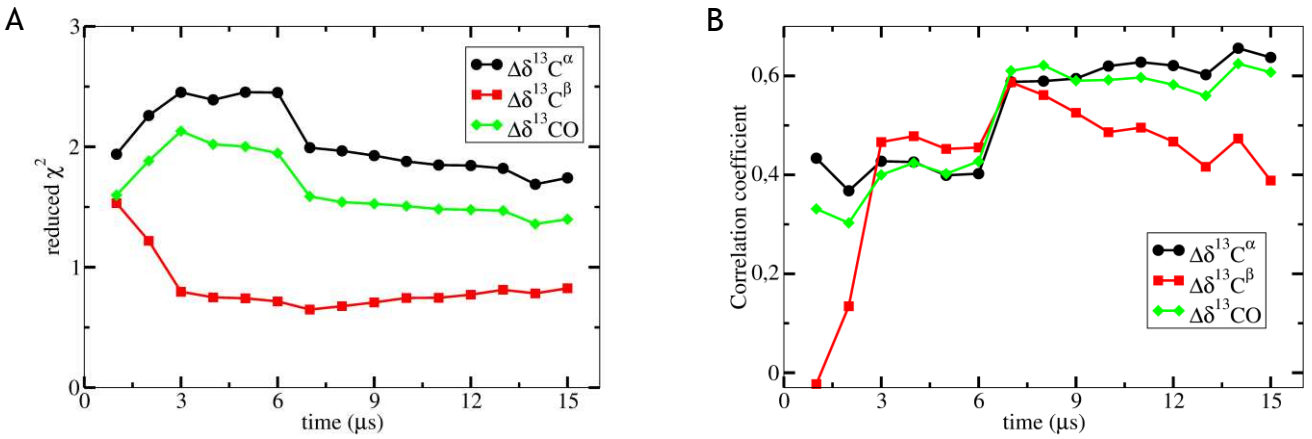


Εικόνα 4.5: Πίνακες RMSD για τα κατάλοιπα του HP21 που έχουν ελικοειδή διαμόρφωση στις δομές των HP36 και HP. Α. Πίνακας RMSD για τα κατάλοιπα 43-49 χρησιμοποιώντας τα άτομα C^{α}. Β. Πίνακας RMSD για τα κατάλοιπα 54-60 χρησιμοποιώντας τα άτομα C^{α}.

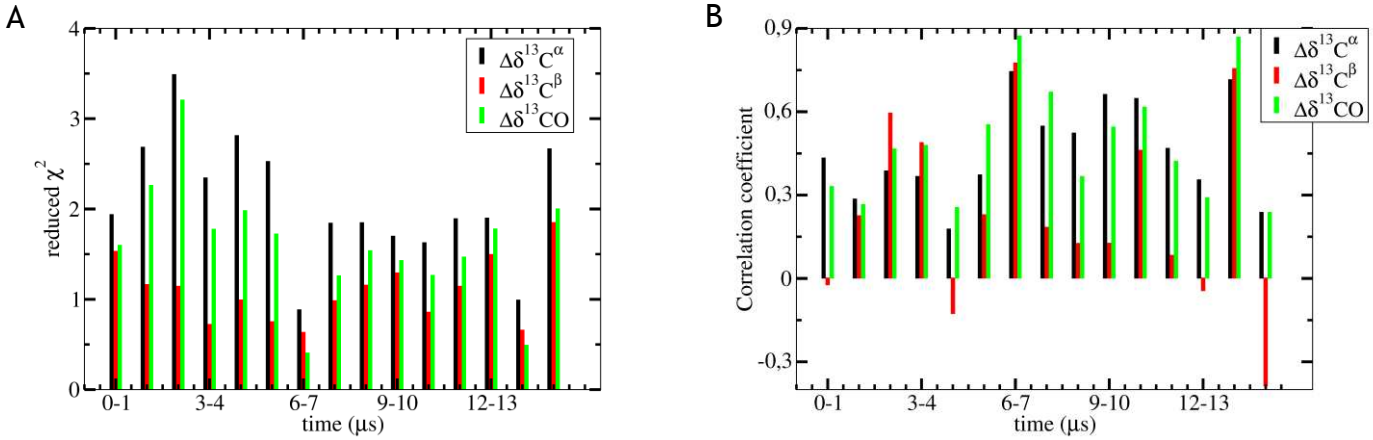
4.4 Σύγκριση με τα πειραματικά δεδομένα

Το επόμενο βήμα ήταν η σύγκριση των δομών του τροχιακού με τα πειραματικά δεδομένα, ώστε να διακρίνουμε πιθανά γεγονότα αναδίπλωσης, οι δομές των οποίων να εμφανίζουν υψηλού βαθμού συσχέτιση με τη φυσική δομή. Η επίτευξη του παραπάνω στόχου βασίστηκε, ως επί το πλείστον, στη χρήση των δευτεροταγών χημικών μετατοπίσεων. Πιο συγκεκριμένα, υπολογίστηκαν οι δευτεροταγείς χημικές μετατοπίσεις για τις διαμορφώσεις όλης της προσομοίωσης και, στη συνέχεια, έγινε σύγκρισή τους με τις αντίστοιχες πειραματικές με τη βοήθεια των στατιστικών αναλύσεων του reduced x^2 και του γραμμικού συντελεστή συσχέτισης. Τα αποτελέσματα της διαδικασίας αυτής παρουσιάζονται στις **Εικόνες 4.6 και 4.7**. Οι γραφικές παραστάσεις της **Εικόνας 4.6** δείχνουν την μεταβολή των τιμών του reduced

x^2 και του συντελεστή συσχέτισης σε συνάρτηση με το χρόνο της προσομοίωσης, ενώ τα ραβδογράμματα της **Εικόνας 4.7** τις τιμές των δύο αυτών στατιστικών δεικτών για κάθε μs της προσομοίωσης ξεχωριστά.



Εικόνα 4.6: Α. Μεταβολή της τιμής του δείκτη συσχέτισης reduced x^2 μεταξύ των πειραματικών δευτεροταγών χημικών μετατοπίσεων και των αντίστοιχων από την προσομοίωση για τα άτομα C^{α}, C^{β} και CO των καταλοίπων 42-60 του πεπτιδίου σε συνάρτηση με το χρόνο. Β. Μεταβολή της τιμής του γραμμικού συντελεστή συσχέτισης μεταξύ των πειραματικών δευτεροταγών χημικών μετατοπίσεων και των αντίστοιχων από την προσομοίωση για τα άτομα C^{α}, C^{β} και CO των καταλοίπων 42-60 του πεπτιδίου σε συνάρτηση με το χρόνο.

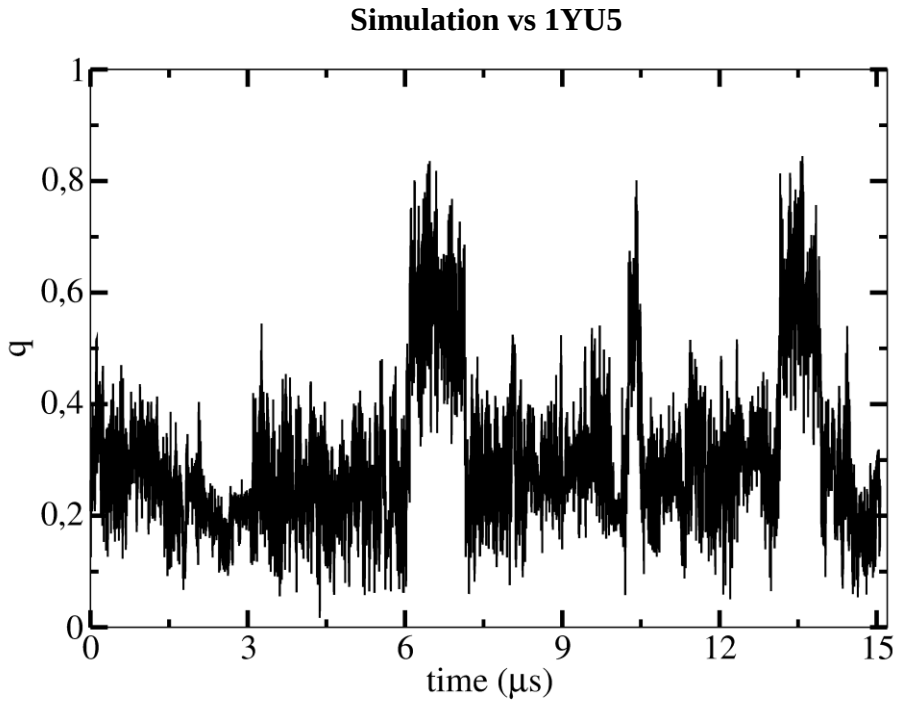


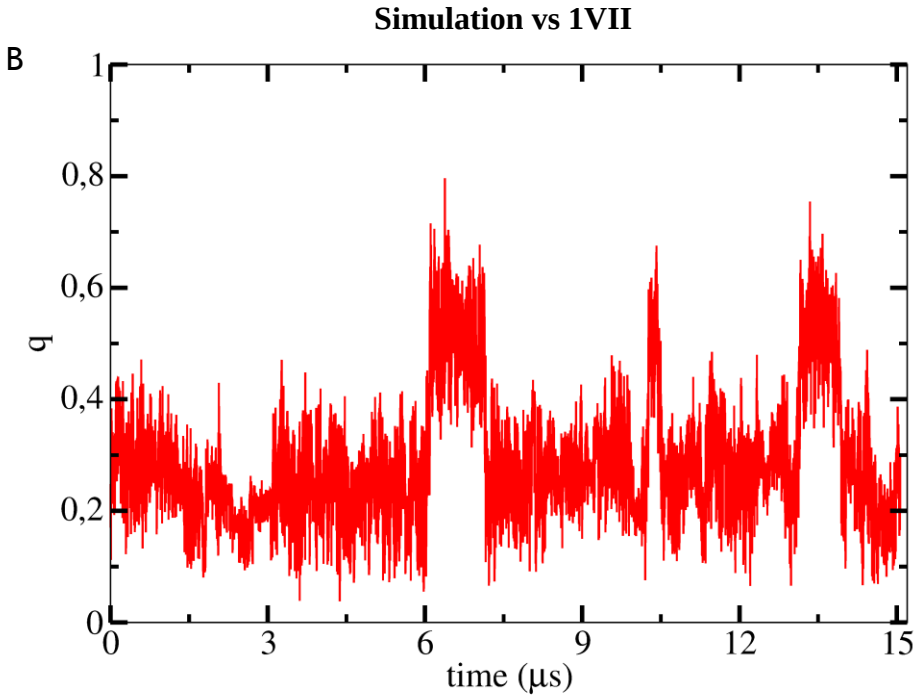
Εικόνα 4.7: Α. Μέση τιμή του δείκτη συσχέτισης reduced x^2 μεταξύ των πειραματικών δευτεροταγών χημικών μετατοπίσεων και των αντίστοιχων από την προσομοίωση για τα άτομα C^{α}, C^{β} και CO των καταλοίπων 42-60 του πεπτιδίου για κάθε μs της προσομοίωσης. Β. Μέση τιμή του γραμμικού συντελεστή συσχέτισης μεταξύ των πειραματικών δευτεροταγών χημικών μετατοπίσεων και των αντίστοιχων από την προσομοίωση για τα άτομα C^{α}, C^{β} και CO των καταλοίπων 42-60 του πεπτιδίου για κάθε μs της προσομοίωσης.

Το συμπέρασμα που μπορεί κανείς να εξαγάγει από τα παραπάνω διαγράμματα είναι το εξής: εφόσον οι πειραματικές τιμές των δευτεροταγών χημικών μετατοπίσεων και αυτές της προσομοίωσης συσχετίζονται σημαντικά και στις δύο στατιστικές αναλύσεις για τα χρονικά διαστήματα 6-7 και 13-14 μs, άρα στα συγκεκριμένα διαστήματα περιέχονται στο τροχιακό διαμορφώσεις που μοιάζουν με την φυσική δομή. Ωστόσο, σύμφωνα με τον πίνακα RMSD όλης της προσομοίωσης (**Εικόνα 4.3**), οι διαμορφώσεις για τα δύο αυτά χρονικά διαστήματα μαζί με εκείνες για το διάστημα 10-10.5 μs, έχουν ταξινομηθεί λόγω της ομοιότητάς μεταξύ τους στην πρώτη ομάδα αναδιπλωμένων διαμορφώσεων. Συνεπώς, φαίνεται ότι οι διαμορφώσεις της ομάδας αυτής υιοθετούν παρόμοια διευθέτηση στο χώρο με τη φυσική δομή.

Θέλοντας να επιβεβαιώσουμε αυτή τη διαπίστωση, προσδιορίσαμε ποσοτικά την ομοιότητα όλων των διαμορφώσεων του τροχιακού σε σχέση με τις πειραματικά προσδιορισμένες δομές της επικράτειας headpiece και του πεπτιδίου HP36 για τα κατάλοιπα που αντιστοιχούν στο HP21 μέσα από τον υπολογισμό των φυσικών επαφών (native contacts) [105]. Τα αποτελέσματα παρουσιάζονται στην **Εικόνα 4.8**.

A



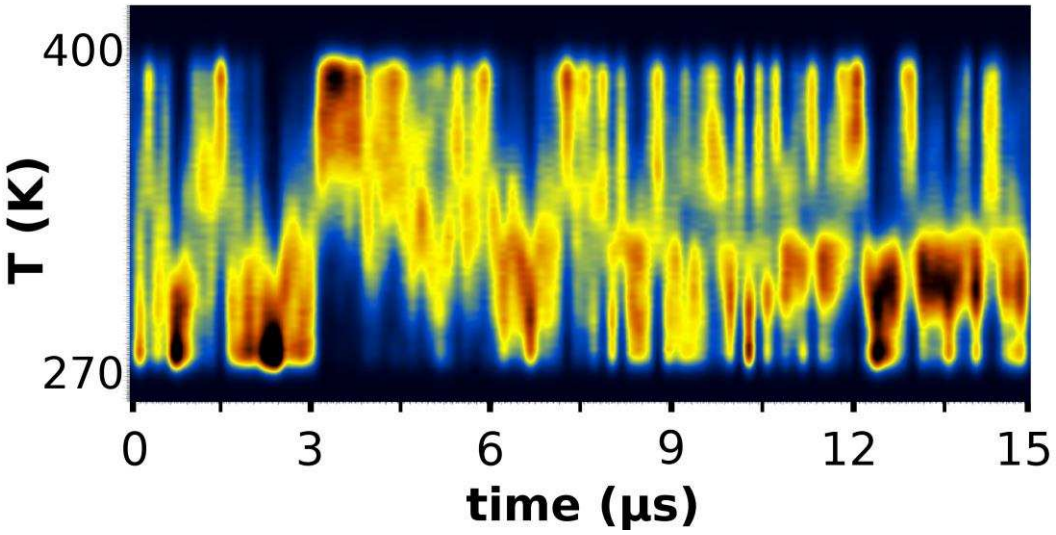


Εικόνα 4.8: Ομοιότητα πειραματικών δομών και δομών από την προσομοίωση σε συνάρτηση με το χρόνο. Ο δείκτης ομοιότητας q υπολογίστηκε με βάση τις φυσικές επαφές και παίρνει τιμές από 0 (παντελώς ανόμοιες δομές) έως 1 (πανομοιότυπες δομές). Α. Γραφική παράσταση της ομοιότητας μεταξύ των διαμορφώσεων της προσομοίωσης και της πειραματικά προσδιορισμένης δομής της επικράτειας headpiece (κωδικός καταχώρησης στην PDB: 1YU5) για τα κατάλοιπα 42-61 σε συνάρτηση με το χρόνο. Β. Γραφική παράσταση της ομοιότητας μεταξύ των διαμορφώσεων της προσομοίωσης και της πειραματικά προσδιορισμένης δομής του πεπτιδίου HP36 (κωδικός καταχώρησης στην PDB: 1VII) για τα κατάλοιπα 41-61 σε συνάρτηση με το χρόνο.

4.5 Αναλύσεις με βάση την θερμοκρασία

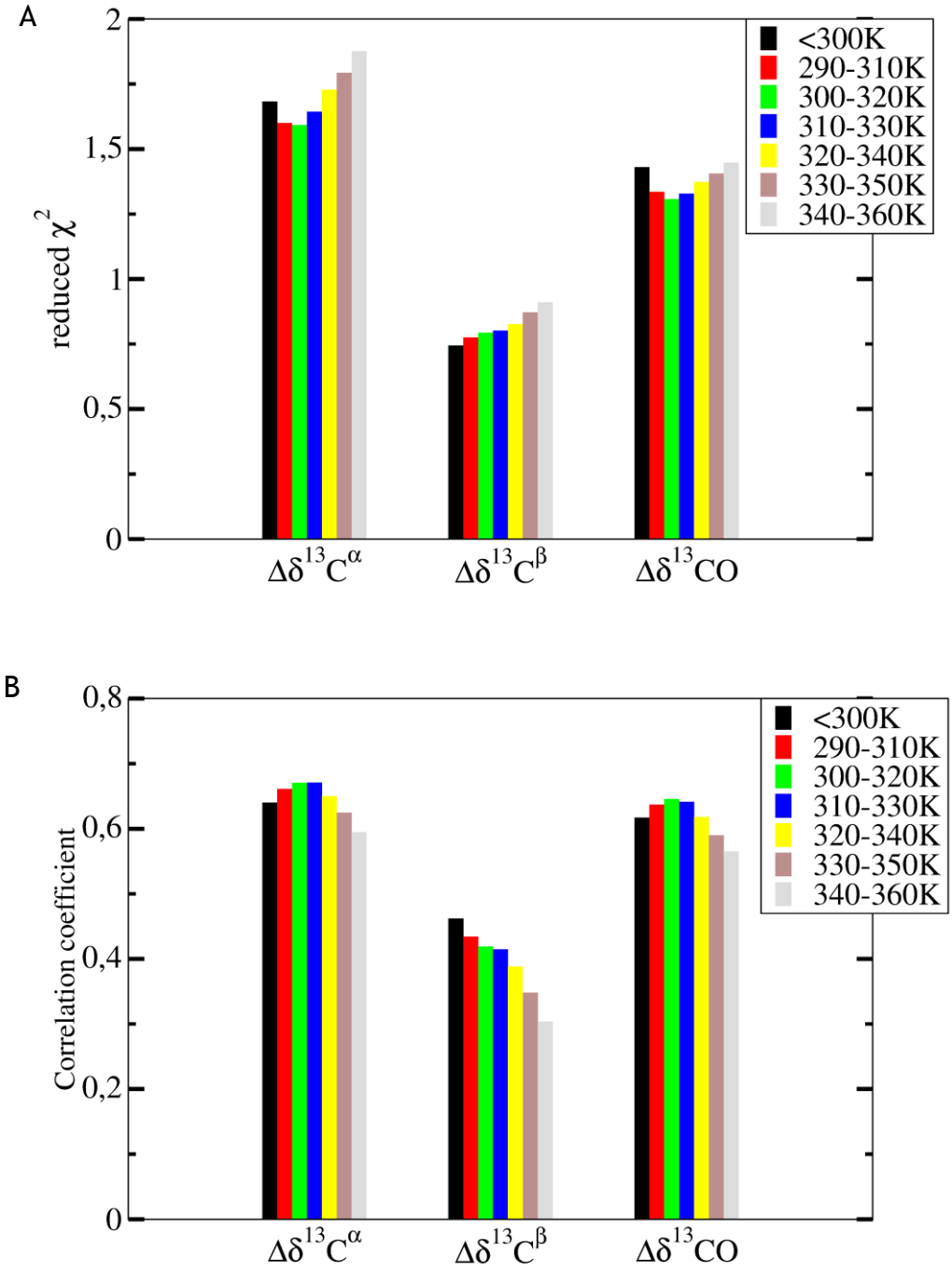
Η χρήση του adaptive tempering κατά τη διεξαγωγή της προσομοίωσης με το NAMD μας έδωσε τη δυνατότητα να αναλύσουμε επιπλέον το τροχιακό που παράχθηκε με βάση την θερμοκρασία δημιουργώντας ένα διάγραμμα κατανομής της θερμοκρασίας κάθε στερεοδιαμόρφωσης σε συνάρτηση με το χρόνο της προσομοίωσης, το οποίο απεικονίζεται στην **Εικόνα 4.9**. Οι μπλε και κίτρινες αποχρώσεις αντιστοιχούν σε μικρό και μεσαίο αριθμό διαμορφώσεων, ενώ οι κόκκινες και μαύρες αποχρώσεις σε μεγάλο πλήθος διαμορφώσεων (κορυφές της κατανομής). Όπως είναι λογικό, οι σταθερότερες δομές και των τριών ομάδων διαμορφώσεων, που διακρίναμε στις προηγούμενες αναλύσεις, συναντώνται σε χαμηλές θερμοκρασίες, μικρότερες από 320 K, (σκούρες περιοχές του διαγράμματος) λόγω της

μειωμένης κινητικότητας του συστήματος. Εξαίρεση αποτελεί μία κορυφή με ασυνήθιστα υψηλές θερμοκρασίες, η οποία εντοπίζεται μετά τα 3 μs, χωρίς, όμως να υπάρχει κάποια άλλη ένδειξη ότι υπάρχει γεγονός αναδίπλωσης για το συγκεκριμένο χρονικό διάστημα.



Εικόνα 4.9: Διάγραμμα κατανομής της θερμοκρασίας σε συνάρτηση με το χρόνο της προσομοίωσης. Οι μπλε και κίτρινες αποχρώσεις αντιστοιχούν σε μικρό και μεσαίο αριθμό διαμορφώσεων, ενώ οι κόκκινες και μαύρες αποχρώσεις σε μεγάλο πλήθος διαμορφώσεων (κορυφές της κατανομής).

Επιπρόσθετα, υπολογίσαμε τις δευτεροταγείς χημικές μετατοπίσεις για τις διαμορφώσεις με θερμοκρασία adaptive tempering που ανήκει στα διαστήματα θερμοκρασιών < 300 K, 290-310 K, 300-320 K, 310-330 K, 320-340 K, 330-350 K, 340-360 K και τις συγκρίναμε με τις πειραματικές δευτεροταγείς χημικές μετατοπίσεις του HP21 χρησιμοποιώντας τόσο το δείκτη reduced x^2 όσο και το γραμμικό συντελεστή συσχέτισης (**Εικόνα 4.10**). Τα αποτελέσματα καταδεικνύουν ότι η μείωση της θερμοκρασίας οδηγεί σε αύξηση της σύγκλισης μεταξύ των πειραματικών δεδομένων και αυτών που προέρχονται από την προσομοίωση. Επομένως, οι διαμορφώσεις που μοιάζουν με τη φυσική δομή έχουν θερμοκρασίες μικρότερες από 320 K. Βέβαια είναι δύσκολο να εντοπίσουμε το διάστημα θερμοκρασιών στο οποίο εμφανίζεται η μέγιστη συσχέτιση μεταξύ πειράματος και προσομοίωσης, καθώς δεν υπάρχει καθολική συμφωνία στις τιμές των στατιστικών δεικτών για τα 3 είδη των δευτεροταγών χημικών μετατοπίσεων.



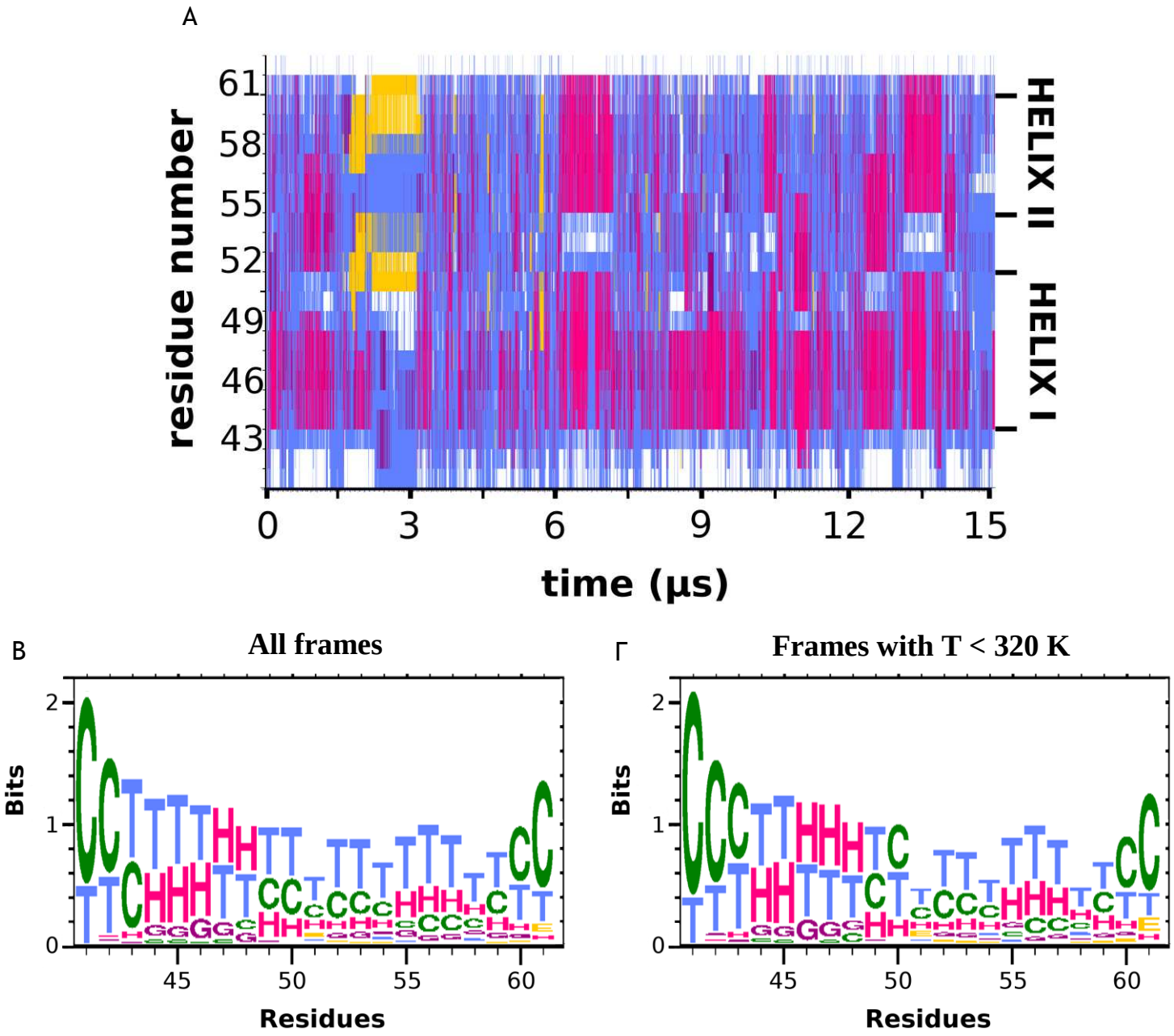
Εικόνα 4.10: Σύγκριση των τιμών δευτεροταγών χημικών μετατοπίσεων για τις διαμορφώσεις με θερμοκρασία adaptive tempering που να ανήκει στα διαστήματα θερμοκρασιών < 300 K, 290-310 K, 300-320 K, 310-330 K, 320-340 K, 330-350 K, 340-360 K με τις τιμές των πειραματικών δευτεροταγών χημικών μετατοπίσεων του πεπτιδίου HP21 για τα άτομα C^{α}, C^{β}, CO των καταλοίπων 42-60. Α. Σύγκριση με βάση τον δείκτη συσχέτισης reduced x^2. Β. Σύγκριση με βάση τον γραμμικό συντελεστή συσχέτισης.

4.6 Πρόβλεψη δευτεροταγούς δομής

Ο προσδιορισμός στοιχείων δευτεροταγούς δομής αποτελεί ένα σημαντικό βήμα για τον χαρακτηρισμό των τρισδιάστατων πρωτεϊνικών δομών. Γενικά, η αναγνώριση δευτεροταγών δομών μέσω ενός αλγορίθμου είναι μια ιδιαίτερα πολύπλοκη διαδικασία. Γι' αυτόν τον λόγο, υπάρχει μια πληθώρα μεθόδων, καθεμία από τις οποίες προσεγγίζει το πρόβλημα με έναν διαφορετικό τρόπο. Μερικές από τις προσεγγίσεις αυτές είναι ο εντοπισμός μοτίβων μεταξύ των αποστάσεων των ατόμων C^{α}, η ανάλυση των γωνιών και του μήκους των δεσμών μεταξύ διαδοχικών ατόμων C^{α}, η ανάλυση μοτίβων για τους δεσμούς υδρογόνου, η σύγκριση των διατομικών πινάκων αποστάσεων δομικών τμημάτων με χαρακτηριστικά για κάθε δευτεροταγή δομή δεδομένα αναφοράς.

Στην περίπτωσή μας η ανάθεση της δευτεροταγούς δομής του τροχιακού του πεπτιδίου HP21 βασίστηκε στον αλγόριθμο STRIDE. Ο αλγόριθμος αυτός προσδιορίζει τα στοιχεία δευτεροταγούς δομής με τη συνδυασμένη χρήση της ενέργειας των δεσμών υδρογόνου και πληροφοριών που αφορούν τις δίεδρες γωνίες της κύριας αλυσίδας [103]. Παρακάτω παρουσιάζονται τα αποτελέσματα του STRIDE με χρωματική αναπαράσταση για όλο το τροχιακό καθώς επίσης και η ανάθεση δευτεροταγούς δομής ανά κατάλοιπο για όλες τις διαμορφώσεις τις προσομοίωσης και για τις διαμορφώσεις με θερμοκρασία adaptive tempering μικρότερη από 320 K με τη βοήθεια του προγράμματος WebLogo [106] (**Εικόνα 4.11**). Η αντιστοίχηση των αποχρώσεων με τα στοιχεία δευτεροταγούς δομής είναι η εξής:

- Α-έλικα (H) → ροζ
- Β-φύλλα (E) → κίτρινο
- 3-10 έλικα (G) → μωβ
- Στροφή (T) → μπλε
- Τυχαίο σπείραμα (C) → λευκό

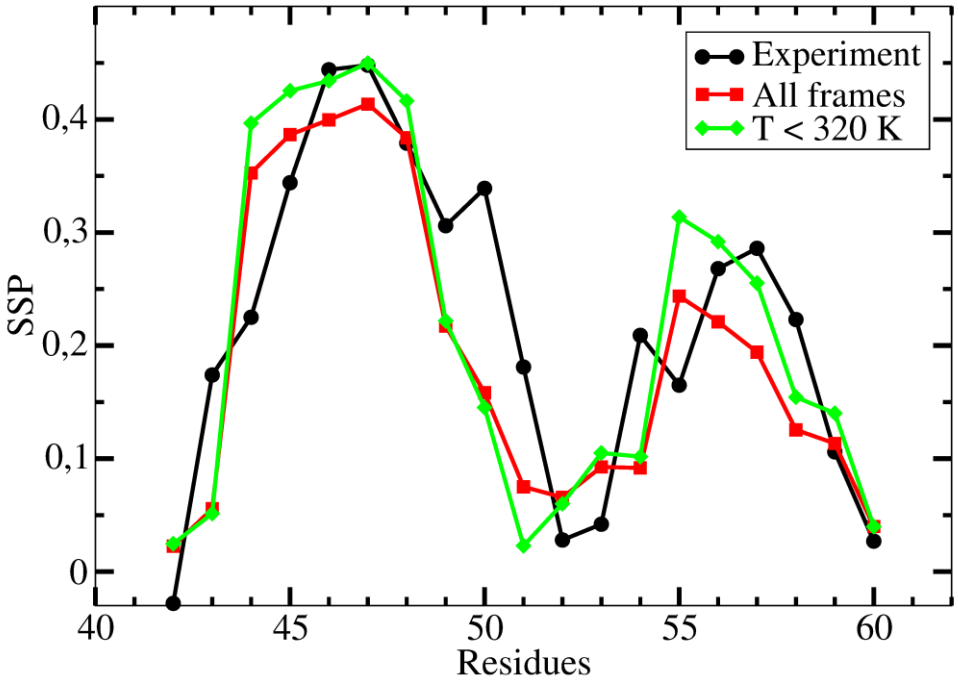


Εικόνα 4.11: Ανάθεση δευτεροταγούς δομής. Α. Χρωματική αναπαράσταση των στοιχείων δευτεροταγούς δομής σε συνάρτηση με το χρόνο της προσομοίωσης. Οι ροζ αποχρώσεις αντιστοιχούν σε α-έλικες, οι κίτρινες σε β-φύλλα, οι μωβ σε 3-10 έλικες, οι μπλε σε στροφές και οι λευκές σε τυχαία σπειράματα. Στα δεξιά της εικόνας έχουν σημανθεί τα όρια των δύο α-ελικών σύμφωνα με την πειραματικά προσδιορισμένη δομή του πεπτιδίου HP36. Β. Διάγραμμα ανάθεσης δευτεροταγούς δομής ανά κατάλοιπο με το WebLogo [106] για όλες τις διαμορφώσεις της προσομοίωσης, για το οποίο το γράμμα H αντιστοιχεί σε α-έλικα, το E σε β-φύλλα, το G σε 3-10 έλικα, το T σε στροφή και το C σε τυχαίο σπείραμα. Γ. Διάγραμμα ανάθεσης δευτεροταγούς δομής ανά κατάλοιπο με το WebLogo για τις διαμορφώσεις της προσομοίωσης με θερμοκρασία adaptive tempering μικρότερη από 320 K. Οι αντιστοιχίσεις των γραμμάτων με τα στοιχεία δευτεροταγούς δομής είναι οι ίδιες με το (Β).

Στηριζόμενοι στα παραπάνω αποτελέσματα μπορούμε ξεκάθαρα να διακρίνουμε μία σταθερή προτίμηση του πεπτιδίου για ελικοειδείς δευτεροταγείς δομές, γεγονός που ταυτίζεται με τα γνωστά πειραματικά

δεδομένα. Ειδικότερα, τα κατάλοιπα του αμινοτελικού άκρου (44-51), που απαρτίζουν την πρώτη έλικα, φαίνεται να υιοθετούν για το μεγαλύτερο μέρος της προσομοίωσης δομή α-έλικας. Παρόμοια είναι η κατάσταση και για τα κατάλοιπα της δεύτερης έλικας (55-60) εξαιρώντας το χρονικό διάστημα 2-3 μs, όπου υπάρχει μια σαφής τάση για σχηματισμό β-πτυχωτών φύλλων.

Ωστόσο, οι αναλύσεις μας δεν σταμάτησαν εδώ. Ορμώμενοι από την απόπειρα της ομάδας του Raleigh να ποσοτικοποιήσει την τάση σχηματισμού δευτεροταγούς δομής (secondary structure propensity – SSP) ανά κατάλοιπο του πεπτιδίου HP21 [74], πραγματοποιήσαμε τον ακόλουθο υπολογισμό: από τα αποτελέσματα ανάθεσης δευτεροταγούς δομής του STRIDE υπολογίσαμε το ποσοστό εμφάνισης της ανάθεσης α-έλικα ανά κατάλοιπο αφαιρώντας τις αναθέσεις που αντιστοιχούσαν στα β-φύλλα τόσο για όλες τις διαμορφώσεις της προσομοίωσης όσο και για τις διαμορφώσεις με θερμοκρασία adaptive tempering μικρότερη από 320 K. Στην **Εικόνα 4.12** περιέχεται μια γραφική αναπαράσταση των αποτελεσμάτων που προέκυψαν από την ανάλυση αυτή. Οι τιμές SSP ίσες με 1 καταδεικνύουν απόλυτη ροπή προς τη δομή α-έλικας, ενώ οι τιμές ίσες με -1 προς τη δομή β-φύλλων.



Εικόνα 4.12: Γραφική παράσταση των τιμών της τάσης δευτεροταγούς δομής (SSP) για τα κατάλοιπα 42-60 του πεπτιδίου HP21. Η μαύρη γραμμή αντιστοιχεί στα πειραματικά αποτελέσματα, η κόκκινη στα αποτελέσματα για όλες τις διαμορφώσεις της προσομοίωσης και η πράσινη στα αποτελέσματα για τις διαμορφώσεις της προσομοίωσης με θερμοκρασία adaptive tempering μικρότερη από 320 K.

Η σύγκλιση μεταξύ των πειραματικών δεδομένων και αυτών της προσομοίωσης είναι ολοφάνερη και στη συγκεκριμένη περίπτωση, αφού η προτίμηση για ελικοειδείς διαμορφώσεις συντηρείται και στην προσομοίωση. Πιο συγκεκριμένα, οι τιμές SSP για τα κατάλοιπα της πρώτης έλικας είναι αρκετά υψηλές φτάνοντας μέχρι και το 0.45 για το κατάλοιπο 47, ενώ για τα κατάλοιπα της δεύτερης έλικας εμφανίζουν μια μικρή πτώση με μέγιστη τιμή το 0.3 για το κατάλοιπο 55. Τέλος, όπως αναμενόταν, οι σταθερότερες διαμορφώσεις (T < 320 K) έχουν υψηλότερες τιμές SSP σε σχέση με το σύνολο του τροχιακού.

4.7 Ανάλυση κύριων συνιστωσών και ομαδοποίηση

Η ανάλυση κύριων συνιστωσών (PCA) αποτελεί μία στατιστική μέθοδο αναγνώρισης μοτίβων σε δεδομένα καθώς επίσης και έκφρασης των δεδομένων αυτών με τέτοιο τρόπο, ώστε να αναδεικνύονται τόσο οι ομοιότητες όσο και οι διαφορές τους. Η χρησιμότητας της ανάλυσης κύριων συνιστωσών έγκειται στην ανάλυση πολυδιάστατων δεδομένων, όπου η γραφική αναπαράσταση είναι αδύνατη, καθώς μειώνει τον αριθμό των διαστάσεων διατηρώντας αναλλοίωτο το μεγαλύτερο ποσοστό της παρεχόμενης πληροφορίας.

Η PCA χρησιμοποιείται ευρύτατα για την ανάλυση του μεγάλου όγκου δεδομένων που προέρχεται από προσομοιώσεις μοριακής δυναμικής, με στόχο την εξαγωγή βιολογικών πληροφοριών δίχως σημαντικές απώλειες. Γενικά, υπάρχουν δύο κατηγορίες ανάλυσης κύριων συνιστωσών που εφαρμόζονται σε τροχιακά μοριακής δυναμικής: στην cPCA (Cartesian PCA) η μείωση των διαστάσεων των δεδομένων βασίζεται στις καρτεσιανές

συντεταγμένες των ατόμων του συστήματος, ενώ στην dPCA (dihedral PCA) [107-109] στις δίεδρες γωνίες της κύριας αλυσίδας.

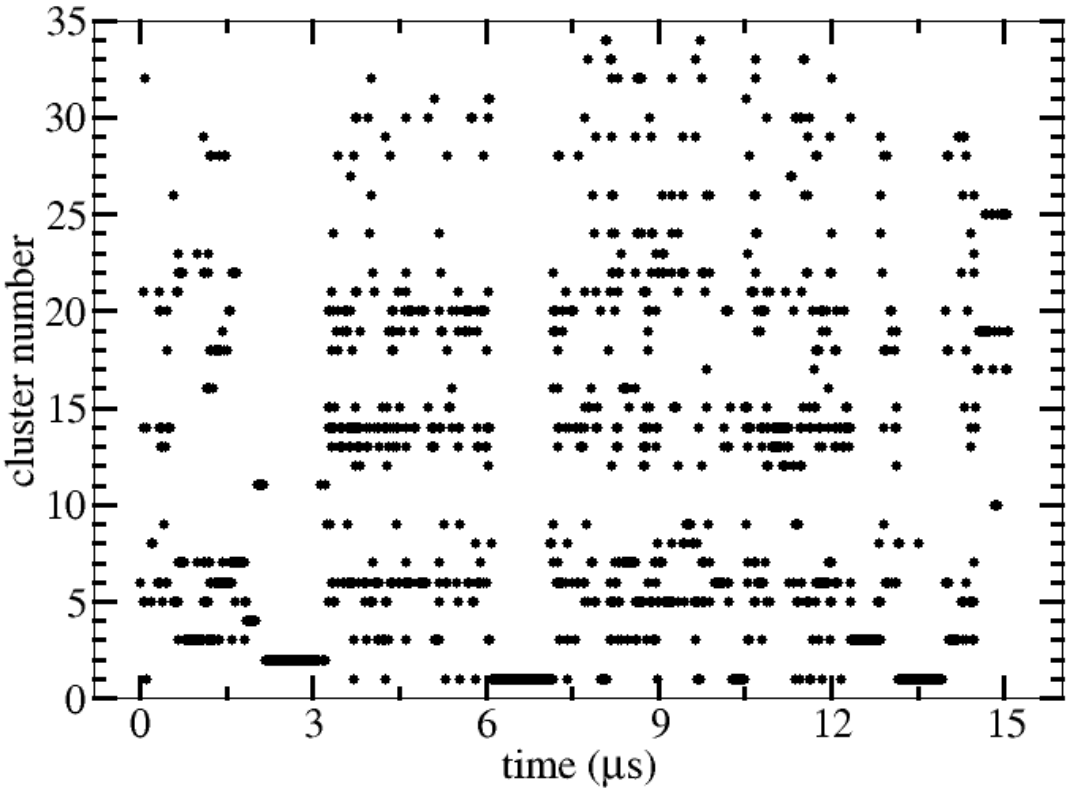
Η ομαδοποίηση (clustering) είναι η διαδικασία κατά την οποία γίνεται διαχωρισμός δεδομένων σε διακριτές ομάδες (clusters) λαμβάνοντας υπόψη τα κοινά τους χαρακτηριστικά. Με αυτόν τον τρόπο, καθίστανται ευκολότερη η κατανόηση και ανάδειξη της πληροφορίας που εμπεριέχεται στα δεδομένα.

Ο συνδυασμός της PCA και της ομαδοποίησης είναι μια από τις σημαντικότερες στρατηγικές για την ανάλυση τροχιακών μοριακής δυναμικής. Οι αναλύσεις PCA που ακολουθούν πραγματοποιήθηκαν με το πρόγραμμα CARMA και οι ομαδοποιήσεις με το cluster5D, ένα λογισμικό που ομαδοποιεί πενταδιάστατα δεδομένα που προέρχονται από ανάλυση PCA (**Παράρτημα 4**).

Αρχικά, επιλέξαμε να αναλύσουμε το τροχιακό μας με βάση τις δίεδρες γωνίες της κύριας αλυσίδας (dPCA) και όχι τις καρτεσιανές συντεταγμένες των ατόμων (cPCA), καθώς γνωρίζαμε ότι στο σύστημα επικρατεί αταξία για το μεγαλύτερο μέρος της προσομοίωσης. Τα αποτελέσματα συνοψίζονται στον παρακάτω πίνακα και την **Εικόνα 4.13**. Αξίζει να σημειωθεί πως για κάθε ομάδα που βρέθηκε έγινε υπολογισμός των δευτεροταγών χημικών μετατοπίσεων και στατιστική συσχέτιση με τα αντίστοιχα πειραματικά δεδομένα.

Αριθμός ομάδας	Πλήθος διαμορφώσεων	Reduced x^2			Γραμμικός συντελεστής συσχέτισης		
		$\Delta\delta^{13}C^{\alpha}$	$\Delta\delta^{13}C^{\beta}$	$\Delta\delta^{13}CO$	$\Delta\delta^{13}C^{\alpha}$	$\Delta\delta^{13}C^{\beta}$	$\Delta\delta^{13}CO$
1	2357930 (12.5 %)	1.03	0.68	0.40	0.71	0.78	0.84
2	1068469 (5.7 %)	3.80	1.38	3.38	0.30	0.57	0.46
3	1244992 (6.6 %)	2.76	2.15	2.16	0.25	-0.01	0.13
4	207213 (1.1 %)	2.88	2.61	3.64	0.64	0.44	0.48
5	168517 (0.9 %)	2.26	3.32	1.85	0.39	-0.13	0.19
6	232566 (1.2 %)	4.16	1.48	2.73	-0.11	0.27	0.26
7	280721 (1.5 %)	3.57	1.67	2.77	0.26	-0.07	0.15
8	43787 (0.23 %)	1.11	1.49	0.62	0.75	0.41	0.78

9	49092 (0.26 %)	2.38	1.27	1.74	0.63	0.52	0.51
10	41324 (0.22 %)	3.89	4.16	2.21	0.11	-0.42	0.07
11	35004 (0.18 %)	2.76	1.84	2.75	0.60	0.54	0.38
12	37508 (0.2 %)	2.34	1.67	1.34	0.36	0.10	0.42
13	50149 (0.26 %)	2.68	1.25	1.55	0.09	0.32	0.35
14	110283 (0.58 %)	1.97	1.56	1.50	-0.02	0.09	0.12
15	59774 (0.31 %)	2.26	1.87	1.18	0.47	0.14	0.56
16	29263 (0.15 %)	2.51	2.09	1.73	0.38	0.13	0.21
17	32070 (0.17 %)	2.75	3.85	1.87	0.34	-0.28	0.22
18	39155 (0.2 %)	4.14	1.71	2.90	-0.18	0.05	0.01
19	42343 (0.22 %)	4.64	3.77	3.27	-0.30	-0.44	-0.13
20	38166 (0.2 %)	4.97	1.08	2.62	-0.17	-0.05	0.12
21	20304 (0.1 %)	2.82	3.37	2.63	0.08	-0.07	-0.17
22	33737 (0.18 %)	2.52	1.85	1.77	0.44	0.11	0.43
23	12068 (0.064 %)	3.24	2.46	1.54	0.15	-0.37	0.35
24	8597 (0.045 %)	2.62	1.88	2.23	0.31	0.30	0.16
25	7126 (0.038 %)	3.38	3.11	2.58	0.34	-0.24	0.21
26	11929 (0.063 %)	2.36	2.85	1.60	0.36	-0.28	0.20
27	9534 (0.05 %)	4.70	2.31	4.05	0.33	0.12	0.05
28	5241 (0.028 %)	5.44	2.29	2.80	-0.35	0.24	-0.12
29	2533 (0.013 %)	2.37	1.83	2.27	0.53	0.22	0.32
30	370 (0.002 %)	1.28	1.12	1.45	0.63	0.34	0.40
31	2021 (0.01 %)	1.55	1.70	1.14	0.54	0.13	0.43
32	633 (0.003 %)	2.35	2.41	1.87	0.42	0.14	0.29
33	1874 (0.01 %)	1.87	2.47	1.48	0.54	-0.24	0.32
34	581 (0.003 %)	2.45	1.96	2.10	0.50	0.26	0.43
Σύνολο	6284874 από 18835852 (33.3 %)						



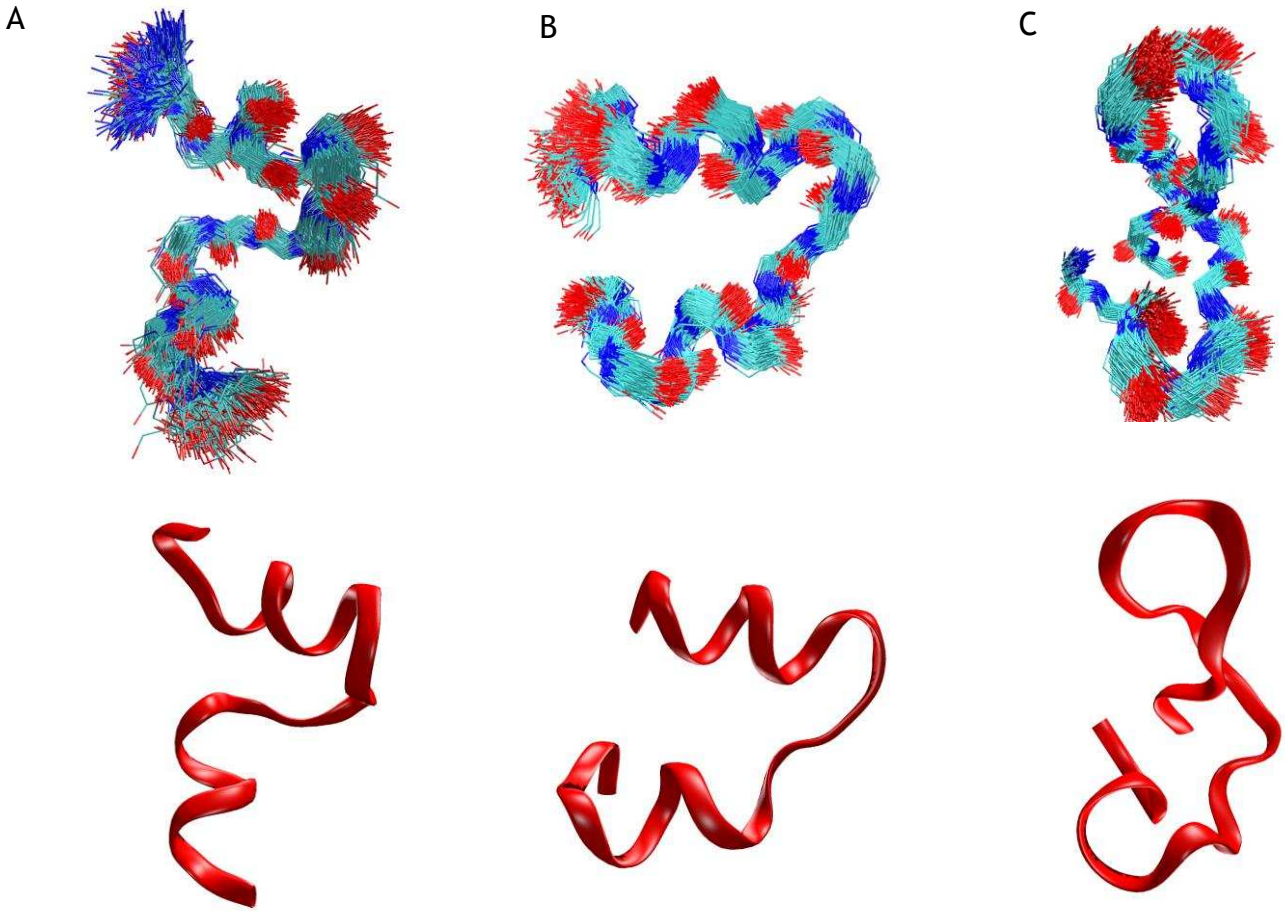
Εικόνα 4.13: Κατανομή των διαμορφώσεων για κάθε ομάδα μετά από dPCA σε ολόκληρο το τροχιακό του πεπτιδίου HP21 σε συνάρτηση με το χρόνο προσομοίωσης.

Όπως γίνεται φανερό, η ασταθής φύση του HP21 είχε ως συνέπεια την παραγωγή ενός μεγάλου αριθμού διακριτών ομάδων, οι περισσότερες από τις οποίες αποτελούνται από λίγες διαμορφώσεις. Γι' αυτό το λόγο, εστιάσαμε την προσοχή μας στις τρεις κυρίαρχες ομάδες που αντιστοιχούν σε ποσοστό περίπου 24% από το σύνολο των διαμορφώσεων. Συγκρίνοντας την κατανομή των διαμορφώσεων για κάθε ομάδα μετά από την dPCA (**Εικόνα 4.13**) με τον πίνακα RMSD όλου του τροχιακού (**Εικόνα 4.3**) παρατηρήσαμε ότι οι τρεις κυρίαρχες ομάδες περιέχουν τις δομές εκείνες που συμμετέχουν στα τρία σημαντικότερα γεγονότα αναδίπλωσης της προσομοίωσης. Πιο αναλυτικά, η ομάδα 1 (A) συνδέεται με τα γεγονότα αναδίπλωσης για τα χρονικά διαστήματα 6-7 μs, 10-10.5 μs και 13-14 μs, η ομάδα 2 (C) για τα χρονικό διάστημα 2-3 μs και, τέλος, η ομάδα 3 (B) για τα χρονικά διαστήματα 0-1 μs, 12-13 μs και 14-14.5 μs.

Το επόμενο βήμα ήταν η απομόνωση των σταθερότερων διαμορφώσεων που ανήκουν στις τρεις κυρίαρχες ομάδες, ώστε να αναγνωριστούν τα δομικά

χαρακτηριστικά της κάθε ομάδας και να προσδιοριστεί η ομοιότητά τους με τη φυσική διαμόρφωση. Για να πετύχουμε τον παραπάνω στόχο ακολουθήσαμε την εξής διαδικασία: αρχικά, επιλέχθηκαν οι διαμορφώσεις της προσομοίωσης με θερμοκρασία adaptive tempering μικρότερη από 300 K. Για τις διαμορφώσεις αυτές έγινε ανάλυση κύριων συνιστωσών με βάση τις δίεδρες γωνίες (dPCA) και ομαδοποίηση. Έπειτα, εντοπίστηκαν οι τρεις ομάδες, που αντιστοιχούν στις τρεις κυρίαρχες ομάδες A,B και C όλου του τροχιακού, και για καθεμία από τις ομάδες αυτές πραγματοποιήθηκε ανάλυση κύριων συνιστωσών με βάση τις καρτεσιανές συντεταγμένες των ατόμων της κύριας αλυσίδας (backbone cPCA) και ομαδοποίηση. Από τις ομάδες που παράχθηκαν συλλέχθηκε η πρώτη κυρίαρχη ομάδα και στις τρεις περιπτώσεις. Τέλος, ακολούθησε μία ακόμη ανάλυση κύριων συνιστωσών αυτή τη φορά με βάση τις καρτεσιανές συντεταγμένες όλων των ατόμων εκτός από τα υδρογόνα (heavy cPCA) και για τις τρεις κυρίαρχες ομάδες από την backbone cPCA ξεχωριστά. Το αποτέλεσμα της παραπάνω διαδικασίας ήταν η δημιουργία τριών αντιπροσωπευτικών ομάδων για τις A, B και C με πλήθος διαμορφώσεων 71800, 32774 και 87441 αντίστοιχα.

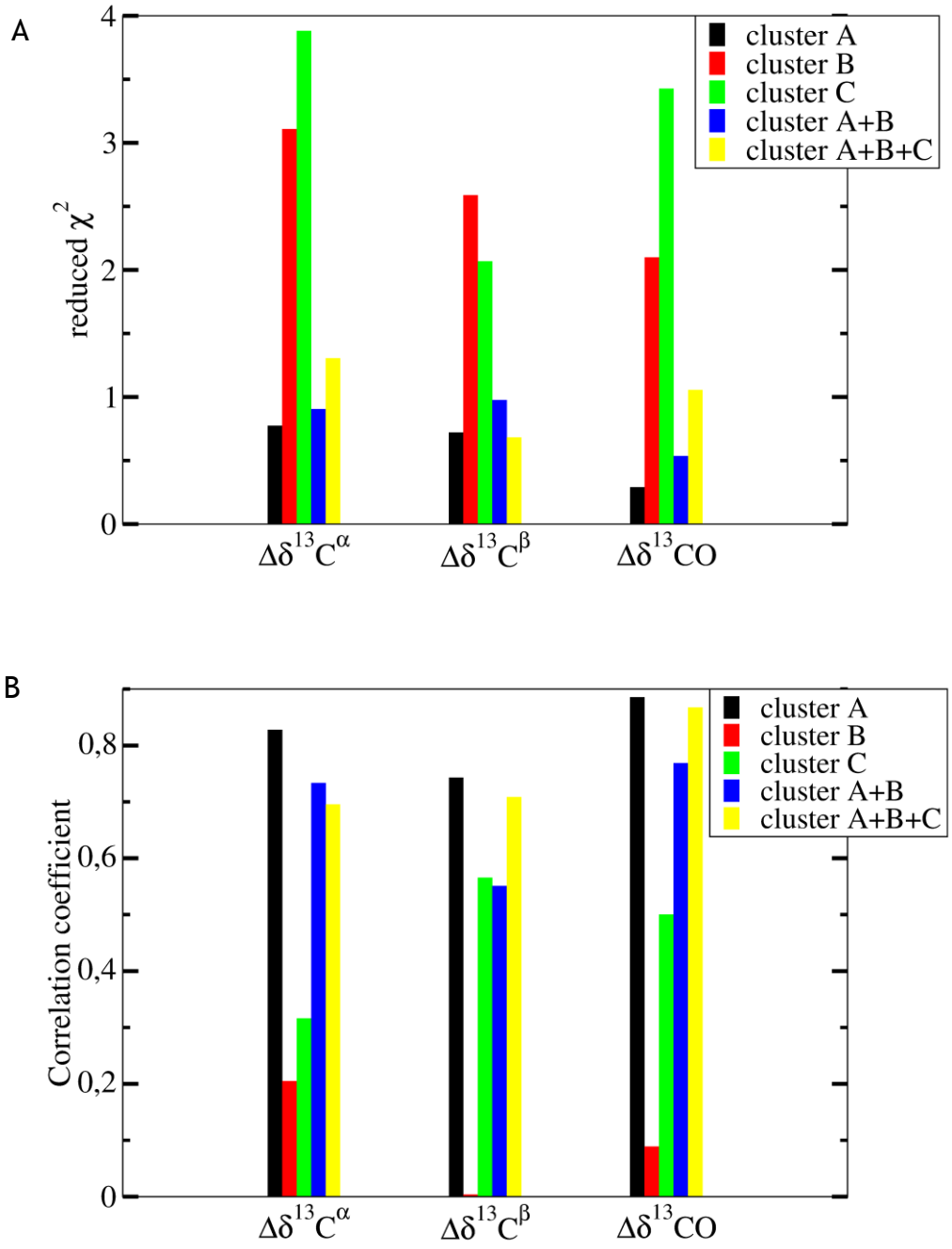
Στην **Εικόνα 4.14** παρουσιάζονται οι δομές των τριών αυτών κυρίαρχων ομάδων. Η ομάδα A αντιπροσωπεύει μία διαμόρφωση παρόμοια με τη φυσική δομή όπου επικρατεί το μοτίβο έλικα-στροφή-έλικα (helix-turn-helix) με παράλληλη διευθέτηση των δύο α-ελίκων. Η διαμόρφωση της ομάδας B είναι επίσης της μορφής έλικα-στροφή-έλικα με αντιπαράλληλη, όμως, διευθέτηση των δύο α-ελίκων. Σε αντίθεση με τις δύο παραπάνω κυρίαρχες ομάδες, η ομάδα C εμφανίζει μία αρκετά διαφορετική διαμόρφωση, η οποία είναι του τύπου α/β. Ειδικότερα, στο αμινοτελικό άκρο σχηματίζεται μια ασταθής α-έλικα, ενώ στο καρβοξυτελικό άκρο υπάρχει μια δομή β-φουρκέτας, η οποία, σύμφωνα και με το διάγραμμα του STRIDE, κάνει την εμφάνιση της μόνο στο χρονικό διάστημα 2-3 μs.



Εικόνα 4.14: Επάνω: Δομές των κυρίαρχων ομάδων A, B, και C (από αριστερά προς τα δεξιά) από την υπέρθεση 500 διαμορφώσεων. Κάτω: Αντιπροσωπευτικές δομές των κυρίαρχων ομάδων A,B και C (από αριστερά προς τα δεξιά).

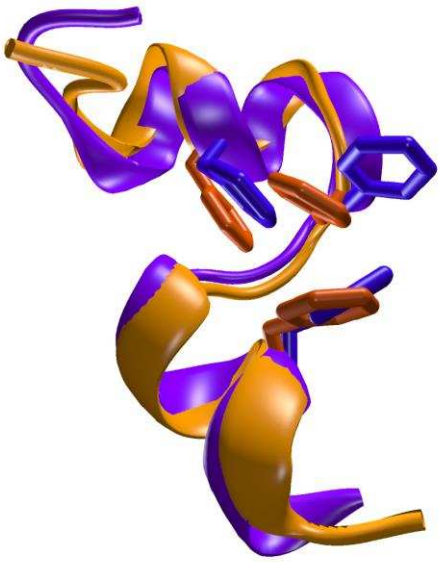
Τέλος επιχειρήσαμε να συγκρίνουμε τις δομές των ομάδων A, B και C τόσο με τις πειραματικά προσδιορισμένες δομές των HP36 (1VII) και HP (1YU5) υπολογίζοντας τις τιμές RMSD για άτομα της κύριας αλυσίδας (backbone) και για όλα τα άτομα εκτός από τα υδρογόνα (heavy) εξαιρώντας τα κατάλοιπα των άκρων (πίνακας παρακάτω) όσο και με την φυσική δομή του HP21 με τη βοήθεια των δευτεροταγών χημικών μετατοπίσεων (**Εικόνα 4.15**).

	1VII (κατάλοιπα 44-59)		1YU5 (κατάλοιπα 44-59)	
	Backbone	Heavy	Backbone	Heavy
Ομάδα A	0.9	1.9	1.2	2.0
Ομάδα B	3.9	5.6	3.8	5.5
Ομάδα C	4.9	6.4	4.9	6.2

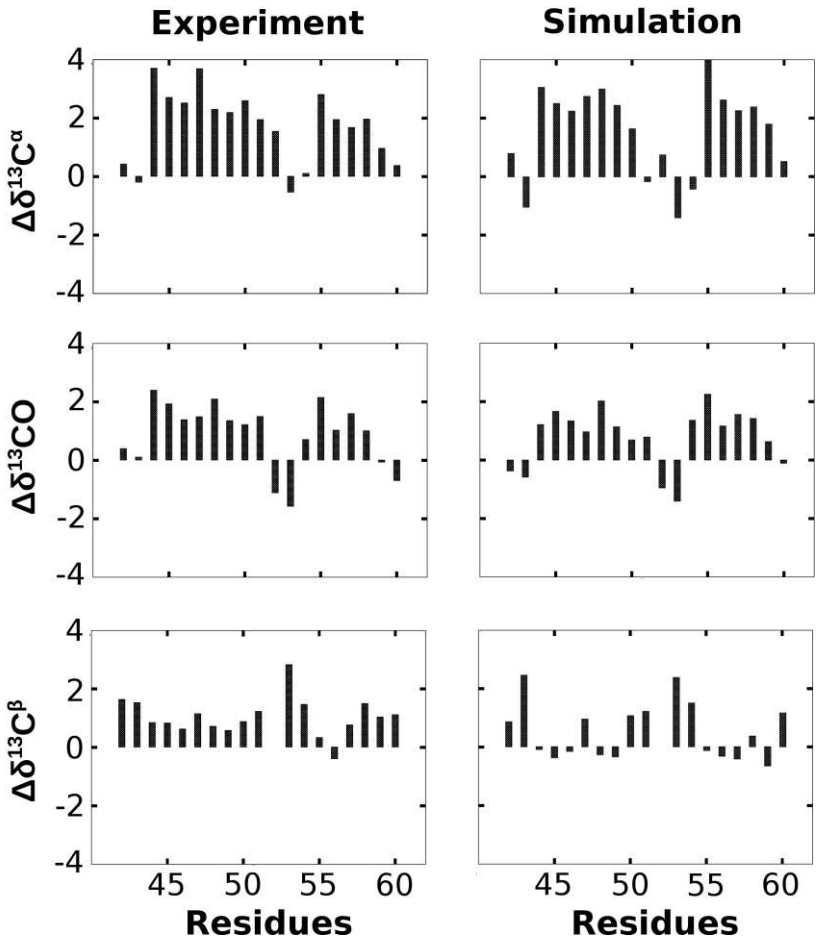


Εικόνα 4.15: Σύγκριση των τιμών των δευτεροταγών χημικών μετατοπίσεων των διαμορφώσεων των ομάδων A,B και C με τις τιμές των πειραματικών δευτεροταγών χημικών μετατοπίσεων του πεπτιδίου HP21 για τα άτομα C^{α}, C^{β} και CO των καταλοίπων 42-60. Α. Σύγκριση με βάση το στατιστικό δείκτη reduced x^2. Β. Σύγκριση με βάση το γραμμικό συντελεστή συσχέτισης.

Συμπερασματικά, οι διαμορφώσεις της ομάδας Α συμφωνούν ικανοποιητικά με τη φυσική διαμόρφωση σε επίπεδο δομών αλλά και δευτεροταγών χημικών μετατοπίσεων, γεγονός που αναδεικνύεται περαιτέρω στις **Εικόνες 4.16 και 4.17**.



Εικόνα 4.16: Υπέρθεση της πειραματικά προσδιορισμένης δομής του πεπτιδίου HP36 για τα κατάλοιπα 41-61 (πορτοκαλί απόχρωση) και της αντιπροσωπευτικής δομής της ομάδας Α της προσομοίωσης (μωβ απόχρωση). Στο κέντρο κάθε δομής υπάρχει ο υδρόφοβος πυρήνας αποτελούμενος από τρεις φαινυλαλανίνες (κατάλοιπα 47,51,58).



Εικόνα 4.17: Σύγκριση ανά κατάλοιπο μεταξύ των τιμών των πειραματικών δευτεροταγών χημικών μετατοπίσεων και των αντίστοιχων της ομάδας Α της προσομοίωσης για τα άτομα C^{α}, CO και C^{β} των καταλοίπων 42-60.

Συμπεράσματα και συζήτηση

Η μελέτη των μηχανισμών αναδίπλωσης συστημάτων που χαρακτηρίζονται από υψηλό βαθμό ευελιξίας με προσομοιώσεις μοριακής δυναμικής αποτελεί μια ιδιαίτερα επίπονη και, συχνά, ατελέσφορη διαδικασία. Το πεπτίδιο HP21, το οποίο χρησιμοποιήθηκε ως μοντέλο στην πτυχιακή αυτή εργασία, είναι ένα τέτοιου είδους σύστημα. Πιο συγκεκριμένα, το HP21 περιλαμβάνει τα κατάλοιπα που σχηματίζουν τις δύο πρώτες έλικες της υποεπικράτειας HP36 της πρωτεΐνης villin (41-61). Ο υπολογισμός των χημικών μετατοπίσεων με πειράματα NMR για το HP21 ανέδειξε την ικανότητα του πεπτιδίου να υιοθετεί σε υδατικό διάλυμα μία δομή παρόμοια με τη φυσική του διαμόρφωση, παρά την ασταθή φύση του [73,74].

Θέλοντας να εξετάσουμε τη συμπεριφορά αναδίπλωσης του HP21 με physics-based μεθόδους, πραγματοποιήσαμε μία προσομοίωση μοριακής δυναμικής εκτεταμένης χρονικής διάρκειας (15 μs) χρησιμοποιώντας το δυναμικό πεδίο AMBER99SB-STAR-ILDN. Τα αποτελέσματα έδειξαν ότι το πεπτίδιο είναι, όπως αναμενόταν, ασταθές για το μεγαλύτερο χρονικό διάστημα της προσομοίωσης εμφανίζοντας, ωστόσο, μεμονωμένα γεγονότα αναδίπλωσης. Μεταξύ των αναδιπλωμένων διαμορφώσεων διακρίθηκαν τρεις κυρίαρχες ομάδες: η πρώτη και σταθερότερη απ' αυτές αντιπροσωπεύει μια δομή που μοιάζει σε υψηλό βαθμό με την φυσική διαμόρφωση, κάτι που αποδείχθηκε τόσο με τις σημαντικά χαμηλές τιμές RMSD με τις πειραματικά προσδιορισμένες δομές της HP36 και της επικράτειας HP για τα αντίστοιχα κατάλοιπα όσο και με τη σύγκλιση των τιμών των πειραματικών δευτεροταγών χημικών μετατοπίσεων με τις αντίστοιχες της προσομοίωσης.

Αναφορικά με την μη αναδιπλωμένη κατάσταση του HP21, η μειωμένη επάρκεια του δείγματος από την προσομοίωση, σύμφωνα με τις εκτιμήσεις του Good Turing, λειτούργησε αποτρεπτικά στην μελέτη των ενδιάμεσων σταδίων της αναδίπλωσης. Παρολ' αυτά, μπορέσαμε να εξαγάγουμε δύο γενικά συμπεράσματα από τις αναλύσεις μας: το πρώτο έχει να κάνει με την προτίμηση για ελικοειδείς διαμορφώσεις και στα δύο άκρα του πεπτιδίου

καθ’ όλη τη διάρκεια της προσομοίωσης, με εξαίρεση το διάστημα 2-3 μs για τα κατάλοιπα του καρβοξυτελικού άκρου. Το δεύτερο αφορά τη μορφή του ενεργειακού τοπίου αναδίπλωσης του HP21, το οποίο φαίνεται να είναι αρκετά τραχύ αποτελούμενο από κινητικά εμπόδια που δεν επιτρέπουν τη σταθεροποίηση της φυσικής κατάστασης.

Βιβλιογραφικές Αναφορές

1. Branden C., Tooze J. (1999), Introduction to protein structure, Garland Publishing, New York
2. Mirsky A. E. and Pauling L. (1936), On the Structure of Native, Denatured, and Coagulated Proteins, Proc. Natl. Acad. Sci. USA 22: 439-447
3. Anfinsen C. B. (1973), Principles that Govern the Folding of Protein Chains, Science 181: 223-230
4. Haber E. and Anfinsen C. B. (1962), Side-chain interactions governing the pairing of half-cystine residues in ribonuclease, J. Biol. Chem. 237: 1839-1844
5. Levinthal C. (1968), Are there pathways of protein folding?, J. Chim. Phys. 65: 44-45
6. Levinthal C. (1969), How to fold graciously, Mossbaun Spectroscopy in Biological Systems Proceedings, Univ. of Illinois Bulletin 41: 22-24
7. Szilagyi A., Kardos J., Osvath S., Barna L., Zavodsky P. (2007), Protein Folding, Springer-Verlag Berlin Heidelberg
8. Honig B. (1999), Protein Folding: From Levinthal Paradox to Structure Prediction, J. Mol. Biol. 293: 283-293
9. Wedemeyer W. J. and Scheraga H. A. (2001), Protein Folding: Overview of Pathways, ENCYCLOPEDIA OF LIFE SCIENCES
10. Karplus M. (1997), The Levinthal paradox: yesterday and today, Elsevier 2: 69-75
11. Wetlaufer D. B. (1973), Nucleation, rapid folding, and globular intrachain regions in proteins, Proc. Natl. Acad. Sci. USA 70: 697-701
12. Karplus M. and Weaver D. L. (1976), Protein-folding dynamics, Nature 260: 404-406
13. Karplus M. and Weaver D. L. (1979), Diffusion-collision model for protein folding, Biopolymers 18: 1421-1437
14. Karplus M. and Weaver D. L. (1994), Folding dynamics: the diffusion-collision model and experimental data, Protein Sci. 3: 650-658
15. Fersht A. R. (1995), Optimization of rates of protein folding: The nucleation-condensation mechanism and its implications, Proc. Natl. Acad. Sci. USA 92: 10869-10873
16. Fersht A. R. (1997), Nucleation mechanisms in protein folding, Curr. Opin. Struct. Biol. 7: 3-9
17. Baldwin R. L. (1989), How does protein folding get started?, Trends Biochem. Sci. 14: 291-294
18. Dill K. A. (1985), Theory for the folding and stability of globular proteins, Biochemistry 24: 1501-1509
19. Dagget V. and Fersht A. R. (2003), Is there a unifying mechanism for protein folding?, Trends Biochem. Sci. 28: 18-25
20. Harrison S. C. and Durbin R. (1985), Is there a single pathway for the folding of a polypeptide chain?, Proc. Natl. Acad. Sci. USA 82: 4028-4030

21. Wolynes P. G., Onuchic J. N., Thirumalai D. (1995), Navigating the folding routes, Science 267: 1619-1620
22. Dill K. A. and Chan H. S. (1997), From Levinthal to pathways to funnels, Nature Structural Biology 4: 1
23. Dill K. A. and Chan H. S. (1998), Protein Folding in the Landscape Perspective: Chevron Plots and Non-Arrhenius Kinetics, PROTEINS: Structure, Function and Genetics 30: 2-33
24. Plaxco K. W. and Dobson C. M. (1996), Time-resolved biophysical methods in the study of protein folding, Current Opinion in Structural Biology 6: 630-636
25. Κοσσίδα Σ. (2008), Βιοπληροφορική Δυνατότητες και Προοπτικές, Εκδόσεις Νέων Τεχνολογιών Αθήνα
26. Dinner A. R., Sali A., Smith L. J., Dobson C. M., Karplus M. (2000), Understanding protein folding via free-energy surfaces from theory and experiment, TIBS 25: 331-339
27. Kelly S. M. and Price N. C. (2000), The Use of Circular Dichroism in the Investigation of Protein Structure and Function, Current Protein and Peptide Science 1: 349-384
28. Sreerama N. and Woody R. W. (2000), Estimation of Protein Secondary Structure from Circular Dichroism Spectra: Comparison of CONTIN, SELCON and CDSSTR Methods with an Expanded Reference Set, Analytical Biochemistry 287: 252-260
29. Eyles S. J. and Kaltashov I. A. (2004), Methods to study protein dynamics and folding by mass spectrometry, Methods 34: 88-99
30. Mertens H. D. T. and Svergun D. I. (2010), Structural characterization of proteins and complexes using small-angle X-ray solution scattering, Journal of Structural Biology 172: 128-141
31. Fabian H. and Naumann D. (2004), Methods to study protein folding by stopped-flow FT-IR, Methods 34: 28-40
32. Morrison R. T. and Boyd R. N. (1992), Organic Chemistry, Prentice Hall
33. Χαμόδρακας Σ. Ι. (1993), Θέματα Μοριακής Βιοφυσικής, Εκδόσεις Συμμετρία Αθήνα
34. Wishart D. S. (2011), Interpreting protein chemical shift data, Progress in Nuclear Magnetic Resonance Spectroscopy 58: 62-87
35. Mielke S. P. and Krishnan V. V. (2009), Characterization of protein secondary structure from NMR chemical shifts, Progress in Nuclear Magnetic Resonance Spectroscopy 54: 141-165
36. Wishart D. S. and Sykes B. D. (1994), The C-13 chemical-shift index – a simple method for the identification of protein secondary structure using C-13 chemical-shift data, J. Biomol. NMR 4: 171-180
37. Mielke S. P. and Krishnan V. V. (2003), Protein structural class identification directly from NMR spectra using averaged chemical shifts, Bioinformatics 19: 2054-2064
38. Vranken W. F. and Rieping W. (2009), Relationship between chemical shift value and accessible surface area for all amino acid atoms, BMC Struct. Biol. 9: 20

39. Shen Y., Delaglio F., Cornilescu G., Bax A. (2009), TALOS+: a hybrid method for predicting protein backbone torsion angles from NMR chemical shifts, J. Biolmol. NMR 44: 213-223
40. Berjanskii M. V. and Wishart D. S. (2007), The RCI server: rapid and accurate calculation of protein flexibility using chemical shifts, Nucleic Acids Res. 35: 531-537
41. Wishart D. S., Arndt D., Berjanskii M., Tang P., Zhou J., Lin G. (2008), CS23D: a web server for rapid protein structure generation using NMR chemical shifts and sequence data, Nucleic Acids Res. 36: 496-502
42. Richarz R. and Wuthrich K. (1978), Carbon-13 NMR chemical shifts of the common amino acid residues measured in aqueous solutions of the linear tetrapeptides H-Gly-Gly-X-LAla-OH, Biopolymers 17: 2133-2141
43. Bundi A. and Wuthrich K. (1979), ^{1}H NMR parameters of the common amino acid a/residues measured in aqueous solutions of the linear tetrapeptides H-Gly-Gly-X-LAla-OH, Biopolymers 18: 285-297
44. Wishart D. S., Bigam C. G., Holm A., Hodges R. S., Sykes B. D. (1995), H-1, C-13 and N-15 random coil NMR chemical shifts of the common amino acids. 1. Investigations of nearest-neighbor effects, J. Biomol. NMR 5: 67-81
45. Schwarzinger S., Kroon G. J. A., Foss T. R., Chung J., Wright P. E., Dyson H. J. (2001), Sequence-dependent correction of random coil NMR chemical shifts, J. Am. Chem. Soc. 123: 2970-2978
46. Baker D. and Sali A. (2001), Protein Structure Prediction and Structural Genomics, Science 294: 93-96
47. Holley H. L. and Karplus M. (1989), Protein secondary structure prediction with a neural network, Proc. Natl. Acad. Sci. USA 86: 152-156
48. Brindha S., Sailo S., Chhakchhuak L., Kalita P., Gurusubramanian G., Kumar S. N. (2011), Protein 3D structure determination using homology modeling and structure analysis, Sci. Vis. 11: 125-133
49. Rost B., Schneider R., Sander C. (1997), Protein Fold Recognition by Prediction-based Threading, J. Mol. Biol. 270: 471-480
50. Hardin C., Pogorelov T. V., Luthey-Schulten Z. (2002), Ab initio protein structure prediction, Current Opinion in Structural Biology 12: 176-181
51. Bonneau R., Tsai J., Ruczinski I., Chivian D., Rohl C., Strauss C. E. M., Baker D. (2001), Rosetta in CASP4: Progress in Ab Initio Protein Structure Prediction, PROTEINS: Structure, Function, and Genetics 5: 119-126
52. Moult J. (2005), A decade of CASP: progress, bottlenecks and prognosis in protein structure prediction, Current Opinion in Structural Biology 15:285-289
53. Karplus M. and Kuriyan J. (2005), Molecular dynamics and protein function, PNAS 102: 6679-6685
54. Zhang C. and Chou K. (1992), Monte Carlo simulation studies on the prediction of protein folding types from amino acid composition, Biophys. J. 63: 1523-1529
55. Rathore N. and de Pablo J. J. (2002), Monte Carlo simulation of proteins through a random walk in energy space, The Journal of Chemical Physics 116: 7225-7230

56. Baldwin R. L. and Rose G. D. (1999), Is protein folding hierarchic? I. Local structure and peptide folding, Trends in Biochemical Sciences 24: 26-33
57. Dyson H. J., Merutka G., Waltho J. P., Lerner R. A., Wright P. E. (1992), Folding of Peptide Fragments Comprising the Complete Sequence of Proteins: Models for Initiation of Protein Folding I. Myohemerythrin, J. Mol. Biol. 226: 795-817
58. Ho B. K. and Dill K. A. (2006), Folding Very Short Peptides Using Molecular Dynamics, PLoS Comput. Biol. 2: 0228:0237
59. Hornak V., Abel R., Okur A., Strockbine B., Roitberg A., Simmerling C. (2006), Comparison of Multiple Amber Force Fields and Development of Improved Protein Backbone Parameters, PROTEINS: Structure, Function and Bioinformatics 65: 712-725
60. Gnanakaran S. and Garcia A. E. (2003), Validation of an All-Atom Protein Force Field: From Dipeptides to Larger Peptides, J. Phys. Chem. B 107: 12555-12557
61. Oostenbrink C., Soares T. A., Van der Vegt N. F. A., Van Gunsteren W. F. (2005), Validation of the 53A6 GROMOS force field, Eur. Biophys. J. 34: 273-284
62. Williams P. D., Pollock D. D., Goldstein R. A. (2006), Functionality and the evolution of marginal stability in proteins: Inferences from lattice simulations, Evolutionary Bioinformatics Online 2: 91-101
63. Taverna D. M. and Goldstein R. A. (2002), Why Are Proteins Marginally Stable?, PROTEINS: Structure, Function and Genetics 46: 105-109
64. Tang K. E. S. and Dill K. A. (1998), Native protein fluctuations: the conformational motion temperature and the inverse correlation of protein flexibility with protein stability, J. Biomol. Struct. Dyn. 16: 397-411
65. Wright P. E. and Dyson H. J. (1999), Intrinsically Unstructured Proteins: Re-assessing the Protein Structure-Function Paradigm, J. Mol. Biol. 293: 321-331
66. Khurana S. and George S. P. (2008), Regulation of cell structure and function by actin-binding proteins: Villin's perspective, FEBS Letters 582: 2128-2139
67. Vardar D., Chishti A. H., Frank B. S., Luna E. J., Noegel A. A., Oh S. W., Schleicher M., McKnight C. J. (2002), Villin-Type Headpiece Domains Show a Wide Range of F-Actin-Binding Affinities, Cell Motility and the Cytoskeleton 52: 9-21
68. Tang Y., Grey M. J., McKnight J., Palmer A. G. III, Raleigh D. P. (2006), Multistate Folding of the Villin Headpiece Domain, J. Mol. Biol. 355: 1066-1077
69. Vardar D., Buckley D. A., Frank B. S., McKnight C. J. (1999), NMR Structure of an F-Actin-binding "Headpiece" Motif from Villin, J. Mol. Biol. 294: 1299-1310
70. Lei H., Su Y., Jin L., Duan Y. (2010), Folding Network of Villin Headpiece Subdomain, Biophysical Journal 99: 3374-3384
71. Wickstrom L., Okur A., Song K., Hornak V., Raleigh D. P., Simmerling C. L. (2006), The Unfolded State of the Villin Headpiece Helical Subdomain: Computational Studies of the Role of Locally Stabilized Structure, J. Mol. Biol. 360: 1094-1107

72. Hocking H. G., Hase F., Madl T., Zacharias M., Rief M., Zoldak G. (2015), A Compact Native 24-Residue Supersecondary Structure Derived from the Villin Headpiece Subdomain, Biophysical Journal 108: 678-686
73. Tang Y., Goger M. J., Raleigh D. P. (2006), NMR Characterization of a Peptide Model Provides Evidence for Significant Structure in the Unfolded State of the Villin Headpiece Helical Subdomain, Biochemistry 45: 6940-6946
74. Meng W., Shan B., Tang Y., Raleigh D. P. (2009), Native like structure in the unfolded state of the villin headpiece helical subdomain, an ultrafast folding protein, PROTEIN SCIENCE 18: 1692-1701
75. Allen M. P. (2004), Introduction to Molecular Dynamics Simulation, Computational Soft Matter: From Synthetic Polymers to Proteins NIC Series 23: 1-28
76. Stote R., Dejaegere A., Kuznetsov D., Falquet L. (1999), Theory of Molecular Dynamics Simulations, http://www.ch.embnet.org/MD_tutorial/
77. Izaguirre J. A., Catarello D. P., Wozniak J. M., Skeel R. D. (2001), Langevin stabilization of molecular dynamics, J. Chem. Phys. 114: 2090-2098
78. Alder B. J. and Wainwright T. E. (1957), Phase Transition for a Hard Sphere System, J. Chem. Phys. 27: 1208
79. Alder B. J. and Wainwright T. E. (1959), Studies in Molecular Dynamics. I. General Method, J. Chem. Phys. 31: 459-466
80. Rahman A. and Stillinger F. H. (1974), Improved Simulation of Liquid Water by Molecular Dynamics, J. Chem. Phys. 60: 1545-1557
81. McCammon J. A., Gelin B. R., Karplus M. (1977), Dynamics of folded proteins, Nature 267: 585-590
82. AMBER force field, http://ambermd.org/
83. CHARMM force field, http://www.charmm.org/
84. GROMOS force field, http://www.gromos.net/
85. Jorgensen W. L. and Tirado-Rives J. (1988), The OPLS Potential Functions for Proteins. Energy Minimizations for Crystals of Cyclic Peptides and Crambin, J. Am. Chem. Soc. 110: 1657-1666
86. Hornak V., Abel R., Okur A., Strockbine B., Roitberg A., Simmerling C. (2006), Comparison of Multiple Amber Force Fields and Development of Improved Protein Backbone Parameters, PROTEINS: Structure, Function and Bioinformatics 65: 712-725
87. Lindorff-Larsen K., Piana S., Palmo K., Maragakis P., Klepeis J. L., Dror R. O., Shaw D. E. (2010), Improved side-chain torsion potentials for the Amber ff99SB protein force field, PROTEINS: Structure, Function and Bioinformatics 78: 1950-1958
88. Best R. B. and Hummer G. (2009), Optimized molecular dynamics force fields applied to the helix-coil transition of polypeptides, J. Phys. Chem. B 113: 9004-9015
89. Phillips J. C., Braun R., Wang W., Gumbart J., Tajkhorshid E., Villa E., Chipot C., Skeel R. D., Kale L., Schulten K. (2005), Scalable Molecular Dynamics with NAMD, Journal of Computational Chemistry 26: 1781-1802, http://www.ks.uiuc.edu/Research/namd/
90. Glykos N. M. (2013), The Norma computing cluster, http://norma.mbg.duth.gr/index.php?id=about:intro

91. Srinivasan R., Ribosome – Program to build coordinates for peptides from sequence, http://folding.chemistry.msstate.edu/~raj/Manuals/ribosome.html
92. Zuo Z., Guo L., Mancera R. L. (2014), Free Energy Binding of Coiled-Coil Complexes with Different Electrostatic Environments: The Influence of Force Field Polarisation and Capping, Nat. Prod. Bioprospect. 4: 285-295
93. Case D. A., Cheatham III T. E., Darden T., Gohlke H., Luo R., Merz K. M. Jr., Onufriev A., Simmerling C., Wang B., Woods R. J. (2005), The Amber Biomolecular Simulation Programs, J. Comput. Chem. 26: 1668-1688
94. Zhang C. and Ma J. (2010), Enhanced sampling and applications in protein folding in explicit solvent, J. Chem. Phys. 132: 244101
95. Darden T., York D., Pedersen L. (1993), Paticle mesh Ewald: an N-log(N) method for Ewald sums in large systems, J. Chem. Phys. 98: 10089-10092
96. Ryckaert J., Ciccotti G., Berendsen H. J. C. (1977), Numerical integration of the Cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes, J. Comput. Phys. 23: 327-341
97. Glykos N. M. (2006), Carma: a molecular dynamics analysis program, J. Comput. Chem. 27: 1765-1768
98. Koukos P. I. and Glykos N. M. (2013), grcarma: A Fully Automated Task-Oriented Interface for the Analysis of Molecular Dynamics Trajectories, J. Comput. Chem. 34: 2310-2312
99. Humphrey W., Dalke A., Schulten K. (1996), VMD – Visual Molecular Dynamics, J. Molec. Graphics 14: 33-38, http://www.ks.uiuc.edu/Research/vmd/
100. Merritt E. A. and Bacon D. J. (1997), Raster3D Photorealistic Molecular Graphics, Methods Enzymol. 277: 505-524
101. Grace, http://plasma-gate.weizmann.ac.il/Grace/
102. Koukos P. I. and Glykos N. M. (2014), On the application of Good-Turing statistics to quantify convergence of biomolecular simulations, J. Chem. Inf. Model. 54: 209-217
103. Frishman D. and Argos P. (1995), Knowledge-Based Protein Secondary Structure Assignment, PROTEINS: Structure, Function and Genetics 23: 566-579
104. Shen Y. and Bax A. (2010), SPARTA+: a modest improvement in empirical NMR chemical shift prediction by means of an artificial neural network, J. Biomol. NMR 48: 13-22
105. Cho S. S., Levy Y., Wolynes P. G. (2006), P versus Q: Structural reaction coordinates capture protein folding on smooth landscapes, PNAS 103: 586-591
106. Crooks G. E., Hon G., Chandonia J. M., Brenner S. E. (2004), WebLogo: A Sequence Logo Generator, Genome Res. 14: 1188-1190
107. Mu Y., Nguyen P. H., Hegger R., Stock G. (2005), Energy Landscape of a Small Peptide Revealed by Dihedral Angle Principal Component Analysis, Proteins 58: 45-52
108. Altis A., Nguyen P. H., Hegger R., Stock G. (2007), Dihedral Angle Principal Component Analysis of Molecular Dynamics Simulations, J. Chem. Phys. 126: 244111

109. Altis A., Otten M., Nguyen P. H., Hegger R., Stock G. (2008), Construction of the free energy landscape of biomolecules via dihedral angle principal component analysis, J. Chem. Phys. 128: 245102

Παράρτημα 1

```
#
# Input files
#
amber                   on
readexclusions          yes
parmfile                hp21.prmtop
coordinates             heat_out.coor
velocities              heat_out.vel
extendedSystem          heat_out.xsc


#
# Adaptive ...
#
adaptTempMD             on
adaptTempTmin           280
adaptTempTmax           380
adaptTempBins           1000
adaptTempRestartFile    output/restart.tempering
adaptTempRestartFreq    10000
adaptTempLangevin       on
adaptTempRescaling      off
adaptTempOutFreq        400
adaptTempDt              0.000050


#
# Output files & writing frequency for DCD
# and restart files
#
outputname              output/equi_out
binaryoutput            off
restartname             output/restart
restartfreq             10000
binaryrestart           yes
dcdFile                 output/equi_out.dcd
dcdFreq                 400
DCDunitcell             yes


#
# Frequencies for logs and the xst file
#
outputEnergies          400
outputTiming            1600
xstFreq                 400

#
# Timestep & friends
#
timestep                2.0
stepsPerCycle           20
nonBondedFreq           1
fullElectFrequency      2

#
# Simulation space partitioning
#
switching               on
switchDist              7
cutoff                  8
pairlistdist            9
# twoAwayX                yes
```

```
#
# Basic dynamics
#
COMmotion               no
dielectric              1.0
exclude                 scaled1-4
1-4scaling              0.833333
rigidbonds              all

#
# Particle Mesh Ewald parameters.
#
Pme                     on
PmeGridsizeX            48                      # <===== CHANGE ME
PmeGridsizeY            48                      # <===== CHANGE ME
PmeGridsizeZ            48                      # <===== CHANGE ME


#
# Periodic boundary things
#
wrapWater               on
wrapNearest             on
wrapAll                 on



#
# Langevin dynamics parameters
#
langevin                on
langevinDamping         1
langevinTemp            320                     # <===== Check me
langevinHydrogen        off

langevinPiston          on
langevinPistonTarget    1.01325
langevinPistonPeriod    400
langevinPistonDecay     200
langevinPistonTemp      320                     # <===== Check me

useGroupPressure        yes

firsttimestep           10000                  # <===== CHANGE ME
run                     500000000               ;# <===== CHANGE ME
```

Παράρτημα 2

```
#!/usr/bin/perl -w

(@ARGV == 2) or die "Usage: calc_shifts <dcd> <psf>\n";

#
# How many shifts we will be collecting ?
#

(`carmanox -atmid ALLID -pdb -first 1 -last 1 $ARGV[0] $ARGV[1]` eq "" ) or
die "Carma made a boo-boo. Too bad ...\n";
`sparta+ -in $ARGV[0].*.pdb > /dev/null 2>&1`;
`/bin/rm -rf $ARGV[0].*.pdb $ARGV[1].*.pdb`;

open ( IN, "pred.tab" ) or die "Can not open pred.tab. Usage: calc_shifts
<dcd> <psf>\n";
while ( $line = <IN> )
  {
    if ( $line =~ /^FORMAT/ )
      {
        last;
      }
  }

$line = <IN>;
$tot = 0;
while ( $line = <IN> )
  {
    $ids[ $tot ] = substr( $line, 0, 14 );
    $tot++;
  }

close( IN );

`/bin/rm -rf *.tab`;

if ( $tot < 1 )
  {
    print "Too few atoms for calculating shifts. Something is wrong.
Bye.\n";
    exit;
  }


print "Will be collecting data for $tot atoms. Starting ...\n";

#
# Will do it in sets of 800 structures ...
#

`mkdir tmp1`;
`mkdir tmp2`;
`mkdir tmp3`;
`mkdir tmp4`;
`mkdir tmp5`;
`mkdir tmp6`;
`mkdir tmp7`;
`mkdir tmp8`;

$first = 1;

print "Now processing set starting at frame          ";
while( 1 )
{
```

```
printf("%8d", $first );

$last = $first + 99;
`cd tmp1 ; carmanox -atmid ALLID -pdb -first $first -last $last ../$ARGV[0]
../$ARGV[1]`;
`cd tmp1 ; sparta+ -in $ARGV[0].*.pdb > /dev/null 2>&1 &`;

$first += 100;
$last = $first + 99;
`cd tmp2 ; carmanox -atmid ALLID -pdb -first $first -last $last ../$ARGV[0]
../$ARGV[1]`;
`cd tmp2 ; sparta+ -in $ARGV[0].*.pdb > /dev/null 2>&1 &`;

$first += 100;
$last = $first + 99;
`cd tmp3 ; carmanox -atmid ALLID -pdb -first $first -last $last ../$ARGV[0]
../$ARGV[1]`;
`cd tmp3 ; sparta+ -in $ARGV[0].*.pdb > /dev/null 2>&1 &`;

$first += 100;
$last = $first + 99;
`cd tmp4 ; carmanox -atmid ALLID -pdb -first $first -last $last ../$ARGV[0]
../$ARGV[1]`;
`cd tmp4 ; sparta+ -in $ARGV[0].*.pdb > /dev/null 2>&1 &`;

$first += 100;
$last = $first + 99;
`cd tmp5 ; carmanox -atmid ALLID -pdb -first $first -last $last ../$ARGV[0]
../$ARGV[1]`;
`cd tmp5 ; sparta+ -in $ARGV[0].*.pdb > /dev/null 2>&1 &`;

$first += 100;
$last = $first + 99;
`cd tmp6 ; carmanox -atmid ALLID -pdb -first $first -last $last ../$ARGV[0]
../$ARGV[1]`;
`cd tmp6 ; sparta+ -in $ARGV[0].*.pdb > /dev/null 2>&1 &`;

$first += 100;
$last = $first + 99;
`cd tmp7 ; carmanox -atmid ALLID -pdb -first $first -last $last ../$ARGV[0]
../$ARGV[1]`;
`cd tmp7 ; sparta+ -in $ARGV[0].*.pdb > /dev/null 2>&1 &`;

$first += 100;
$last = $first + 99;
`cd tmp8 ; carmanox -atmid ALLID -pdb -first $first -last $last ../$ARGV[0]
../$ARGV[1]`;
`cd tmp8 ; sparta+ -in $ARGV[0].*.pdb > /dev/null 2>&1 &`;


$first += 100;

$procs = `ps -aef | grep 'sparta+' | wc -l`;
while( $procs > 2 )
  {
    sleep(1);
    $procs = `ps -aef | grep 'sparta+' | wc -l`;
  }

`cd tmp1 ; /bin/rm -rf $ARGV[0].*.pdb $ARGV[1].*.pdb *_struct.tab ; mv * ../
> /dev/null 2>&1`;
`cd tmp2 ; /bin/rm -rf $ARGV[0].*.pdb $ARGV[1].*.pdb *_struct.tab ; mv * ../
> /dev/null 2>&1`;
`cd tmp3 ; /bin/rm -rf $ARGV[0].*.pdb $ARGV[1].*.pdb *_struct.tab ; mv * ../
> /dev/null 2>&1`;
`cd tmp4 ; /bin/rm -rf $ARGV[0].*.pdb $ARGV[1].*.pdb *_struct.tab ; mv * ../
> /dev/null 2>&1`;
```

```
`cd tmp5 ; /bin/rm -rf $ARGV[0].*.pdb $ARGV[1].*.pdb *_struct.tab ; mv * ../
> /dev/null 2>&1`;
`cd tmp6 ; /bin/rm -rf $ARGV[0].*.pdb $ARGV[1].*.pdb *_struct.tab ; mv * ../
> /dev/null 2>&1`;
`cd tmp7 ; /bin/rm -rf $ARGV[0].*.pdb $ARGV[1].*.pdb *_struct.tab ; mv * ../
> /dev/null 2>&1`;
`cd tmp8 ; /bin/rm -rf $ARGV[0].*.pdb $ARGV[1].*.pdb *_struct.tab ; mv * ../
> /dev/null 2>&1`;


@files = glob("$ARGV[0]*.tab");

if ( @files == 0 )
  {
    last;
  }

foreach $file ( @files )
{

`tail -$tot $file | awk '{printf "%8.3f ", \$5}' >> SHIFTS`;
`echo >> SHIFTS`;
`tail -$tot $file | awk '{printf "%8.3f ", \$4}' >> SS_SHIFTS`;
`echo >> SS_SHIFTS`;
}

`/bin/rm -rf $ARGV[0]*.tab`;


}

`rmdir tmp1 tmp2 tmp3 tmp4 tmp5 tmp6 tmp7 tmp8`;
print "\n\n";

#
# Calculate means + sigmas
#
open ( IN, "SHIFTS" ) or die "Can not open SHIFTS ??? How did this happen
???\n";
open ( IN_SS, "SS_SHIFTS" ) or die "Can not open SS_SHIFTS ??? How did this
happen ???\n";
open ( OUT,'>', "OUTPUT") or die "Cannot open output file!\n";

for ( $i=0 ; $i < $tot ; $i++ )
{
    $mean= 0.0;
    $nof_lines = 0;
    $std = 0.0;

    $mean_SS = 0.0;
        $std_SS = 0.0;

    while ( defined($line = <IN>) && defined($line_SS = <IN_SS>) )
      {
        @data = split( ' ', $line );

        $nof_lines++;
        $delta = $data[ $i ] - $mean;
        $mean += $delta / $nof_lines;
        $std += $delta * ($data[ $i ] - $mean);

        @data_SS = split ( ' ', $line_SS );

        $delta_SS = $data_SS[ $i ] - $mean_SS;
        $mean_SS += $delta_SS / $nof_lines;
        $std_SS += $delta_SS * ($data_SS[ $i ] - $mean_SS);
      }
```

```
        printf OUT "%s    %8.4f %8.4f %8.4f %8.4f\n", $ids[ $i ], $mean,
sqrt( $std / ($nof_lines -1)), $mean_SS, sqrt( $std_SS / ($nof_lines -1));
        seek( IN, 0, 0 );
        seek ( IN_SS, 0, 0 );
}

close( OUT );
close( IN );
close (IN_SS);

print "\nAll done.\n\n";
```

Παράρτημα 3

Υπολογισμός reduced x^2:

```
#!/usr/bin/perl -w

open ( IN , "$ARGV[0]") or die "Usage: calc_chi-square <input_file>\n";
$sum = 0;
$num_of_lines= 0;
while ( $line = <IN> )
{
  @data = split (' ', $line);
  $col_num = @data;
  if ( $col_num == 3 )
  {
    $num_of_lines++;
    $subtract = $data[0] - $data[2];
    $val = $subtract * $subtract / $data[1] * $data[1];
    $sum += $val;
  }
  else
  {
    die "Not 3 columns in input file\n";
  }
}

close (IN);

print "The reduced chi-square value is\t", $sum / ($num_of_lines - 1), "\n";
```

Υπολογισμός γραμμικού συντελεστή συσχέτισης:

```
#!/usr/bin/perl -w


(@ARGV == 2) or die "Usage: calc_corr <file1> <file2>\n";
open (IN_1, "$ARGV[0]") or die "Cannot open <file1>\n";
open (IN_2, "$ARGV[1]") or die "Cannot open <file2>\n";

@file1 = <IN_1>;
@file2 = <IN_2>;

close (IN_1);
close (IN_2);

$N = @file1;
$sum1 = 0;
$sum2 = 0;

for ( $i = 0; $i < $N; $i++)
{
```

```
    $sum1 += $file1[$i];
    $sum2 += $file2[$i];
}

$mean1 = $sum1 / $N;
$mean2 = $sum2 / $N;

$sum_xy = 0;
$sum_x_square = 0;
$sum_y_square = 0;

for ( $i = 0; $i < $N; $i++)
{
    $x = $file1[$i] - $mean1;
    $y = $file2[$i] - $mean2;

    $xy = $x * $y;
    $sum_xy += $xy;
    $sum_x_square += $x * $x;
    $sum_y_square += $y * $y;
}


print "corr =\t", $sum_xy / sqrt($sum_x_square * $sum_y_square), "\n";
```

Παράρτημα 4

Πηγαίος κώδικας του προγράμματος ομαδοποίησης πενταδιάστατων δεδομένων από PCA - cluster5D (https://github.com/athbaltzis/cluster5D):

```
#include <stdio.h>
#include <math.h>
#include <dirent.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>

#define     YES     1

#define     NO      0

#define     DIMENSIONS      40

#define     MIN_ADD     1000000

#define     MAX_SD_POINTER      40


/*********************************/
/*                               */
/*Variable declarations          */
/*                               */
/*********************************/


clock_t begin, end;
double  time_spent;

int AUTO_FILENAME = NO;
int AUTO_FRACTION = YES;
int FRACTION = 0;
int VERBOSE = NO;

int
original_matrix[DIMENSIONS+1][DIMENSIONS+1][DIMENSIONS+1][DIMENSIONS+1][DIME
NSIONS+1];
int
matrix[DIMENSIONS+1][DIMENSIONS+1][DIMENSIONS+1][DIMENSIONS+1][DIMENSIONS+1]
;
int projection[DIMENSIONS+1][DIMENSIONS+1];
int DIM;
float cutoff;
void recursion (int, int, int, int, int);
void smoothing (int, int, int, int, int);
int add = MIN_ADD;
int main(int argc,char *argv[])
{

int count , count_dpca , count_pca ;
DIR *dir;
struct dirent   *dp;
char    line[10000];
```

```
int num;
FILE    *fp;
float   pca1,pca2,pca3,pca4,pca5;
float   pca1_max, pca1_min;
float   pca2_max, pca2_min;
float   pca3_max, pca3_min;
float   pca4_max, pca4_min;
float   pca5_max, pca5_min;
int have_limits;

int pointer1;
int pointer2;
int pointer3;
int pointer4;
int pointer5;

int i, k, j, l, m;
int value,value1;

float   sum;
float   N;
float   mean;
float   val;
float   sum_sd;
float   variance , sd;

int frames_count;

double  sd_pointer;
float   local_val;
float   local_sum_sd;
float   local_variance;

float   variance_explained;
float   variance_explained_array[200];
float   cluster_num_array[200];
float   func_array[200];
float   sd_pointer_array[200];
float   first_diff[200];
float   sec_diff[200];
float   third_diff[200];
float   val_diff;
float   max_third_diff;
float   max_variance_explained;
float   cluster_num_of_max;

int frames;
int all_frames;
int sum_frames = 0;
int cluster_frames;
int times;


int matrix_max;

FILE *op;
int cluster_num = 1;

int cluster_pixels;
float   func;

int max_cluster_frames = 0;
float   cutoff_number_of_frames = -1;

setlinebuf( stdout );

begin = clock();
```

```
/*************************************************************/
/*                                                           */
/*Arguments sanity checks, opening file                      */
/*                                                           */
/*************************************************************/


for (i = 0; i < argc; i++)
{
    if ( strncasecmp( argv[i], "-F", 2) == 0 )
    {
        if ( sscanf( argv[i+1], "%d", &FRACTION ) != 1 )
        {
            printf("Error : -fraction expects an integer argument\n");
            printf("Usage : cluster5D [-v] [-fract <integer>] [PCA
filename]\n");
            exit(1);
        }
        if ( FRACTION < 0 || FRACTION > 100 )
        {
            printf("Error : argument to -fraction should be an integer
between 0 and 100.\n");
            printf("Usage : cluster5D [-v] [-fract <integer>] [PCA
filename]\n");
            exit(1);
        }
        AUTO_FRACTION = NO;
        i++;
    }

    if ( strncasecmp (argv[i], "-V", 2) == 0 )
    {
        VERBOSE = YES;
    }
    if ( i == argc - 1)
    {
        if ( i == 0 )
        {
            AUTO_FILENAME = YES;
        }
        else
        {
            fp = fopen(argv[i], "r");
            if (fp == NULL)
            {

                AUTO_FILENAME = YES;
            }
        }
    }
}

if ( AUTO_FILENAME == YES )
{
    count = 0;
    count_dpca = 0;
    count_pca = 0;
    if ( (dir = opendir(".")) != NULL )
    {
        while ( (dp = readdir(dir)) != NULL)
        {
            if ( strcmp(dp->d_name,"carma.dPCA.fluctuations.dat") == 0 )
            {
                count_dpca++;
```

```
                count++;
            }
            if ( strcmp(dp->d_name,"carma.PCA.fluctuations.dat") == 0 )
            {
                count_pca++;
                count++;
            }
        }
        if ( count == 1 )
        {
            if ( count_dpca == 1)
            {
                fp = fopen("carma.dPCA.fluctuations.dat", "r");
                if (fp == NULL)
                {
                    printf ("Error: Cannot open file
carma.dPCA.fluctuations.dat\n");
                    exit(1);
                }
            }
            if ( count_pca == 1)
            {
                fp = fopen("carma.PCA.fluctuations.dat", "r");
                if (fp == NULL)
                {
                    printf ("Error: Cannot open file
carma.PCA.fluctuations.dat\n");
                    exit(1);
                }
            }
        }
        else
        if ( count == 2 )
        {
            printf ("Error: There are more than one PCA files in the current
folder. Please select one PCA file.\n");
            exit(1);
        }
        else
        {
            {
                printf ("Error: There is neither an argument for a PCA file
nor a PCA file in the current directory.\n");
                exit(1);
            }
        }
    }
}
while ( (fgets(line, sizeof(line), fp)) != NULL )
{
    if ( strlen(line) > 9999)
    {
        printf ("Error: Too big number of columns in the PCA file.\n");
        exit(1);
    }
    if ( (sscanf(line,"%d   %f   %f   %f   %f
%f",&num,&pca1,&pca2,&pca3,&pca4,&pca5)) != 6 )
    {
            printf ("Error: Invalid file. It must have at least 6
columns.\n");
            exit(1);
    }
}


/*****************************************************/
/*                                                   */
/*First pass to determine limits for each PCA row    */
```

```
/*                                                    */
/******************************************************/


rewind(fp);
have_limits = 0;
all_frames = 0;
if( VERBOSE == YES)
{
printf("First pass to determine limits ...\n");
printf("Now processing frame ");
}
while( ( fgets (line, sizeof(line), fp) ) != NULL )
{
    if ( (sscanf(line,"%d   %f   %f   %f   %f
%f",&num,&pca1,&pca2,&pca3,&pca4,&pca5)) == 6 )
    {
        all_frames++;
        if ( VERBOSE == YES)
        {printf("%8d\b\b\b\b\b\b\b\b",num);}
        if (have_limits == 0)
        {
            pca1_max = pca1;
            pca1_min = pca1;
            pca2_max = pca2;
            pca2_min = pca2;
            pca3_max = pca3;
            pca3_min = pca3;
            pca4_max = pca4;
            pca4_min = pca4;
            pca5_max = pca5;
            pca5_min = pca5;

            have_limits = 1;
        }

        if ( pca1 > pca1_max )
        pca1_max = pca1;
        if ( pca1 < pca1_min )
        pca1_min = pca1;
        if ( pca2 > pca2_max)
        pca2_max = pca2;
        if ( pca2 < pca2_min)
        pca2_min = pca2;
        if (pca3 > pca3_max)
        pca3_max = pca3;
        if (pca3 < pca3_min)
        pca3_min = pca3;
        if (pca4 > pca4_max)
        pca4_max = pca4;
        if (pca4 <pca4_min)
        pca4_min = pca4;
        if (pca5 > pca5_max)
        pca5_max = pca5;
        if (pca5 < pca5_min)
        pca5_min = pca5;
    }
}

if(VERBOSE == YES)
{printf("\n");
printf("%d\tframes will enter the calculation.\n", num);}


/******************************************************/
/*                                                    */
/*Second pass to populate the 5-dimensional matrix    */
/*                                                    */
```

```
/*******************************************************/

DIM = (int) ( pow(all_frames*2, 0.2 ) + 0.5);
rewind (fp);
if(VERBOSE == YES)
{printf("Second pass to populate the 5D matrix ...\n");
printf("Now processing frame ");}
while( ( fgets (line, sizeof(line), fp) ) != NULL )
{
    if ( (sscanf(line,"%d   %f   %f   %f   %f
%f",&num,&pca1,&pca2,&pca3,&pca4,&pca5)) == 6 )
    {
        if(VERBOSE == YES)
        {printf("%8d\b\b\b\b\b\b\b\b",num);}
        pointer1= (int)( ((pca1 - pca1_min) / (pca1_max - pca1_min)) * DIM +
0.5);
        pointer2= (int)( ((pca2 - pca2_min) / (pca2_max - pca2_min)) * DIM +
0.5);
        pointer3= (int)( ((pca3 - pca3_min) / (pca3_max - pca3_min)) * DIM +
0.5);
        pointer4= (int)( ((pca4 - pca4_min) / (pca4_max - pca4_min)) * DIM +
0.5);
        pointer5= (int)( ((pca5 - pca5_min) / (pca5_max - pca5_min)) * DIM +
0.5);

        original_matrix[pointer1][pointer2][pointer3][pointer4][pointer5]++;
    }
}

if(VERBOSE == YES)
{printf("\n");
fflush(stdout);}

/*****************************************/
/*                                       */
/*Smoothing                              */
/*                                       */
/*****************************************/

if(VERBOSE == YES)
{printf("Now smoothing ...");
fflush(stdout);}

for ( i=0 ; i <= DIM ; i++ )
for ( k=0 ; k <= DIM ; k++ )
for ( j=0 ; j <= DIM ; j++ )
for ( l=0 ; l <= DIM ; l++ )
for ( m=0 ; m <= DIM ; m++ )
{
    smoothing(i,k,j,l,m);
}

if(VERBOSE == YES)
{printf("\n");}


/*****************************************************/
/*                                                   */
/*Calculation of the density threshold               */
/*                                                   */
/*****************************************************/
```

```
/*************************************************************/
/*                                                           */
/*Calculation of mean, standard deviation and variance       */
/*                                                           */
/*************************************************************/


N = 0;
sum = 0.0;
mean = 0.0;
for ( i=0 ; i <= DIM ; i++ )
for ( k=0 ; k <= DIM ; k++ )
for ( j=0 ; j <= DIM ; j++ )
for ( l=0 ; l <= DIM ; l++ )
for ( m=0 ; m <= DIM ; m++ )
    {
        value = matrix[i][k][j][l][m];
        if ( value > 0 )
        {
            sum += value;
            N++;
        }
        if ( value > add )
        {
            printf("Error. Increase ... and recompile.\n");
            exit(1);
        }
    }
mean = sum / N;


sum_sd = 0.0;
val = 0.0;
for ( i=0 ; i <= DIM ; i++ )
for ( k=0 ; k <= DIM ; k++ )
for ( j=0 ; j <= DIM ; j++ )
for ( l=0 ; l <= DIM ; l++ )
for ( m=0 ; m <= DIM ; m++ )
    {
        value = matrix[i][k][j][l][m];
        if ( value > 0 )
        {
            val = ( value - mean);
            sum_sd += (val * val);
        }
    }
variance = sum_sd / N;
sd = sqrt (variance);

/*************************************************************************/
/*                                                                       */
/*Calculation of the density threshold from the given argument           */
/*                                                                       */
/*************************************************************************/

if ( AUTO_FRACTION == NO )
{
    if (VERBOSE == YES)
    {printf ("Testing density threshold: ");
    fflush(stdout);}
    frames =(int) ( (all_frames * FRACTION / 100) + 0.5 );
    frames_count = all_frames;
    times = 0;
    sd_pointer = 0.0;
    while ( frames <= frames_count )
    {
```

```
        cutoff = mean + ( sd_pointer * sd );
        frames_count = 0;
        for ( i=0 ; i <= DIM ; i++ )
        for ( k=0 ; k <= DIM ; k++ )
        for ( j=0 ; j <= DIM ; j++ )
        for ( l=0 ; l <= DIM ; l++ )
        for ( m=0 ; m <= DIM ; m++ )
        {
            value1 = original_matrix[i][k][j][l][m];
            value = matrix[i][k][j][l][m];
            if ( (float) value >= cutoff)
            {
                frames_count += value1;
            }
        }
        if (VERBOSE == YES)
        {printf("%10.2f\b\b\b\b\b\b\b\b\b\b", cutoff);
        fflush(stdout);}
        if ( frames >= frames_count )
        {
            break;
        }
        times++;
        sd_pointer = times * 0.10001;
    }
if (VERBOSE == YES)
{printf("\n");}
if (VERBOSE == YES)
{printf ("Density threshold set to %.2f.\n", cutoff);}
}



/*************************************************************/
/*                                                           */
/*Automatically calculation of the density threshold         */
/*                                                           */
/*************************************************************/

if ( AUTO_FRACTION == YES )
{
    for ( i=0 ; i <= DIM ; i++ )
    for ( k=0 ; k <= DIM ; k++ )
    for ( j=0 ; j <= DIM ; j++ )
    for ( l=0 ; l <= DIM ; l++ )
    for ( m=0 ; m <= DIM ; m++ )
    {
        original_matrix[i][k][j][l][m] = matrix[i][k][j][l][m];

    }

    op = fopen ("cluster5D_variance_explained.dat", "w+");
    times = 0;
    sd_pointer = 1.0;
    local_variance = variance ;
    variance_explained = 100 * (local_variance / variance);
    if (VERBOSE == YES)
    {printf ("Testing density threshold: ");
    fflush(stdout);}
    while ( sd_pointer <= 15.0 )
    {
        cutoff = mean + ( sd_pointer * sd );

        if (VERBOSE == YES)
        {printf("%10.2f\b\b\b\b\b\b\b\b\b\b", cutoff);
        fflush(stdout);}

        add = MIN_ADD;
```

```
local_sum_sd = 0.0;
cluster_num = 0;
for ( i=0 ; i <= DIM ; i++ )
for ( k=0 ; k <= DIM ; k++ )
for ( j=0 ; j <= DIM ; j++ )
for ( l=0 ; l <= DIM ; l++ )
for ( m=0 ; m <= DIM ; m++ )
{
    value = matrix[i][k][j][l][m];
    if ( (float) value >= cutoff )
    {
        local_val = (value - mean);
        local_sum_sd += (local_val * local_val);
    }
}

local_variance = local_sum_sd / N ;

matrix_max = matrix[0][0][0][0][0];
for ( i=0 ; i <= DIM ; i++ )
for ( k=0 ; k <= DIM ; k++ )
for ( j=0 ; j <= DIM ; j++ )
for ( l=0 ; l <= DIM ; l++ )
for ( m=0 ; m <= DIM ; m++ )
{
    value = matrix[i][k][j][l][m];
    if ( value > matrix_max )
    {
        matrix_max = value;
        pointer1 = i;
        pointer2 = k;
        pointer3 = j;
        pointer4 = l;
        pointer5 = m;
    }
}



while ( (float) matrix_max > cutoff )
{
    recursion (pointer1,pointer2,pointer3,pointer4,pointer5);
    cluster_num++;
    matrix_max = matrix[0][0][0][0][0];
    for ( i=0 ; i <= DIM ; i++ )
    for ( k=0 ; k <= DIM ; k++ )
    for ( j=0 ; j <= DIM ; j++ )
    for ( l=0 ; l <= DIM ; l++ )
    for ( m=0 ; m <= DIM ; m++ )
    {
        value = matrix[i][k][j][l][m];
        if ( value > matrix_max && value < MIN_ADD )
        {
            matrix_max = value;
            pointer1 = i;
            pointer2 = k;
            pointer3 = j;
            pointer4 = l;
            pointer5 = m;
        }
    }

    add += 1000000;

}


variance_explained = 100 * (local_variance / variance);
```

```
        func = variance_explained * cluster_num;
        func_array[times] = func;
        sd_pointer_array[times] = sd_pointer;
        variance_explained_array[times] = variance_explained;
        cluster_num_array[times] = cluster_num;

fprintf(op,"%4d\t%7.4f\t%7.2f\n",cluster_num,variance_explained,cutoff);
        times++;
        sd_pointer += 0.10001;
        for ( i=0 ; i <= DIM ; i++ )
        for ( k=0 ; k <= DIM ; k++ )
        for ( j=0 ; j <= DIM ; j++ )
        for ( l=0 ; l <= DIM ; l++ )
        for ( m=0 ; m <= DIM ; m++ )
        {
            matrix[i][k][j][l][m] = original_matrix[i][k][j][l][m];

        }
    }


    for ( i = 0 ; i <= times - 1 ; i++)
    {
        val_diff = ( func_array[i+1] - func_array[i] ) / (
sd_pointer_array[i+1] - sd_pointer_array[i] );
        first_diff[i] = val_diff;
    }

    for ( i = 0 ; i <= times - 2 ; i++)
    {
        val_diff = ( first_diff[i+1] - first_diff[i] ) / (
sd_pointer_array[i+1] - sd_pointer_array[i] );
        sec_diff[i] = val_diff;
    }

    for ( i = 0 ; i <= times - 3 ; i++)
    {
        val_diff = ( sec_diff[i+1] - sec_diff[i] ) / ( sd_pointer_array[i+1]
- sd_pointer_array[i] );
        third_diff[i] = val_diff;
    }

    max_third_diff = third_diff[0];
    sd_pointer = sd_pointer_array[0];
    max_variance_explained = 0.0;
    cluster_num_of_max = 0;

    for ( i = 0 ; i <= times - 3 ; i++)
    {
        val_diff = third_diff[i];
        if ( val_diff > max_third_diff )
        {
            max_third_diff = val_diff;
            cluster_num_of_max = cluster_num_array[i];
        }
    }

    for ( j = 0 ; j <= times - 3 ; j++)
    {
        if ( ( cluster_num_array[j] == cluster_num_of_max ) && (
variance_explained_array[j] > max_variance_explained ) )
                {
                    max_variance_explained = variance_explained_array[j];

                    sd_pointer = sd_pointer_array[j];
                }
    }
```

```
    cutoff =  mean + ( sd_pointer * sd );
    if (VERBOSE == YES)
    {printf("\n");}
    if (VERBOSE == YES)
    {printf ("Density threshold set to %.2f.\n", cutoff);}

    fclose(op);

    for ( i=0 ; i <= DIM ; i++ )
    for ( k=0 ; k <= DIM ; k++ )
    for ( j=0 ; j <= DIM ; j++ )
    for ( l=0 ; l <= DIM ; l++ )
    for ( m=0 ; m <= DIM ; m++ )
    {
        matrix[i][k][j][l][m] = original_matrix[i][k][j][l][m];

    }

}


cluster_num = 1;

/*************************/
/*                       */
/*Clustering             */
/*                       */
/*************************/



/*****************************************************/
/*                                                   */
/*Finding the pixel with the maximum value          */
/*                                                   */
/*****************************************************/


matrix_max = matrix[0][0][0][0][0];
    for ( i=0 ; i <= DIM ; i++ )
    for ( k=0 ; k <= DIM ; k++ )
    for ( j=0 ; j <= DIM ; j++ )
    for ( l=0 ; l <= DIM ; l++ )
    for ( m=0 ; m <= DIM ; m++ )
    {
        value = matrix[i][k][j][l][m];
        if ( value > matrix_max )
        {
            matrix_max = value;
            pointer1 = i;
            pointer2 = k;
            pointer3 = j;
            pointer4 = l;
            pointer5 = m;
        }
    }



/*****************************************************************/
/*                                                               */
/*Creating the output files and clustering using the recursive   */
/*function                                                       */
/*                                                               */
/*****************************************************************/

op = fopen ("carma.5-D.clusters.dat", "w+");
if (VERBOSE == YES)
```

```
{printf("Clustering now ...\n");}

while ( (float) matrix_max > cutoff )
{

    cluster_pixels = 0;
    cluster_frames = 0;
    recursion (pointer1,pointer2,pointer3,pointer4,pointer5);


/*****************************************************************/
/*                                                               */
/* PCA file pass to write the frames that belong to this cluster */
/*and to calculate the percentage of pixels and frames           */
/*                                                               */
/*****************************************************************/



    rewind(fp);
    while( ( fgets (line, sizeof(line), fp) ) != NULL )
    {
        if ( (sscanf(line,"%d   %f   %f   %f   %f
%f",&num,&pca1,&pca2,&pca3,&pca4,&pca5)) == 6 )
        {
            pointer1= (int)( ((pca1 - pca1_min) / (pca1_max - pca1_min)) *
DIM + 0.5);
            pointer2= (int)( ((pca2 - pca2_min) / (pca2_max - pca2_min)) *
DIM + 0.5);
            pointer3= (int)( ((pca3 - pca3_min) / (pca3_max - pca3_min)) *
DIM + 0.5);
            pointer4= (int)( ((pca4 - pca4_min) / (pca4_max - pca4_min)) *
DIM + 0.5);
            pointer5= (int)( ((pca5 - pca5_min) / (pca5_max - pca5_min)) *
DIM + 0.5);

            if ( -matrix[pointer1][pointer2][pointer3][pointer4][pointer5]
>= add + cutoff )
            {
                fprintf (op,"%10d  %10d  %10f   %10f   %10f   %10f
%10f\n", num,cluster_num,pca1,pca2,pca3,pca4,pca5);
                cluster_frames++;
            }
        }
    }
    for ( i=0 ; i <= DIM ; i++ )
    for ( k=0 ; k <= DIM ; k++ )
    for ( j=0 ; j <= DIM ; j++ )
    for ( l=0 ; l <= DIM ; l++ )
    for ( m=0 ; m <= DIM ; m++ )
    {
        value = -matrix[i][k][j][l][m];
        if ( value >= add + cutoff )
        {
            cluster_pixels++;
        }
    }
    if ( cluster_frames != 0)
    {
        if (VERBOSE == YES)
        {printf ("Cluster %5d located, contains %10d frames.\n",
cluster_num, cluster_frames);}
        sum_frames += cluster_frames;
        if ( cluster_frames > max_cluster_frames )
        {
            max_cluster_frames = cluster_frames;
        }
        if ( cluster_num == 20 )
```

```
        {
            cutoff_number_of_frames = max_cluster_frames / 10000;
        }
        if ( cutoff_number_of_frames > 0 && cluster_frames <
cutoff_number_of_frames )
        {
            if (VERBOSE == YES)
            {printf ("With several small (<%d frames) clusters following
...\n", (int) cutoff_number_of_frames);}
            end = clock();
            time_spent = (double) (end - begin) / CLOCKS_PER_SEC;
            if (VERBOSE == YES)
            {printf("All done in %.1f minutes.\n", time_spent / 60);}
            exit(1);
        }

        cluster_num++;
    }




/*****************************************************************/
/*                                                               */
/*Finding the pixel with the next maximum value and repeating    */
/*the above steps until the value of pixel reaches the threshold */
/*                                                               */
/*****************************************************************/


    matrix_max = matrix[0][0][0][0][0];
    for ( i=0 ; i <= DIM ; i++ )
    for ( k=0 ; k <= DIM ; k++ )
    for ( j=0 ; j <= DIM ; j++ )
    for ( l=0 ; l <= DIM ; l++ )
    for ( m=0 ; m <= DIM ; m++ )
    {
        value = matrix[i][k][j][l][m];
        if ( value > matrix_max && value < MIN_ADD )
            {
                matrix_max = value;
                pointer1 = i;
                pointer2 = k;
                pointer3 = j;
                pointer4 = l;
                pointer5 = m;
            }
    }
    add += 1000000;
}
    fclose(op);
    if ( AUTO_FRACTION == YES )
    {
        if (VERBOSE == YES)
        {printf("%.2f%% of frames have been clustered.\n", 100.0 *
sum_frames / all_frames);}
    }
```

```
/*************************************************************/
/*                                                           */
/*If the percentage of frames assigned to clusters is less than*/
/*10%, repeat the whole procedure without smoothing          */
/*************************************************************/

    if ( (100.0 * sum_frames / all_frames) < 10 )
    {
        if (VERBOSE == YES)
        {printf("Too few frames assigned to clusters.\nRepeating the
procedure without smoothing:\n");}

        sum_frames = 0;

        for ( i=0 ; i <= DIM ; i++ )
        for ( k=0 ; k <= DIM ; k++ )
        for ( j=0 ; j <= DIM ; j++ )
        for ( l=0 ; l <= DIM ; l++ )
        for ( m=0 ; m <= DIM ; m++ )
        {
            original_matrix[i][k][j][l][m] = 0;
        }

/*******************************************************/
/*                                                     */
/*Second pass to populate the 5-dimensional matrix     */
/*                                                     */
/*******************************************************/

        DIM = (int) ( pow(all_frames*2, 0.2 ) + 0.5);
        rewind (fp);
        if(VERBOSE == YES)
        {printf("Second pass to populate the 5D matrix ...\n");
        printf("Now processing frame ");}
        while( ( fgets (line, sizeof(line), fp) ) != NULL )
        {
            if ( (sscanf(line,"%d   %f   %f   %f   %f
%f",&num,&pca1,&pca2,&pca3,&pca4,&pca5)) == 6 )
            {
                if(VERBOSE == YES)
                {printf("%8d\b\b\b\b\b\b\b\b",num);}
                pointer1= (int)( ((pca1 - pca1_min) / (pca1_max - pca1_min))
* DIM + 0.5);
                pointer2= (int)( ((pca2 - pca2_min) / (pca2_max - pca2_min))
* DIM + 0.5);
                pointer3= (int)( ((pca3 - pca3_min) / (pca3_max - pca3_min))
* DIM + 0.5);
                pointer4= (int)( ((pca4 - pca4_min) / (pca4_max - pca4_min))
* DIM + 0.5);
                pointer5= (int)( ((pca5 - pca5_min) / (pca5_max - pca5_min))
* DIM + 0.5);


original_matrix[pointer1][pointer2][pointer3][pointer4][pointer5]++;
            }
        }

        if(VERBOSE == YES)
        {printf("\n");
        fflush(stdout);}



        for ( i=0 ; i <= DIM ; i++ )
        for ( k=0 ; k <= DIM ; k++ )
        for ( j=0 ; j <= DIM ; j++ )
```

```
        for ( l=0 ; l <= DIM ; l++ )
        for ( m=0 ; m <= DIM ; m++ )
        {
            matrix[i][k][j][l][m] = original_matrix[i][k][j][l][m];
        }
```

```
/*****************************************************************/
/*                                                               */
/*Calculation of the density threshold                           */
/*                                                               */
/*****************************************************************/
```

```
/*****************************************************************/
/*                                                               */
/*Calculation of mean, standard deviation and variance           */
/*                                                               */
/*****************************************************************/
```

```
        N = 0;
        sum = 0.0;
        mean = 0.0;
        for ( i=0 ; i <= DIM ; i++ )
        for ( k=0 ; k <= DIM ; k++ )
        for ( j=0 ; j <= DIM ; j++ )
        for ( l=0 ; l <= DIM ; l++ )
        for ( m=0 ; m <= DIM ; m++ )
        {
            value = matrix[i][k][j][l][m];
            if ( value > 0 )
            {
                sum += value;
                N++;
            }
            if ( value > add )
            {
                printf("Error. Increase ... and recompile.\n");
                exit(1);
            }
        }
        mean = sum / N;


        sum_sd = 0.0;
        val = 0.0;
        for ( i=0 ; i <= DIM ; i++ )
        for ( k=0 ; k <= DIM ; k++ )
        for ( j=0 ; j <= DIM ; j++ )
        for ( l=0 ; l <= DIM ; l++ )
        for ( m=0 ; m <= DIM ; m++ )
        {
            value = matrix[i][k][j][l][m];
            if ( value > 0 )
            {
                val = ( value - mean);
                sum_sd += (val * val);
            }
```

```
        }
        variance = sum_sd / N;
        sd = sqrt (variance);

/*****************************************************************/
/*                                                               */
/*Calculation of the density threshold from the given argument   */
/*                                                               */
/*****************************************************************/

        if ( AUTO_FRACTION == NO )
        {
            if (VERBOSE == YES)
            {printf ("Testing density threshold: ");
            fflush(stdout);}
            frames =(int) ( (all_frames * FRACTION / 100) + 0.5 );
            frames_count = all_frames;
            times = 0;
            sd_pointer = 0.0;
            while ( frames <= frames_count )
            {
                cutoff = mean + ( sd_pointer * sd );
                frames_count = 0;
                for ( i=0 ; i <= DIM ; i++ )
                for ( k=0 ; k <= DIM ; k++ )
                for ( j=0 ; j <= DIM ; j++ )
                for ( l=0 ; l <= DIM ; l++ )
                for ( m=0 ; m <= DIM ; m++ )
                {
                    value1 = original_matrix[i][k][j][l][m];
                    value = matrix[i][k][j][l][m];
                    if ( (float) value >= cutoff)
                    {
                        frames_count += value1;
                    }
                }
                if (VERBOSE == YES)
                {printf("%10.2f\b\b\b\b\b\b\b\b\b\b", cutoff);
                fflush(stdout);}
                if ( frames >= frames_count )
                {
                    break;
                }
                times++;
                sd_pointer = times * 0.10001;
            }
        if (VERBOSE == YES)
        {printf("\n");}
        if (VERBOSE == YES)
        {printf ("Density threshold set to %.2f.\n", cutoff);}
        }


/*****************************************************************/
/*                                                               */
/*Automatically calculation of the density threshold             */
/*                                                               */
/*****************************************************************/

        if ( AUTO_FRACTION == YES )
        {
            op = fopen ("cluster5D_variance_explained.dat", "w+");
            times = 0;
            sd_pointer = 0.0;
            local_variance = variance ;
            variance_explained = 100 * (local_variance / variance);
            if (VERBOSE == YES)
```

```
            {printf ("Testing density threshold: ");
            fflush(stdout);}
            while ( sd_pointer <= 15.0 )
            {
                cutoff = mean + ( sd_pointer * sd );

                if (VERBOSE == YES)
                {printf("%10.2f\b\b\b\b\b\b\b\b\b\b", cutoff);
                fflush(stdout);}

                add = MIN_ADD;
                local_sum_sd = 0.0;
                cluster_num = 0;
                for ( i=0 ; i <= DIM ; i++ )
                for ( k=0 ; k <= DIM ; k++ )
                for ( j=0 ; j <= DIM ; j++ )
                for ( l=0 ; l <= DIM ; l++ )
                for ( m=0 ; m <= DIM ; m++ )
                {
                    value = matrix[i][k][j][l][m];
                    if ( (float) value >= cutoff )
                    {
                        local_val = (value - mean);
                        local_sum_sd += (local_val * local_val);
                    }
                }

                local_variance = local_sum_sd / N ;

                matrix_max = matrix[0][0][0][0][0];
                for ( i=0 ; i <= DIM ; i++ )
                for ( k=0 ; k <= DIM ; k++ )
                for ( j=0 ; j <= DIM ; j++ )
                for ( l=0 ; l <= DIM ; l++ )
                for ( m=0 ; m <= DIM ; m++ )
                {
                    value = matrix[i][k][j][l][m];
                    if ( value > matrix_max )
                    {
                        matrix_max = value;
                        pointer1 = i;
                        pointer2 = k;
                        pointer3 = j;
                        pointer4 = l;
                        pointer5 = m;
                    }
                }



                while ( (float) matrix_max > cutoff )
                {
                    recursion
(pointer1,pointer2,pointer3,pointer4,pointer5);
                    cluster_num++;
                    matrix_max = matrix[0][0][0][0][0];
                    for ( i=0 ; i <= DIM ; i++ )
                    for ( k=0 ; k <= DIM ; k++ )
                    for ( j=0 ; j <= DIM ; j++ )
                    for ( l=0 ; l <= DIM ; l++ )
                    for ( m=0 ; m <= DIM ; m++ )
                    {
                        value = matrix[i][k][j][l][m];
                        if ( value > matrix_max && value < MIN_ADD )
                        {
                            matrix_max = value;
                            pointer1 = i;
                            pointer2 = k;
```

```
                        pointer3 = j;
                        pointer4 = l;
                        pointer5 = m;
                    }
                }

                add += 1000000;

            }


            variance_explained = 100 * (local_variance / variance);
            func = variance_explained * cluster_num;
            func_array[times] = func;
            sd_pointer_array[times] = sd_pointer;
            variance_explained_array[times] = variance_explained;
            cluster_num_array[times] = cluster_num;

fprintf(op,"%4d\t%7.4f\t%7.2f\n",cluster_num,variance_explained,cutoff);
            times++;
            sd_pointer += 0.10001;
            for ( i=0 ; i <= DIM ; i++ )
            for ( k=0 ; k <= DIM ; k++ )
            for ( j=0 ; j <= DIM ; j++ )
            for ( l=0 ; l <= DIM ; l++ )
            for ( m=0 ; m <= DIM ; m++ )
            {
                matrix[i][k][j][l][m] = original_matrix[i][k][j][l][m];

            }
        }


        for ( i = 0 ; i <= times - 1 ; i++)
        {
            val_diff = ( func_array[i+1] - func_array[i] ) / (
sd_pointer_array[i+1] - sd_pointer_array[i] );
            first_diff[i] = val_diff;
        }

        for ( i = 0 ; i <= times - 2 ; i++)
        {
            val_diff = ( first_diff[i+1] - first_diff[i] ) / (
sd_pointer_array[i+1] - sd_pointer_array[i] );
            sec_diff[i] = val_diff;
        }

        for ( i = 0 ; i <= times - 3 ; i++)
        {
            val_diff = ( sec_diff[i+1] - sec_diff[i] ) / (
sd_pointer_array[i+1] - sd_pointer_array[i] );
            third_diff[i] = val_diff;
        }

        max_third_diff = third_diff[0];
        sd_pointer = sd_pointer_array[0];
        max_variance_explained = 0.0;
        cluster_num_of_max = 0;

        for ( i = 0 ; i <= times - 3 ; i++)
        {
            val_diff = third_diff[i];
            if ( val_diff > max_third_diff )
            {
                max_third_diff = val_diff;
                cluster_num_of_max = cluster_num_array[i];
            }
        }
```

```
            for ( j = 0 ; j <= times - 3 ; j++)
            {
                if ( ( cluster_num_array[j] == cluster_num_of_max ) && (
variance_explained_array[j] > max_variance_explained ) )
                {
                    max_variance_explained = variance_explained_array[j];

                    sd_pointer = sd_pointer_array[j];
                }
            }

            cutoff =  mean + ( sd_pointer * sd );
            if (VERBOSE == YES)
            {printf("\n");}
            if (VERBOSE == YES)
            {printf ("Density threshold set to %.2f.\n", cutoff);}

            fclose(op);
        }

        for ( i=0 ; i <= DIM ; i++ )
        for ( k=0 ; k <= DIM ; k++ )
        for ( j=0 ; j <= DIM ; j++ )
        for ( l=0 ; l <= DIM ; l++ )
        for ( m=0 ; m <= DIM ; m++ )
        {
            matrix[i][k][j][l][m] = original_matrix[i][k][j][l][m];

        }

        cluster_num = 1;


        *************/
        /*           */
        /*Clustering*/
        /*           */
        *************/



/*****************************************************/
/*                                                   */
/*Finding the pixel with the maximum value           */
/*                                                   */
/*****************************************************/


        matrix_max = matrix[0][0][0][0][0];
            for ( i=0 ; i <= DIM ; i++ )
            for ( k=0 ; k <= DIM ; k++ )
            for ( j=0 ; j <= DIM ; j++ )
            for ( l=0 ; l <= DIM ; l++ )
            for ( m=0 ; m <= DIM ; m++ )
            {
                value = matrix[i][k][j][l][m];
                if ( value > matrix_max )
                {
                    matrix_max = value;
                    pointer1 = i;
                    pointer2 = k;
                    pointer3 = j;
                    pointer4 = l;
                    pointer5 = m;
                }
            }
```

```
/*****************************************************************/
/*                                                               */
/*Creating the output files and clustering using the recursive   */
/*function                                                       */
/*                                                               */
/*****************************************************************/

        op = fopen ("carma.5-D.clusters.dat", "w+");
        if (VERBOSE == YES)
        {printf("Clustering now ...\n");}

        while ( (float) matrix_max > cutoff )
        {

            cluster_pixels = 0;
            cluster_frames = 0;
            recursion (pointer1,pointer2,pointer3,pointer4,pointer5);


/*****************************************************************/
/*                                                               */
/*PCA file pass to write the frames that belong to this cluster  */
/*and to calculate the percentage of pixels and frames           */
/*                                                               */
/*****************************************************************/


            rewind(fp);
            while( ( fgets (line, sizeof(line), fp) ) != NULL )
            {
                if ( (sscanf(line,"%d   %f   %f   %f   %f
%f",&num,&pca1,&pca2,&pca3,&pca4,&pca5)) == 6 )
                {
                    pointer1= (int)( ((pca1 - pca1_min) / (pca1_max -
pca1_min)) * DIM + 0.5);
                    pointer2= (int)( ((pca2 - pca2_min) / (pca2_max -
pca2_min)) * DIM + 0.5);
                    pointer3= (int)( ((pca3 - pca3_min) / (pca3_max -
pca3_min)) * DIM + 0.5);
                    pointer4= (int)( ((pca4 - pca4_min) / (pca4_max -
pca4_min)) * DIM + 0.5);
                    pointer5= (int)( ((pca5 - pca5_min) / (pca5_max -
pca5_min)) * DIM + 0.5);

                    if ( -
matrix[pointer1][pointer2][pointer3][pointer4][pointer5] >= add + cutoff )
                    {
                        fprintf (op,"%10d  %10d  %10f   %10f   %10f   %10f
%10f\n", num,cluster_num,pca1,pca2,pca3,pca4,pca5);
                        cluster_frames++;
                    }
                }
            }
            for ( i=0 ; i <= DIM ; i++ )
            for ( k=0 ; k <= DIM ; k++ )
```

```
            for ( j=0 ; j <= DIM ; j++ )
            for ( l=0 ; l <= DIM ; l++ )
            for ( m=0 ; m <= DIM ; m++ )
            {
                value = -matrix[i][k][j][l][m];
                if ( value >= add + cutoff )
                {
                    cluster_pixels++;
                }
            }

            if (VERBOSE == YES)
            {printf ("Cluster %5d located, contains %10d frames.\n",
cluster_num, cluster_frames);}
            sum_frames += cluster_frames;
            if ( cluster_frames > max_cluster_frames )
            {
                max_cluster_frames = cluster_frames;
            }
            if ( cluster_num == 20 )
            {
                cutoff_number_of_frames = max_cluster_frames / 10000;
            }
            if ( cutoff_number_of_frames > 0 && cluster_frames <
cutoff_number_of_frames )
            {
                if (VERBOSE == YES)
                {printf ("With several small (<%d frames) clusters following
...\n", (int) cutoff_number_of_frames);}
                end = clock();
                time_spent = (double) (end - begin) / CLOCKS_PER_SEC;
                if (VERBOSE == YES)
                {printf("All done in %.1f minutes.\n", time_spent / 60);}
                exit(1);
            }
            cluster_num++;


/*****************************************************************/
/*                                                               */
/*Finding the pixel with the next maximum value and repeating    */
/*the above steps until the value of pixel reaches the threshold */
/*                                                               */
/*****************************************************************/


            matrix_max = matrix[0][0][0][0][0];
            for ( i=0 ; i <= DIM ; i++ )
            for ( k=0 ; k <= DIM ; k++ )
            for ( j=0 ; j <= DIM ; j++ )
            for ( l=0 ; l <= DIM ; l++ )
            for ( m=0 ; m <= DIM ; m++ )
            {
                value = matrix[i][k][j][l][m];
                if ( value > matrix_max && value < MIN_ADD )
                {
                    matrix_max = value;
                    pointer1 = i;
                    pointer2 = k;
                    pointer3 = j;
                    pointer4 = l;
                    pointer5 = m;
                }
            }
            add += 1000000;
        }
        fclose(op);
```

```
        if ( AUTO_FRACTION == YES )
        {
            if (VERBOSE == YES)
            {printf("%.2f%% of frames have been clustered.\n", 100.0 *
sum_frames / all_frames);}
        }




    }



    /*******************************/
    /*                             */
    /*End of repeating procedure   */
    /*                             */
    /*******************************/



    end = clock();
    time_spent = (double) (end - begin) / CLOCKS_PER_SEC;

    if (VERBOSE == YES)
    {printf("All done in %.1f minutes.\n", time_spent / 60);}
    return 0;
}


/*********************************************************************/
/*                                                                   */
/*The recursive function "recursion" checks the value of pixels      */
/*that are around the pixel with the maximum value.                  */
/*For these pixels that have value >= threshold, it checks           */
/*recursively the value of their around pixels                       */
/*                                                                   */
/*********************************************************************/



void recursion(int p1, int p2, int p3, int p4, int p5)
    {
        int i,k,j,l,m;
        int val;
        for ( i = p1 - 1; i <= p1 + 1; i++ )
        for ( k = p2 - 1; k <= p2 + 1; k++ )
        for ( j = p3 - 1; j <= p3 + 1; j++ )
        for ( l = p4 - 1; l <= p4 + 1; l++ )
        for ( m = p5 - 1; m <= p5 + 1; m++ )
        {
        if( i >= 0 && i <= DIM && k >= 0 && k <= DIM && j >= 0 && j <= DIM
&& l >= 0 && l <= DIM && m >= 0 && m <= DIM )
        {
            val = matrix[i][k][j][l][m];
            if ( (float) val >= cutoff && val < add )
            {
                val += add;
                matrix[i][k][j][l][m] = -val;
                recursion(i,k,j,l,m);
            }
        }
        }
    }
```

```
/*********************************/
/*                               */
/*Smoothing function             */
/*                               */
/*********************************/

void smoothing(int p1, int p2, int p3, int p4, int p5)
{
    int i,k,j,l,m;
    int val;
    int sum, count, average;
    sum = 0;
    count = 0;
    for ( i = p1 - 1; i <= p1 + 1; i++ )
    for ( k = p2 - 1; k <= p2 + 1; k++ )
    for ( j = p3 - 1; j <= p3 + 1; j++ )
    for ( l = p4 - 1; l <= p4 + 1; l++ )
    for ( m = p5 - 1; m <= p5 + 1; m++ )
    {
        if( i >= 0 && i <= DIM && k >= 0 && k <= DIM && j >= 0 && j <= DIM
&& l >= 0 && l <= DIM && m >= 0 && m <= DIM )
        {
            val = original_matrix[i][k][j][l][m];
            sum += val;
            count++;
        }
    }
    average = (int) ( sum / count + 0.5);
    matrix[p1][p2][p3][p4][p5] = average;
}
```