P.P

Prof. dr hab. Andrzej Molski

Prof. dr hab. Waldemar Nowicki

z prośbą o ocenę

Poznań, 21 czerwca 2011r.

Mateusz Najsztub

Odzyskiwanie parametrów kinetycznych rozplatania kinazy tytyny z symulacji dynamiki molekularnej

Retrieval of Kinetic Parameters of Titin Kinase Unfolding from Molecular Dynamics Simulations

Praca przedstawiona Komisji Egzaminacyjnej
Wydziału Chemii UAM w Poznaniu
celem uzyskania tytułu magistra chemii

Poznań, 21 czerwca 2011r.

Dziękuje profesorowi Andrzejowi Molskiemu za poświęcony czas oraz cierpliwość, dzięki którym napisanie tej pracy stało się możliwe.

Dziękuję rodzicom, którzy umożliwili mi te studia.

Dziękuję Kasi, Marcie, Ani i Madzi, które wprowadzały niezapomnianą atmosferę zarówno w pracy jak i po pracy...

Spis treści

Spis rysunków **6**

1 Wstęp **9**

2 Wprowadzenie **11**

2.1 Tytyna 11

2.2 AFM 13

2.3 Symulacje dynamiki molekularnej 14

2.3.1 Gromacs 16

2.3.2 NAMD 17

2.4 Model teoretyczny Hummera-Szabo 17

3 Cel pracy **22**

4 Metodologia pracy **24**

4.1 Budowa i konfiguracja klastra 24

4.1.1 Instalacja biblioteki OpenMPI 25

4.1.2 Instalacja biblioteki FFTW 26

4.1.3 Instalacja programu Gromacs 27

4.2 Przygotowanie struktury do symulacji 27

4.3 Równowagowanie 31

4.3.1 Minimalizacja energii 31

4.3.2 Równowagowanie rozpuszczalnika 33

4.3.3 Termostat 35
4.3.4 Barostat 36
4.3.5 Równowagowanie układu 37
4.4 Rozciąganie 40

5 Analiza wyników **45**
5.1 Znajdowanie siły rozplatania 45
5.2 Dopasowanie krzywej teoretycznej do wyników 47
5.2.1 Mechanizm postępowania 47
5.2.2 Analiza metody dopasowania 49
5.3 Analiza wydajności klastra 54

6 Wnioski **57**

Bibliografia **60**

Dodatek A. Spis zawartości dołączonej płyty CD **63**

Spis rysunków

2.1 Schemat budowy sarkomeru. 12
2.2 Model tytyny w sakromerze. 12
2.3 Wizualizacja struktury trzeciorzędowej domeny kinazy tytyny. 13
2.4 Schemat eksperymentu AFM. 14
2.5 Schemat ideowy zmiany powierzchni entalpii swobodnej pod wpływem rozciągania. 18

4.1 Schemat przebiegu równowagowania i symulacji. 31
4.2 Energia potencjalna układu podczas minimalizacji. 33
4.3 Zmiana RMSD, ciśnienia, objętości i temperatury w czasie podczas równowagowania. 38
4.4 Wizualizacja symulowanego układu z białkiem, atomami tlenu od wody w kolorze czerwonym, jonami sodu w kolorze żółtym oraz jonami chloru w kolorze jasnoniebieskim. 40

5.1 Wygładzone krzywe zależności siły od rozciągnięcia dla 6 prędkości rozciągania dla końców białka. 46
5.2 Wykresy siły rozplatania w zależności od szybkości rozciągania dla danych z symulacji (górny panel) oraz dla danych literaturowych (dolny panel) wraz z dopasowanymi krzywymi teoretycznymi. 50
5.3 Wykres krzywej teoretycznej z parametrami z pracy Schultena oraz po dopasowaniu z naniesionymi danymi z eksperymentu AFM. 53

5.4 Wykresy przepustowości oraz opóźnienia dla poszczególnych kart sieciowych oraz dla kart sieciowych działających razem w zależności od wielkości przesyłanych danych. 54
5.5 Wykres wydajności obliczeń w zależności od liczby użytych rdzeni oraz ich odchyleń od wartości teoretycznych. 56

Rozdział 1

Wstęp

Wraz z rozwojem techniki aparaturowej pojawiły się nowe możliwości badania procesów molekularnych na poziomie pojedynczej cząsteczki. Jednym z takich narzędzi jest mikroskop sił atomowych (AFM), który zostanie omówiony w dalszej części pracy. Nasuwa się pytanie czy możliwym jest uzyskanie wyników zbliżonych do danych eksperymentalnych tylko za pomocą symulacji komputerowych?

Temat tej pracy magisterskiej obejmuje próbę ekstrapolacji wyników symulacji rozplatania białka. Przedstawiona jest także próba odzyskania parametrów kinetycznych procesu rozplatania na podstawie symulacji. Obliczenia były prowadzone za pomocą pakietu do symulacji dynamiki molekularnej Gromacs [1]. Program ten był już wcześniej używany na Pracowni Dynamiki Procesów Molekularnych [2], jednak ze względu na ograniczenia wynikające z długiego czasu obliczeń oraz ilości miejsca potrzebnego na gromadzenie danych używanie go przy zasobach dostępnych w 2007 roku było kłopotliwe. Wraz z rozwojem tego programu oraz z zakupem coraz to nowszych komputerów pojawiła się szansa na przezwyciężenie tych ograniczeń.

W pracy przedstawione zostaną wyniki symulacji rozciągania pojedynczego modułu tytyny 1TKI oraz próba połączenia tych wyników z dostępnymi danymi uzyskanymi w wyniku poprzednich eksperymentów. Ponadto przedstawiona zostanie analiza wydajności klastra oraz perspektyw jego rozbudowy.

Na wstępie zostaną zaprezentowane metody eksperymentalne AFM oraz omó-

wione programy do symulacji dynamiki molekularnej. Następnie przedstawiony zostanie proces przygotowania oraz zbierania wyników gdzie opisano szczegółowo metodologię pracy. Osobny rozdział poświęcony jest analizie danych oraz procedurze dopasowania do nich krzywej teoretycznej.

Rozdział 2

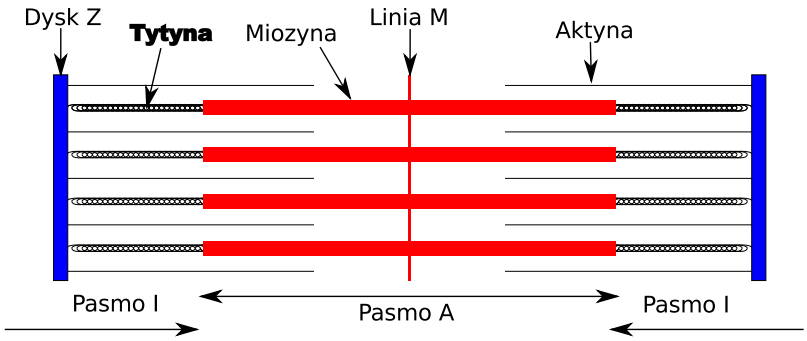
Wprowadzenie

2.1 Tytyna

Tytyna lub też inaczej konektyna jest największym poznanym do tej pory białkiem, które składa się z około 30000 aminokwasów. Stanowi ono jeden ze składników sarkomerów i odgrywa ważną rolę w ich pracy [3, 4].

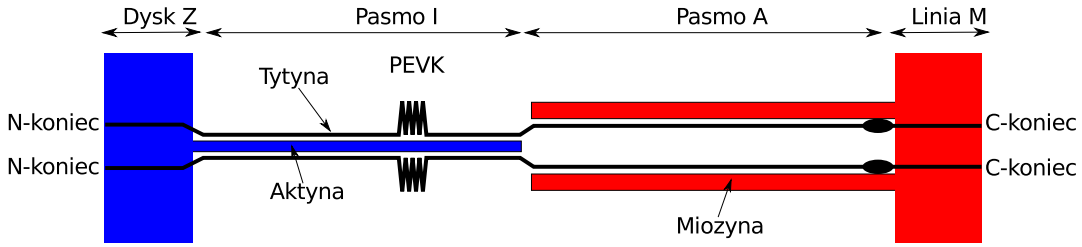
Sarkomer stanowi podstawowy element budulcowy mięśni poprzecznie prążkowanych i odpowiedzialny jest za ich kurczliwość. Na podstawie obrazu z mikroskopu optycznego podzielony został na pasma I (od izotropowe) oraz pasma A (od anizotropowe). Zbudowany jest z leżących naprzeciwko siebie filamentów aktyny oraz miozyny, które są przyłączone do dysku Z oraz linii M. Aktyna oraz miozyna odpowiadają za skurcze sarkomerów. Pojedyncza cząsteczka tytyny rozciąga się na długości około 1 μm od dysku Z do linii M sarkomeru. [5]

Tytyna składa się z około 300 powtórzeń domen fibrynokonektyny typu III (Fn-3), która znajduje się tylko w paśmie A oraz domen immunoglobuliny (Ig), które występują wzdłuż całej jej długości. W skład wchodzi także obszar PEVK w pasmie I, który nie posiada jednej, stabilnej struktury. Długość regionu PEVK oraz ilość domen Ig w paśmie I waha się znacznie w zależności od miejsca występowania mięśnia poprzecznie prążkowanego w organizmie. Dla przykładu w mięśniach szkieletowych



Rys. 2.1: Schemat budowy sarkomeru.

stwierdzono występowanie 90 domen Ig przy regionie PEVK składającym się z 2174 aminokwasów, kiedy w mięśniu sercowym wartości te wynoszą 37 domen Ig oraz 163 aminokwasów regionu PEVK [5].

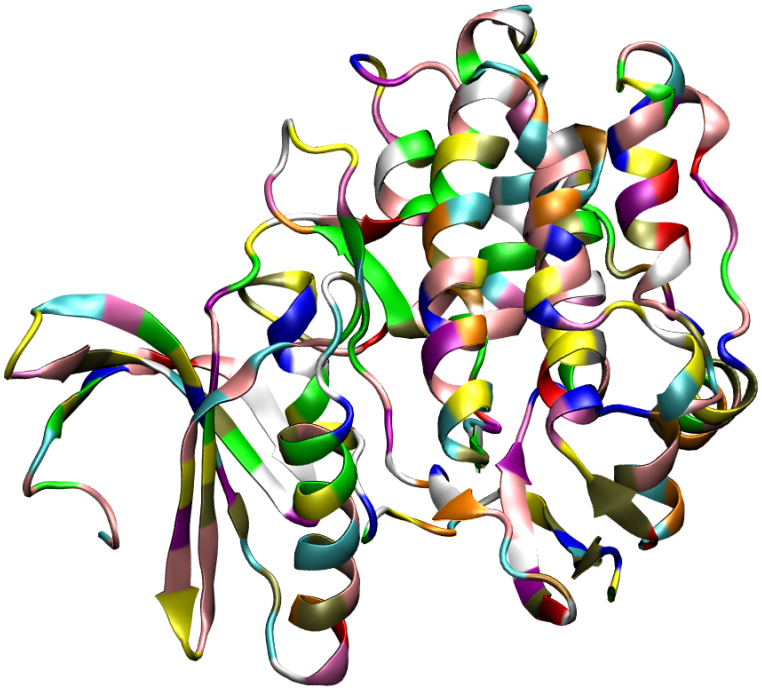


Rys. 2.2: Model tytyny w sakromerze.

Podczas skurczu mięśnia tytyna odpowiada za utrzymanie elementów budulcowych sarkomeru. Podczas rozciągania natomiast gromadzi w sobie powstające naprężenia, przeciwdziała zerwaniu oraz powoduje powrót do stanu wyjściowego po ustaniu zewnętrznej siły powodującej rozciągnięcie. Stwierdzono, że obszar PEVK odpowiada za elastyczność tytyny przy niskich rozciągnięciach. Po rozwinięciu tego obszaru stabilnie pofałdowane domeny Ig stawiają opór dalszemu rozciąganiu powodując wzrost naprężenia. Tego typu mechanizm podwójnej sprężyny wyjaśnia niejednorodny wzrost naprężenia przy rozciąganiu pojedynczego sarkomeru.

W tej pracy szczególna uwaga poświęcona została kinazie tytyny, której struktura została dokładnie zbadana [6] oraz istnieje szereg publikacji z danymi eksperymentów AFM oraz symulacji dynamiki molekularnej [7, 8]. Białko to znajduje się blisko C-końca tytyny, w linii M. Polipeptyd ten składa się z 321 aminokwasów. Podczas

powstawania naprężenia w mięśniu część tego naprężenia przekazywana jest na kinazę tytyny prowadząc do częściowego jej rozwinięcia, które powoduje odsłonięcie jej centrum aktywnego. Aktywna kinaza katalizuje fosforylację innego białka mięśniowego, telethoniny, które odgrywa rolę przy tworzeniu się mięśni, a także wpływa na ekspresję genów [9]. W ten sposób kinaza ta spełnia rolę molekularnego czujnika naprężenia.

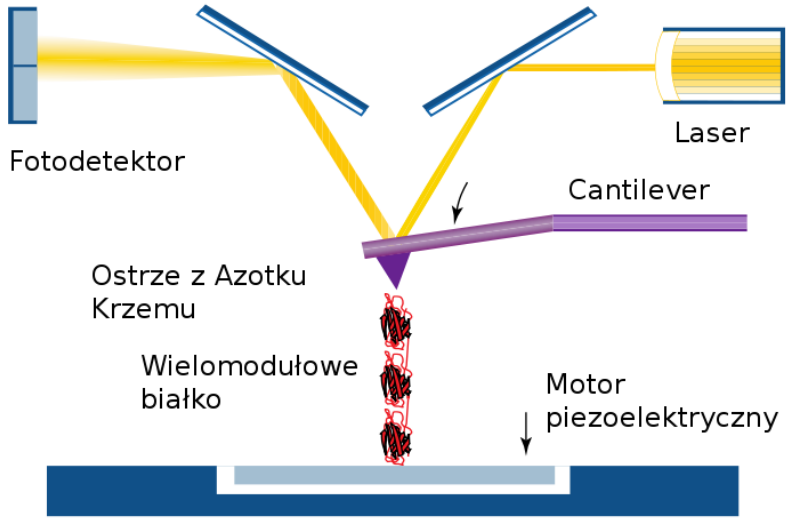


Rys. 2.3: Wizualizacja struktury trzeciorzędowej domeny kinazy tytyny.

2.2 AFM

Jednym z narzędzi, które pozwala na uzyskanie danych dla pojedynczych cząsteczek jest mikroskop sił atomowych (ang. Atomic Force Microscope — AFM). W omawianych w tej pracy eksperymentach mamy do czynienia z cząsteczkami przytwierdzonymi do ostrza mikrodźwigni (ang. cantilever) z jednej strony, a do podstawy z drugiej. Następuje ruch podstawy w pionie i mierzone jest odchylenie sprężystego

ramienia za pomocą wiązki laserowej. Na tej podstawie obliczana jest siła działająca na badaną cząsteczkę. W zależności od rodzaju eksperymentu możemy mieć do czynienia z rozciąganiem przy stałej sile, dzięki czemu można badać kinetykę reakcji w zależności od przyłożonego naprężenia [10] albo przy stałej szybkości rozciągania, gdzie można badać miedzy innymi siły zerwania bądź też rozplatania [11, 12].



Rys. 2.4: Schemat eksperymentu AFM.

2.3 Symulacje dynamiki molekularnej

Symulacje dynamiki molekularnej (**MD** – ang. Molecular Dynamics) stanowią jeden z działów chemii obliczeniowej (ang. Computational chemistry). W przeciwieństwie do symulacji Monte Carlo symulacje **MD** mogą być używane także do obliczeń układów nie będących w stanie równowagi oraz do wydarzeń dynamicznych, zachodzących w czasie. Z tego powodu symulacje **MD** są bardziej uniwersalne i znajdują zastosowanie w omawianym w tej pracy problemie [1].

Podstawowym zadaniem, które stoi przed tymi obliczeniami jest rozwiązanie równań ruchu Newtona (2.1) dla układu składającego się z N atomów oddziałujących na siebie zarówno za pomocą wiązań chemicznych jak i też za pomocą oddziaływań

elektrostatycznych i van der Waalsa.

$$m_i \frac{\partial^2 \vec{r_i}}{\partial t^2} = \vec{F_i}, \quad \mathrm{i} = 1 \ldots N \tag{2.1}$$

Gdzie m_i oznacza masę, $\vec{r_i}$ wektor położenia, a $\vec{F_i}$ siłę działającą na i-ty atom.

Dane na temat siły działającej na i-ty atom znajduje się obliczając ujemny gradient z funkcji potencjału $V(\vec{r_1}, \vec{r_2}, \ldots, \vec{r_N})$

$$\vec{F_i} = -\frac{\partial V(\vec{r_i})}{\partial \vec{r_i}} \tag{2.2}$$

Równania te rozwiązywane są numerycznie z małym krokiem czasowym. Co pewien czas w zależności od parametrów symulacji system jest kontrolowany pod względem temperatury oraz ciśnienia. Położenia oraz prędkości atomów są zapisywane w postaci trajektorii. Po początkowych dużych zmianach system po pewnym czasie osiąga stan równowagi.

Symulacje te posiadają jednak pewne ograniczenia, z których należy zdawać sobie sprawę przy przeprowadzaniu i planowaniu obliczeń.

Symulacje odbywają się według mechaniki klasycznej

Rozwiązując równania ruchu Newtona (2.1) pomijamy efekty kwantowe. Co za tym idzie symulacje **MD** nie nadają się do badania mechanizmów reakcji czy też stanów przejściowych. Istnieje jednak możliwość połączenia z programami do obliczeń kwantowomechanicznych, na przykład GAUSSIAN, jednak obliczenia te są bardzo czasochłonne.

Elektrony znajdują się w stanie podstawowym

Wykorzystywane pola siłowe zawierają definicje oddziaływań atomów w stanie podstawowym. W obliczeniach nie uwzględnia się ruchu elektronów. Z tego powodu wszelkie procesy związane z transferem elektronów oraz te, gdzie stan wzbudzony odgrywa kluczową rolę w eksperymencie nie mogą być w pełni badane.

Pola siłowe są tylko przybliżone

Pola siłowe zawierają opis oddziaływań pomiędzy atomami w różnych cząsteczkach. Nie uwzględniają jednak możliwości polaryzacji wiązań.

Oddziaływania są w pewnej odległości ucinane

Aby uprościć obliczenia oraz aby zapobiec nakładaniu się tych samych oddziaływań przez periodyczność stosuje się ograniczenie ich przy pewnej zdefiniowanej przez parametry symulacji odległości.

Periodyczność pudła

Aby zapobiec rozpraszaniu się atomów do próżni oraz aby uniknąć nienaturalnie małego układu badawczego wprowadzono periodyczność. Przez periodyczność w tym przypadku należy rozumieć fakt, że jeśli atom wybiega poza granice układu, to automatycznie pojawia się on po przeciwnej stronie. W ten sposób przenoszone są także oddziaływania pomiędzy atomami. Częściowo zbliża to badany układ do kryształu, co może powodować błędy w obliczeniach.

2.3.1 Gromacs

Gromacs stanowi jeden z wielu dostępnych programów do symulacji dynamiki molekularnej, jednak ze względu na ilość materiałów szkoleniowych [13] i ilość publikacji, w których występuje, właśnie to narzędzie zostało użyte do obliczeń.

Gromacs stanowi pakiet programów do przygotowania symulacji, ich przeprowadzenia oraz analizy wyników. Dostępny jest bezpłatnie, na licencji GNU General Public License. Napisany został przez grupę Hermana Berendsensena (Department of Biophysical Chemistry of Groningen University). Obecnie rozwijany jest przez Erika Lindahla (Stockholm Center for Biomembrane Research, SE), Davida van der Spoela (Biomedical Centre, Uppsala, SE) oraz Berka Hessa (Stockholm Center for Biomembrane Research, Stockholm, SE).

Program został domyślnie napisany na systemy Uniksowe, jednak dostępny jest

również pod system Windows. Spośród innych narzędzi wyróżnia się przede wszystkim wydajnością wynikającą ze zoptymalizowanego kodu, który wykorzystuje także multimedialne instrukcje procesorów jak MMX czy SSE (ang. Streaming SIMD Extensions) i SSE2. Przy wykorzystaniu bibliotek MPI (ang. Message Passing Interface) możliwe jest wykonywanie obliczeń na kilku jednostkach jednocześnie, co zdecydowanie skraca czas obliczeń.

2.3.2 NAMD

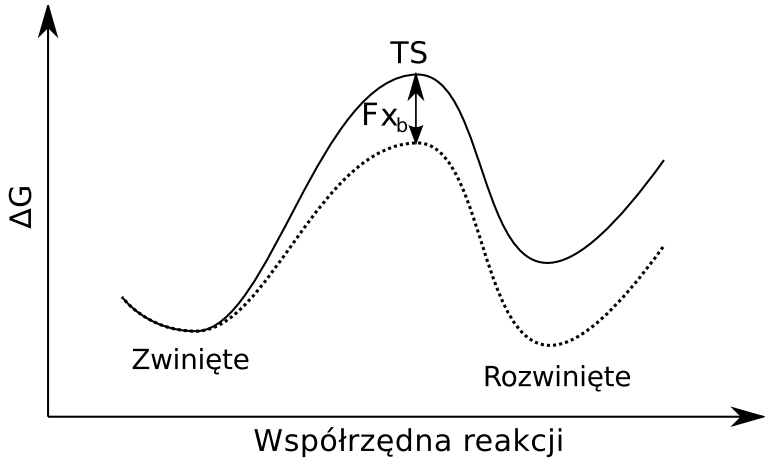
W wielu publikacjach do symulacji używany jest program NAMD [14]. Na stronie autorów [15] dostępne są tutoriale, które pozwalają na łatwiejsze wdrożenie się do pracy z tym programem. Jednak dla obliczeń prowadzonych przy zbliżonych warunkach NAMD okazuje się kilkukrotnie wolniejszy od programu Gromacs [16]. Gromacs pozwala na zastosowanie dwukrotnie większego kroku czasowego przy tym samym błędzie obliczeń. Ponadto NAMD wykazuje gorszą skalowalność w porównaniu do Gromacsa i co za tym idzie obserwuje się większy spadek wydajności obliczeń na jednostkę obliczeniową przy wzroście liczby procesorów. Stąd też wybór programu Gromacs jako lepszego narzędzia do prowadzenia symulacji.

2.4 Model teoretyczny Hummera-Szabo

Podczas rozciągania cząsteczki pod wpływem siły dochodzi do obniżenia bariery energetycznej stanu przejściowego, który cząsteczka musi przekroczyć by ulec rozwinięciu, które następuje w wyniku fluktuacji termicznych. Wprowadzenie siły do układu zwiększa prawdopodobieństwo przekroczenia bariery energetycznej. Do opisu fenomenologicznego tego zjawiska służy równanie Bell'a 2.3.

$$k(t) = k_0 \exp\left(\frac{F(t)x_b}{k_B T}\right) \tag{2.3}$$

Gdzie $k(t)$ oznacza stałą szybkości rozplatania przy sile $F(t)$, k_0 stanowi stałą



Rys. 2.5: Schemat ideowy zmiany powierzchni entalpii swobodnej pod wpływem rozciągania.

szybkości rozplatania bez obecności siły, x_b stanowi odległość bariery od minimum na powierzchni energii swobodnej, k_B stanowi stałą Boltzmanna, a T temperaturę bezwzględną.

Model fenomenologiczny zaprezentowany w pracy Hummera i Szabo [17] jest jednak niewystarczający do analizy wyników dla szerokiego zakresu szybkości rozciągania, ponieważ istniejące przybliżone rozwiązania analityczne mają zastosowanie tylko do wąskich, eksperymentalnych zakresów szybkości rozciągania i nie mogą być używane do analizy danych dla szybkości używanych w symulacjach.

W przypadku modelu mikroskopowego nie istnieją ścisłe rozwiązania analityczne, jednak przedstawione rozwiązanie numeryczne znajduje zastosowanie dla szerokiego zakresu prędkości.

W modelu mikroskopowym zakładamy, że siła jest zdefiniowana potencjałem harmonicznym:

$$V_s(x - vt) = \frac{1}{2} k_B T \kappa_s \left(x - vt\right)^2 \tag{2.4}$$

Gdzie V_s oznacza potencjał po współrzędnej reakcji x (odległość ramienia od położenia 0), v prędkość rozciągania, t czas, a κ_s oznacza stałą siłową wirtualnego cantilevera.

Kształt powierzchni entalpii swobodnej opisany jest zależnością:

$$V_0(x) = \begin{cases} \frac{1}{2}k_B T \kappa_m x^2 & \text{dla } x < x_b \\ -\infty & \text{dla } x \geqslant x_b \end{cases} \tag{2.5}$$

Gdzie κ_m oznacza stałą sprężystości cząsteczki, a x_b barierę, po której potencjał spada do $-\infty$.

System porusza się po następującej powierzchni potencjału:

$$V(x,t) = \begin{cases} \frac{1}{2}k_B T \kappa_m x^2 + \frac{1}{2}k_B T \kappa_s \left(x - vt\right)^2 & \text{dla } x < x_b \\ -\infty & \text{dla } x \geqslant x_b \end{cases} \tag{2.6}$$

Zakładamy również, że system porusza się ruchem Browna po powierzchni potencjału wyznaczonej przez 2.6. Wobec tego spełnione jest równanie dynamiki Browna:

$$\dot{x} = -D \bigtriangledown V(x,t) + R(t) \tag{2.7}$$

Gdzie $R(t)$ stanowi losową siłę z rozkładu Gaussa, gdzie jej wartość średnia wynosi 0, a wariancja $\overline{(t)R(t')} = 2D\delta(t - t')$.

Po podstawieniu 2.6 do 2.7 otrzymujemy równanie ruchu:

$$\dot{x} = -D\kappa_m x - D\kappa_s(x - vt) + R(t) \tag{2.8}$$

Przy założeniu, że $\overline{R(t)} = 0$ można obliczyć $\overline{x}(t)$, oznaczające średnie położenie w chwili t, rozwiązując równianie 2.8, które przy pominięciu składnika $R(t)$ przy uśrednianiu przekształca się w równanie różniczkowe pierwszego rzędu. Równianie to rozwiązano przy użyciu programu Wolfram Alpha [18]. Jako rozwiązanie ogólne otrzymano:

$$\overline{x}(t) = c_1 e^{-(\kappa_m + \kappa_s)Dt} + \frac{\kappa_s v((\kappa_m + \kappa_s)Dt - 1)}{D(\kappa_m + \kappa_s)^2} \tag{2.9}$$

Z warunku $\overline{x}(0) = 0$ otrzymujemy współczynnik c_1 oraz rozwiązanie szczególne:

$$\overline{x}(t) = \frac{\kappa_s v((\kappa_m + \kappa_s)Dt + e^{-(\kappa_m+\kappa_s)Dt} - 1)}{D(\kappa_m + \kappa_s)^2} \tag{2.10}$$

Aby znaleźć średnią siłę w danym czasie $\overline{F}(t)$ należy posłużyć się równaniem 2.4 na zależność potencjału od czasu i położenia. W wyniku obliczenia ujemnej pochodnej po x otrzymujemy siłę.

$$\overline{F}(t) = -k_B t \kappa_s (\overline{x}(t) - vt) \tag{2.11}$$

Aby znaleźć średnią siłę zerwania należy przyjąć, że zerwanie następuje przy przejściu przez barierę x_b, które to następuje po czasie τ. Czas ten można obliczyć z równania 2.10 oraz warunku, że średnie położenie w momencie zerwania τ wynosi x_b, czyli $\overline{x}(\tau) = x_b$.

Aby znaleźć prawdopodobieństwo przeżycia systemu w czasie t, $S(t)$ czyli prawdopodobieństwo tego, że cząstka nie uległa rozpleceniu po czasie t należy skorzystać z warunku:

$$\dot{S}(t) = -k(t)S(t) \tag{2.12}$$

Dającego:

$$S(t) = \exp\left(-\int_0^t k(t')dt'\right) \tag{2.13}$$

Przy założeniu, że prędkość rozciągania jest na tyle niska, że energia aktywacji w trakcie rozplatania wciąż jest wysoka i nie ulega drastycznej zmianie, można użyć przybliżenia adiabatycznego $\Delta t = x(t) - \overline{x}(t)$, które prowadzi do wyrażenia:

$$S(t) = \exp\left[-\int_0^t k_0(x_b - \overline{x}(t'))dt'\right] \tag{2.14}$$

Gdzie k_0 stanowi stałą szybkości uzyskaną z teorii Kramera:

$$k_0(x_b) = \frac{1}{\sqrt{2\pi}} D\kappa_m^{3/2} x_b e^{-\kappa_m x_b^2/2} \tag{2.15}$$

Po podstawieniu 2.15 do 2.14 otrzymuje się:

$$S(t) = \exp\left[-\frac{k_0 e^{-\kappa_s x_b^2/2}}{v\kappa_s x_b (\kappa_m/(\kappa_m+\kappa_s))^{3/2}} (\exp(\kappa_s V x_b t - \frac{1}{2}(\kappa_s v t)^2/(\kappa_m+\kappa_s)) - 1) \right] \tag{2.16}$$

Wyrażenie to podstawione do:

$$\overline{F} = -k_B T \kappa_s \left(x_b - v \int_o^\tau S(t) dt \right) \tag{2.17}$$

Pozwala na znalezienie zależności średniej siły rozplatania od prędkości rozciągania, $\overline{F}(v)$. Zależne jest ono od parametrów: k_0, x_b, oraz κ_m, które są dopasowywane do wartości sił rozplatania wziętych z symulacji dla różnych prędkości rozciągania.

Rozdział 3

Cel pracy

Cel pośredni pracy stanowiła symulacja rozplatania pojedynczego modułu kinazy tytyny przy różnych szybkościach rozciągania za pomocą programu Gromacs. Jako strukturę wyjściową wzięto strukturę krystalograficzną białka o symbolu 1TKI z bazy danych RCSB PDB [19]. Wyniki symulacji zostały porównane z wynikami literaturowymi.

Celem bezpośrednim było opracowanie uniwersalnej, niezależnej od badanej cząsteczki procedury numerycznej mającej za zadanie dopasowanie do danych z symulacji rozciągania krzywej teoretycznej z modelu mikroskopowego Hummera–Szabo [17]. Dopasowanie to pozwoliło na odzyskanie parametrów kinetycznych oraz porównanie wyników z danymi z eksperymentów AFM.

Podczas symulacji prowadzono także testy obliczeniowe klastra, dzięki czemu możliwa stała się analiza wydajności oraz ocena korzyści płynących z dalszej rozbudowy klastra.

Praca składała się z kilku etapów:

- Budowa klastra obliczeniowego
- Symulacje
  - Dobór parametrów symulacji

  - Równowagowanie
  - Właściwe obliczenia
- Analiza wyników
  - Odzyskanie sił rozplatania z wyników symulacji
  - Dopasowanie modelu do danych i odzyskanie parametrów kinetycznych
- Wnioski

Praca ta ma także stanowić krótkie omówienie procedury przeprowadzania obliczeń programem Gromacs, kilku najważniejszych parametrów i problemów, które może napotkać początkujący użytkownik, tak aby zdobyta wiedza stanowiła w przyszłości punkt wyjścia dla będących zainteresowanymi rozpoczęciem pracy z tym programem.

Rozdział 4

Metodologia pracy

4.1 Budowa i konfiguracja klastra

Klaster obliczeniowy stanowi zespół komputerów połączonych w ten sposób, by prowadzić równolegle obliczenia, dzięki czemu czas potrzebny na ich wykonanie ulega skróceniu w porównaniu do pojedynczej jednostki obliczeniowej. Dzięki temu uzyskujemy wzrost wydajności bez potrzeby oczekiwania na coraz to nowszy sprzęt.

Budowany klaster składał się z dwóch dwuprocesorowych jednostek centralnych Dell Precision T5500, z czego każdy procesor posiadał 4 rdzenie. Razem daje to możliwość uruchomienia 16 niezależnych procesów. Specyfikacja sprzętu przedstawiona została w tabeli 4.1.

Podzespół	Opis
Procesor	2 × Intel® Xeon® E5630; 4 rdzenie; 64 bit; taktowanie 2.53 GHz; 12 MB Cache
Płyt główna	Intel® 5520 chipset; dwuprocesorowa
Pamięć RAM	12 GB; DDR3
Dysk twardy	500 GB; SATA
karta sieciowa	Dwie karty sieciowe 1Gbps; jedna wbudowana na płycie głównej, druga w gnieździe PCI

Tablica 4.1: Specyfikacja komputerów.

Każdy z procesorów posiadał opcję HyperThreding, co pozwala na wirtualne

podwojenie liczny rdzeni, jednak wcześniejsze obliczenia programem Gromacs pokazały, że włączenie opcji HT nie zwiększa wydajności obliczeń, a może ją wręcz zmniejszać w wyniku zwiększenia komunikacji pomiędzy wątkami przy dotychczasowej przepustowości.

Aby fizycznie zbudować klaster wystarczyło połączyć komputery do jednej sieci Ethernet za pomocą switcha (ośmioportowy D-link DGS-1008D 1000Mbps). Na tak połączonych komputerach należało następnie zainstalować system operacyjny. Ze względu na fakt, że pakiet Gromacs był domyślnie pisany pod systemy Uniksowe oraz rozpowszechnienie na pracowni wybór padł na system Ubuntu w serwerowej wersji 10.04 LTS (64 bit). Ponadto system ten jest darmowy oraz dobrze udokumentowany. Następnym etapem była instalacja bibliotek niezbędnych do działania Gromacsa na każdym z komputerów.

4.1.1 Instalacja biblioteki OpenMPI

Przy obliczeniach równoległych podstawowym problemem jest komunikacja pomiędzy poszczególnymi procesami (węzłami) klastra. W celu ułatwienia tworzenia programów działających równolegle stworzono na początku lat dziewięćdziesiątych protokół komunikacji **MPI** (ang. Message Passing Interface), który ze względu na rozpowszechnienie stał się standardem i występuje w postaci bibliotek dla różnych języków programowania i systemów operacyjnych. Protokół ten jest odpowiedzialny między innymi za zarządzanie pamięcią, synchronizację danych, komunikację pomiędzy procesami oraz rozdział procesów na poszczególne węzły.

Również Gromacs korzysta ze standardu MPI. W związku z czym potrzebna była instalacja bibliotek obsługujących ten standard. Ze względu na rozpowszechnienie oraz na częste aktualizacje i szybkie poprawianie błędów wybrano bibliotekę OpenMPI w wersji 1.4.2, która dystrybuowana jest na licencji Open Source, dzięki czemu jest darmowa. Instalacja polega na ściągnięciu oraz rozpakowaniu archiwum z kodem źródłowym bibliotek, konfiguracji oraz kompilacji. Przy domyślnie zainsta-

lowanym systemie instalacja sprowadza się do wykonania 3 podstawowych poleceń.

1. `./configure` – uruchomienie skryptu konfiguracyjnego
2. `make` – kompilacja
3. `make install` – instalacja

Przy konfiguracji należy pamiętać, by jako domyślną ścieżkę instalacji bibliotek podać katalog systemowy `/usr/lib`, po czym instalację wykonywać przy uprawnieniach superużytkownika (`sudo make install`). Dzięki temu uniknie się problemów przy dostępie do tych bibliotek przez inne programy i innych użytkowników.

4.1.2 Instalacja biblioteki FFTW

Kolejnym krokiem jest instalacja bibliotek do wykonywania transformaty Fourierowskiej. Ze względu na szybkość największą popularnością cieszy się biblioteka FFTW. Pomimo dostępności w repozytorium systemu Ubuntu zaleca się kompilację ze źródeł, ponieważ dzięki temu uwzględnia się wszystkie możliwe optymalizacje dostępne pod dany procesor, a nie tylko te najbardziej uniwersalne.

Do instalacji użyto biblioteki FFTW w wersji 3.2.2. Instalacja przebiega podobnie jak w przypadku OpenMPI. Domyślnie biblioteka ta kompiluje się w podwójnej precyzji, kiedy większość obliczeń Gromacsem odbywa się w pojedynczej precyzji, przez co podwójna precyzja biblioteki FFTW nie jest potrzebna.

Kompilację i instalację prowadzi się przez wykonanie skryptu `./configure` z opcją `--enable-single`. Dzięki temu kompiluje się tę bibliotekę w pojedynczej precyzji. Ponadto można uwzględnić opcję obliczeń wielowątkowych dla systemów wielordzeniowych `--enable-threads` oraz instrukcje SSE dla rodziny procesorów i686 i nowszych `--enable-sse`.

Do poprawnej komunikacji pomiędzy programami wykorzystującymi bibliotekę OpenMPI na różnych komputerach wymagana jest możliwość zdalnego wywoływania poleceń na wszystkich komputerach wchodzących w skład klastra. Do tego celu służy

protokół SSH (ang. *Secure Shell*). Do obsługi wymagana jest instalacja serwera SSH na każdym z komputerów oraz możliwość bezhasłowego logowania się pomiędzy tymi węzłami.

4.1.3 Instalacja programu Gromacs

Przy symulacjach użyto programu Gromacs w wersji 4.5.1. Jego instalacja jest zbliżona do instalacji bibliotek, jednak przy konfiguracji należy podać odpowiednie parametry do uzyskania wersji obsługującej obliczenia równoległe.

Dogodnie jest wykonać dwie instalacje. Jedną dla wersji bez obsługi protokołu MPI, a drugą z obsługą. Pierwszą instalację konfiguruje się standardowym skryptem `./configure` z podaniem ścieżki instalacji oraz wyborem zainstalowanej wersji bibliotek do transformaty fourierowskiej, w tym przypadku FFTW `--enable-fft=fftw3`. Całość kompiluje się poleceniem `make` oraz instaluje poleceniem `sudo make install` jeśli potrzeba, by program był dostępny dla wszystkich użytkowników oraz podana ścieżka instalacji nie znajduje się w katalogu użytkownika.

Aby dodać obsługę MPI przy konfiguracji należy podać opcję `--enable-mpi`. Aby rozróżnić instalację Gromacsa z MPI należy przy konfiguracji uwzględnić opcję, która do nazwy programu z obsługą MPI doda sufiks _mpi `--program-suffix=_mpi`.

Poza głównym programem odpowiedzialnym za symulacje dynamiki pozostałe programy pakietu Gromacs nie korzystają z obsługi MPI, w związku z czym wystarczy skompilować i zainstalować tylko `mdrun` z obsługą MPI. Po konfiguracji służą do tego polecenia `make mdrun` oraz `sudo make install-mdrun`.

4.2 Przygotowanie struktury do symulacji

Jako przygotowanie cząsteczki do symulacji rozumie się wszelkie działania na plikach, podczas których nie wykonujemy obliczeń symulacji dynamiki molekularnej, które należy jednak przeprowadzić w celu uzyskania prawidłowej struktury wejściowej. Proces ten można rozbić na kilka najczęściej występujących etapów:

- Konwersja struktury na wewnętrzny format Gromacsa
- Umieszczenie cząsteczki w pudełku
- Dodatek wody do układu
- Dodatek jonów do układu

Aby przeprowadzić symulację dynamiki danej cząsteczki należy dysponować jej strukturą, którą uzyskuje się w wyniku badań krystalograficzncyh lub NMR. W internecie istnieje otwarta i darmowa baza danych z rozwiązanymi strukturami białek, RCSB Protein Data Bank [19], z której można pobrać plik w formacie PDB.

Jeśli badane białko zawiera aminokwasy kwasowe bądź zasadowe, to należy pamiętać, że różne stany sprotonowania prowadzą do różnej struktury, co może mieć wpływ na otrzymane wyniki. W celu optymalizacji struktury należy posłużyć się programem WHATIF, który jest dostępny w postaci usługi sieciowej z poziomu przeglądarki internetowej [20]. W pliku PDB zawierającym definicję białka 1TKI należało ręcznie usunąć linie zawierające wpisy położeń wody krystalizacyjnej oraz jednego z dwóch łańcuchów.

Posiadając już odpowiedni plik PDB należy przekonwertować go na wewnętrzny format używany przez Gromacsa. Do tego celu służy jeden z programów pakietu Gromacs, mianowicie `pdb2gmx`. W wyniku jego działania otrzymujemy plik `.gro` zawierający informacje o wielkości pudła, liczbie atomów w układzie, ich położeniach i prędkościach oraz plik z topologią `.top`, w którym znajdują się definicje typów atomów, ich masy, ładunki, typy wiązań oraz definicje oddziaływań i odwołania do pól siłowych.

Pole siłowe stanowi bazę zawierającą definicje dla ładunków cząstkowych atomów, długości wiązań, ich stałych siłowych, oraz innych oddziaływań dla różnego typu cząsteczek (ang. *residues*) takich jak aminokwasy, nukleotydy, cukry czy też parametry dla jonów. W zależności od dokładności definicji atomów pola siłowe dzielimy na pola typu *all atom*, które zawierają definicje dla ładunków oraz oddziaływań

dla wszystkich atomów występujących w cząsteczce oraz na pola typu *united atom*, które pomijają definicje dla atomów wodorów niepolarnych. Jeśli zamierza się symulować dynamikę cząsteczek nieujętych w definicji pola siłowego należy uzyskać na podstawie obliczeń kwantowomechanicznych ładunki cząstkowe oraz parametry dla oddziaływań w cząsteczce i dodać tę definicję. Komplikuje to używanie programów **MD** do niestandardowych układów.

Program używany do konwersji, `pdb2gmx` pozwala na wybór pola siłowego, modelu wody oraz stanu sprotonowania poszczególnych aminokwasów. Do symulacji użyto pola siłowego GROMOS, które jest polem typu *united atom* oraz modelu wody SPC. Stany sprotonowania histydyn zostały wybrane w oparciu o strukturę w pliku PDB otrzymanego w wyniku użycia programu WHATIF. Program ten dodaje również atomy wodoru nie wykryte podczas ustalania struktury. Wybrane pole siłowe nie zawiera definicji wszystkich atomów wodorów występujących w strukturze w pliku PDB, co prowadzi do błędów podczas konwersji na format plików Gromacsa. Aby temu zapobiec stosuje się program `pdb2gmx` z opcją `-ignh`, która powoduje pobranie informacji o położeniach atomów wodoru nie z pliku PDB, ale z definicji pola siłowego. Dodatkowa opcja `-his` pozwala na interaktywny wybór stanu sprotonowania histydyny, dzięki czemu informacja o wodorach biorących udział w tworzeniu wiązań wodorowych nie jest tracona.

Aby zdefiniować rozmiary pudełka wykorzystuje się kolejny z programów Gromacsa, `editconf`. Początkowe wymiary pudełka obieramy na podstawie publikacji [8] na 8.8 x 7.8 x 7.6 nm. Białko umieszcza się w pudełku poleceniem `editconf` z parametrami pliku wejściowego, wyjściowego oraz opcją `-box 8.8 7.8 7.6` określającą wymiary w nm oraz `-c` umieszczającą środek geometryczny białka w środku pudełka.

Kolejnym etapem jest dodatek cząsteczek wody do układu. Gromacs zawiera zdefiniowane kilka różniących się od siebie modeli wody:

- **SPC** — prosty, trójpunktowy model wody

- **SPC/E** — rozszerzony trójpunktowy model wody

- **TIP3P** — trójpunktowy model wody stosowany w polach siłowych CHARMM i AMBER

- **TIP4P** — czteropunktowy model wody, gdzie 3 punkty stanowią atomy tlenu i wodoru, natomiast czwarty punkt stanowi zdelokalizowany ładunek. Stosowany najczęściej dla pola siłowego OPLS/AA

- **TIP5P** — model pięciopunktowy, gdzie dwa dodatkowe punkty odpowiadają definicjom wolnych par elektronowych tlenu

Chcąc powtórzyć wyniki pracy [8] wybrano model SPC.

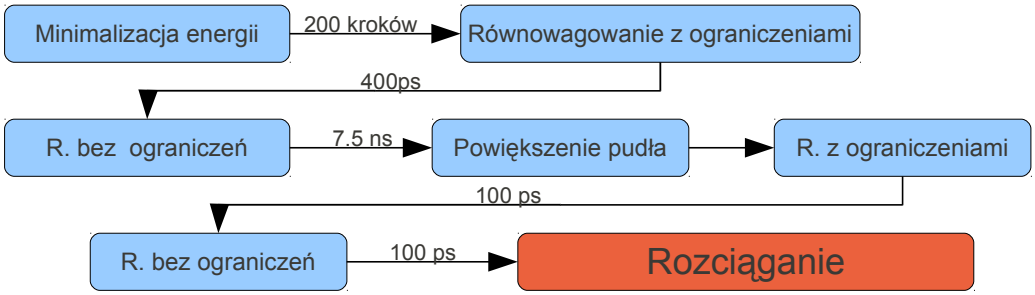
Do addycji wody do układu służy program `genbox`. W linii poleceń oprócz pliku wejściowego, wyjściowego oraz pliku z topologią należy podać plik ze strukturą wody, która zostanie powielona. W tym przypadku definicje wody znajdują się w pliku `spc216.gro` znajdującym się w katalogu głównym Gromacsa. W wyniku działania programu otrzymuje się układ składający się z 49606 atomów.

Następnie należało dodać do układu jony w celu zrównoważenia ładunku +2e białka oraz aby zbliżyć układ do warunków eksperymentalnych. Wykorzystuje się do tego program `genion`, będący częścią składową Gromacsa. Przed jego użyciem należy posiadać plik wsadowy do symulacji `.tpr`, który generuje się komendą `grompp`. Jako plik z parametrami posłużył plik do minimalizacji energii `em.mdp`. Posiadając już plik wsadowy należy w poleceniu `genion` podać go jako plik wejściowy, podać plik wyjściowy, plik z topologią oraz opcją `-nn` liczbę jonów ujemnych oraz `-np` liczbę jonów dodatnich. Jako domyślne program ten bierze kationy sodu oraz aniony chlorkowe, których liczba wynosi odpowiednio 29 i 27.

Istnieje również możliwość ustalenia stężenia jonów za pomocą opcji `-conc` w mol/litr oraz opcji `-neutral`, która odpowiada za otrzymanie układu elektrycznie obojętnego.

Mając tak zdefiniowany układ można przejść do właściwego etapu, jakim jest symulacja.

4.3 Równowagowanie



Rys. 4.1: Schemat przebiegu równowagowania i symulacji.

Przed właściwą symulacją należy przeprowadzić równowagowanie całego układu tak, aby wyeliminować w jak największym stopniu różnice między układem symulowanym, a rzeczywistym. Różnice te wynikają przede wszystkim z faktu, że dysponujemy strukturą krystaliczną, która nie odpowiada w pełni strukturze białka w roztworze, do której dąży się poprzez równowagowanie.

4.3.1 Minimalizacja energii

Pierwszym etapem jest minimalizacja energii całego układu. W wyniku dodawania wody, jonów, a także nieprawidłowości w samej strykturze pliku PDB mogą powstać miejsca, w których atomy znajdują się zbyt blisko siebie lub wręcz nakładają się wiązaniami. Podczas swobodnej symulacji prowadziłoby to do nadania tym atomom znacznie zawyżonych wartości energii kinetycznych, przez co badany układ by się rozpadł. Zjawisko to nazywa się eksplozją układu.

W celu ograniczenia tego efektu prowadzi się minimalizację energii potencjalnej

układu. W wyniku tej operacji otrzymuje się układ o jak najmniejszych oddziaływaniach pomiędzy atomami.

W tym celu utworzono plik z parametrami symulacji `em.mdp`. Najważniejsze na tym etapie parametry, które służą do kontroli symulacji to wybór integratora czyli algorytmu obliczeniowego oraz nadanie wartości specyficznym dla niego parametrom. W przypadku minimalizacji energii najczęściej wykorzystywanym i najszybszym, ale mało dokładnym algorytmem jest metoda najszybszego spadku. Do jej wyboru służy opcja `integrator = steep`. Dla tej metody charakterystyczne są parametry:

- `emtol` — Określa maksymalną siłę, poniżej której optymalizacja jest uznana za zakończoną.
- `nsteps` — Maksymalna liczba kroków optymalizacji.
- `emstep` — Krok.

Posiadając już plik `.mdp` wraz ze strukturą oraz topologią należy stworzyć binarny plik wejściowy z rozszerzeniem `.tpr` dla właściwych obliczeń. Służy do tego program `grompp`.

```
grompp -f em -o em -c ions.gro -p topol
```

Posiadając już plik `em.tpr` można uruchomić właściwą symulację. Służy do tego polecenie `mdrun`. Jednak ze względu na używaną wielowątkowość należy poprzedzić je poleceniem środowiska OpenMPI `mpirun`.

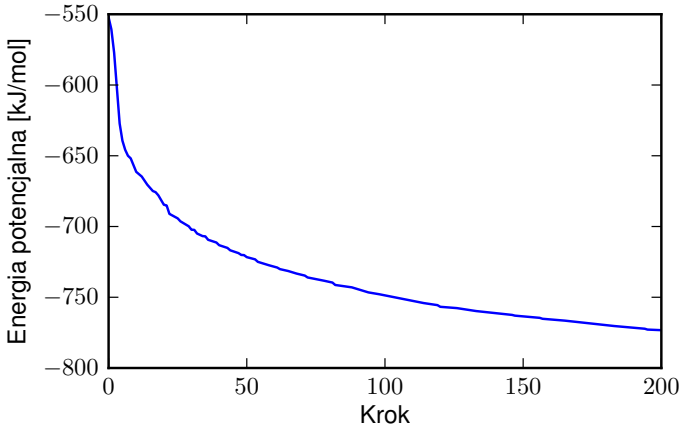
Do poprawnego działania wymaga ono podania liczby uruchamianych procesów `-np 16`, od 16 uruchamianych procesów oraz adresów sieciowych komputerów, na których prowadzone będą obliczenia. Można je podać bezpośrednio w linii komend, jednak wygodniej jest umieścić je w pliku, który wskazuje się opcją `-machinefile`. Na strukturę tego pliku składają się adresy sieciowe lub nazwy hostów komputerów. Opcjonalnie można po spacji podać maksymalną liczbę możliwych do uruchomienia wątków na danym komputerze, w tym przypadku było to 8 `max-slots=8` oraz liczbę wątków, które mają zostać wykorzystane, `slots=8`. Po zapisaniu tych linii do pliku `hosts`, komenda wygląda następująco:

```
mpirun -np 16 -hostfile ./hosts mdrun_mpi -v -deffnm em
```

W wyniku obliczeń uzyskuje się kilka plików z następującymi rozszerzeniami:

- `.log` — Zawiera podsumowanie obliczeń
- `.edr` — Plik z wartościami ciśnienia, temperatury, objętości, wyszczególnionych energii i innymi parametrami w zależności czasowej.
- `.trr` — plik z trajektorią. Przez trajektorię rozumie się położenia atomów oraz ich prędkości w zależności od czasu.
- `.gro` — Struktura po optymalizacji.

Korzystając z programu `g_energy` można pokazać jak zmieniała się energia potencjalna całego układu wraz z postępem optymalizacji. Na wykresie widać szybki spadek energii potencjalnej na początku minimalizacji oraz niewielkie jej zmiany pod koniec.



Rys. 4.2: Energia potencjalna układu podczas minimalizacji.

4.3.2 Równowagowanie rozpuszczalnika

Kolejnym krokiem w równowagowaniu jest już właściwa symulacja z ograniczeniami nałożonymi na atomy białka tak, by nie zmieniły one swojego położenia. Pozawala

to na swobodne rozmieszczenie rozpuszczalnika wraz z jonami wokół białka. Podczas tworzenia topologii przy konwersji z pliku PDB automatycznie tworzony jest plik z ograniczeniami (`.itp`) dla każdego atomu białka. Jego definicja jest automatycznie dodawania do pliku topologii, jednak domyślnie jest on nieaktywny. Można go aktywować w pliku `.mdp` dodając linię `define = -DPOSRES`. Nakłada on na każdy atom białka potencjał harmoniczny ze stałą siłową domyślnie 1000 kJ mol^{-1} nm^{-2} w każdym z wymiarów przestrzennych. Dzięki temu próba przesunięcia atomu w którymkolwiek kierunku prowadzi do niekorzystnego, wyższego stanu energetycznego układu. Metoda ta ma tę zaletę, że nie prowadzi do całkowitego unieruchomienia wybranych atomów.

W przypadku właściwej symulacji najczęściej stosowanym integratorem jest `md`. Tak też jest w tym przypadku. Do najważniejszych parametrów tej opcji należy krok czasowy `dt` [ps], który po zastosowaniu dalszych algorytmów optymalizacyjnych może być zmniejszony do 2 fs. Mając już ustalony krok czasowy oraz znając czas trwania symulacji można podać jako parametr `nsteps` liczbę kroków symulacji. W przypadku symulacji trwającej 400 ps liczbę kroków ustalono na 200000.

Grupa parametrów z prefiksem `nst` opisuje częstość zapisu wartości takich jak położenia atomów, ich prędkości, energie czy też zwykłe informacje kontrolne na temat symulacji do plików. Zawiera informację co ile kroków parametry te są zapisywane. Ma to wpływ na rozdzielczość późniejszych danych do analizy oraz ilość przestrzeni dyskowej zajmowanej przez zapisaną trajektorię.

Rozpoczynając symulację należy nadać prędkości początkowe atomom. Służy do tego parametr `gen_vel`, który przyjmuje wartości `yes` albo `no`. Prędkości dobierane są losowo z rozkładu Maxwella do temperatury podanej w parametrze `gen_temp` [K]. Do ustalenia ziarna służy parametr `gen_seed`, który ustawiony na `-1` przybiera wartość automatyczną na podstawie numeru procesu w systemie. Chcąc kontynuować symulację z nowymi parametrami należy pamiętać, by w nowym pliku `.mdp` ustawić `gen_vel` na `no`. Zapobiegnie to nadaniu nowych prędkości już zrównowagowanemu układowi.

4.3.3 Termostat

Kolejną kwestią jest kontrola temperatury. W wyniku zmniejszonej precyzji obliczeń oraz błędów numerycznych temperatura układu może ulegać znacznym fluktuacjom. Aby tego uniknąć stosuje się algorytmy kontroli stałości temperatury zwane termostatami. W tym przypadku układ równowagowany był do stałej temperatury 300 K.

Gromacs oferuje do wyboru trzy algorytmy kontroli temperatury (`berendsen`, `V-rescale` oraz `nose-hoover`) określane parametrem `tcoupl`, gdzie `no` oznacza brak termostatu.

Termostat Berendsena stanowi algorytm prostego skalowania prędkości wszystkich cząsteczek w danej grupie. Jako parametr opisujący szybkość doprowadzania układu do zadanej temperatury służy `tau_t`, natomiast temperaturę układu, do której się dąży opisuje `ref_t`. Aby uniknąć sytuacji, kiedy to większość energii kinetycznej przekazywana jest do rozpuszczalnika, prowadząc do problemu gorącego rozpuszczalnika i zimnej substancji rozpuszczonej zaleca się oddzielne sprzęganie rozpuszczalnika i substancji rozpuszczonej. Grupy te definiuje się parametrem `tc_grps` rozdzielając je spacją i podając temperaturę dla każdej z nich.

Termostat Berendsena jest najszybszy, jednak może prowadzić do powstawania błędów, gdyż nie uwzględnia rozkładu statystycznego skalowanych prędkości. Dlatego też w nowszej wersji programu Gromacs zaleca się stosowanie termostatu V-rescale, który jest zmodyfikowanym termostatem Berendsena, w ten sposób, że uwzględnia poprawny rozkład energii kinetycznej w układzie.

Algorytm Nosé-Hoover jest najwolniejszym, ale też najdokładniejszym z omawianych algorytmów. Pozwala na fluktuację temperatury wokół danej wartości. Jeśli poprzednie dwa termostaty pozawalają na szybkie doprowadzenie układu do pożądanej temperatury, tak ten termostat pozawala na najbardziej wierne jej zachowanie w trakcie symulacji. Jednak w publikacji, na którą się powołano próbując odtworzyć wyniki [8] używa się termostatu Berendsena przy temperaturze 300 K oraz stałej sprzęgania τ_T=0.1 ps, przez co w symulacjach użyto termostatu V-rescale.

4.3.4 Barostat

Mając ściśle zdefiniowane, stałe początkowe wymiary pudełka traci się kontrolę nad ciśnieniem oraz nad gęstością układu. Aby jak najbardziej zbliżyć symulację do rzeczywistego eksperymentu wprowadzono kontrolę ciśnienia poprzez stosowanie barostatów parametrem `pcoupl`. Najczęściej stosowanymi dwoma algorytmami są `berendsen` oraz `parrinello-rahman`.

Tak jak w przypadku termostatu algorytm Berendsena stanowy szybki i prosty algorytm skalowania, w tym przypadku wymiarów pudełka tak, by doprowadzić ciśnienie układu do pożądanej wartości. Może to prowadzić do błędów, ponieważ nie odwzorowuje prawidłowo układu stałej liczby atomów, ciśnienia i temperatury (NPT). W tym celu lepszym rozwiązaniem jest stosowanie algorytmu Parrinello-Rahmana, jednak należy pamiętać, że działanie tego algorytmu opiera się na oscylacji wokół stałego ciśnienia, co przy początku symulacji z dużej różnicy ciśnień może doprowadzić do nadania atomom niebezpiecznie wysokich prędkości. W omawianej symulacji używa się algorytmu Berendsena.

Do kontroli stałego ciśnienia służą następujące parametry:

- `pcoupltype` — Definiuje rodzaj skalowania pudełka. Najczęściej używanymi są `isotropic`, gdzie pudełko jest skalowane we wszystkich wymiarach tak samo, `semiisotropic`, gdzie skalowanie w wymiarze `x` oraz `y` jest takie samo, natomiast wymiar `z` skalowany jest niezależnie (zastosowanie przy symulacji błon lipidowych) oraz `anisotropic`, gdzie każdy z wymiarów jest skalowany niezależnie, a ponadto skalowane są dodatkowo 3 kierunki diagonalne.

- `tau_p` — Określa stałą czasową relaksacji τ_p w symulacji określoną jako 1.0 ps.

- `ref_p` — Określa ciśnienie do którego sprzęgany jest układ. Należy zdefiniować każdy wymiar. Aby w skalowaniu anizotropowym uniknąć skalowania ciśnienia w kierunkach diagonalnych ustala się ciśnienie na 0. Dzięki temu symulowane pudełko nie przekształca się w pudełko trygonalne. W pozostałych wymiarach

ciśnienie ustala się na 1.0 bar.

- `compressibility` — Ściśliwość, ustalana w każdym z wymiarów, domyślnie dla wody wynosi 4.5×10^{-5} bar^{-1}. Tak jak w przypadku ciśnienia należy również ustalić 0 dla diagonalnych.

Wszystkie parametry dla tego etapu symulacji umieszczono w pliku `npt_res.mpd`. Posiadając już ten plik można zastosować polecenia z wcześniejszego etapu. Całość tego symulacji zajęła 48 min 40 s na 16 rdzeniach.

4.3.5 Równowagowanie układu

Posiadając już strukturę z poprzedniej symulacji można przejść po usunięciu ograniczeń nałożonych na atomy białka do etapu równowagowania całego układu. Dzięki temu struktura krystaliczna białka zbliży się do tej, która występuje w roztworze.

Po usunięciu ograniczeń położenia atomów białka mogą one w wyniku drgań termicznych ulegać zmianie, w wyniku czego całe białko ulega rotacji i translacji. W wyniku określenia periodyczności układu nie stanowi to problemu podczas równowagowania, jednak przygotowując białko do rozciągania należy to uwzględnić w dalszej symulacji.

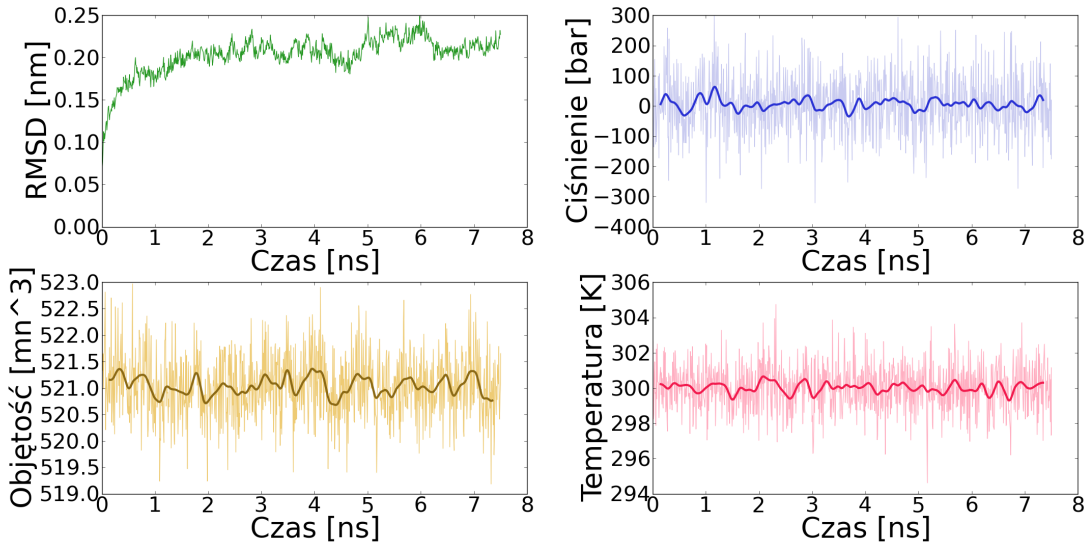
Aby mimo wszystko zachować położenie białka w centrum i pozwolić na ustalenie się struktury trzeciorzędowej zbliżonej do tej występującej w roztworze należy usunąć ruch środka ciężkości białka. W tym celu należy zdefiniować w pliku `mdp` parametr `comm_mode` jako `Linear`, który usuwa ruch translacyjny środka masy, `Angular`, który usuwa ruch translacyjny jak i rotację cząsteczki lub grupy cząsteczek wokół środka masy lub `No`.

Tryb `Linear` może być użyty w obliczeniach wielowątkowych, podczas których zachodzi podział pudełka na obszary, które liczą się oddzielnie (*domain decomposition*), natomiast tryb `Angular` może być użyty tylko w trybie *particle decomposition*, co oznacza podział nie na obszary, ale na cząsteczki liczone na oddzielnych wątkach. Dlatego też aby przyspieszyć obliczenia nie ustalano parametru `comm_mode` w pliku

`.mdp`, dzięki czemu Gromacs automatycznie ustawił go na usuwanie ruchu translacyjnego środka masy umożliwiając tym samym tryb pracy *domain decomposition*.

Równowagowanie prowadzone było przez 7.5 ns, gdzie czas obliczeń na 16 rdzeniach wyniósł 13h 38min 15s.

Do kontroli procesu równowagowania służy obserwacja temperatury i ciśnienia układu, a także RMSD (root-mean-square deviation) czyli pierwiastka ze średniej kwadratów odchyleń położeń atomów od położenia początkowego dla łańcucha głównego białka. Do uzyskania tych danych z obliczonej trajektorii służą polecenia `g_rms` oraz `g_energy`. Jako parametry wejściowe polecenia te biorą plik `.tpr` oraz odpowiednio trajektorię `.trr` lub `.xtc` oraz plik z danymi o układzie `.edr`. Na wyjściu otrzymuje się pliki w formacie `.xvg`, które można otwierać bezpośrednio programem `xmgrace` lub też jak w tym przypadku poddawać je analizie w innym środowisku, na przykład Python.



Rys. 4.3: Zmiana RMSD, ciśnienia, objętości i temperatury w czasie podczas równowagowania.

Analiza wykresów p, T i V wskazuje na ustalenie się ich wartości i fluktuację wokół zadanych parametrów, co jest zachowaniem pożądanym. Wykres RMSD także szybko rośnie do pewnego momentu, po czym nie przekracza wartości 0.25 nm,

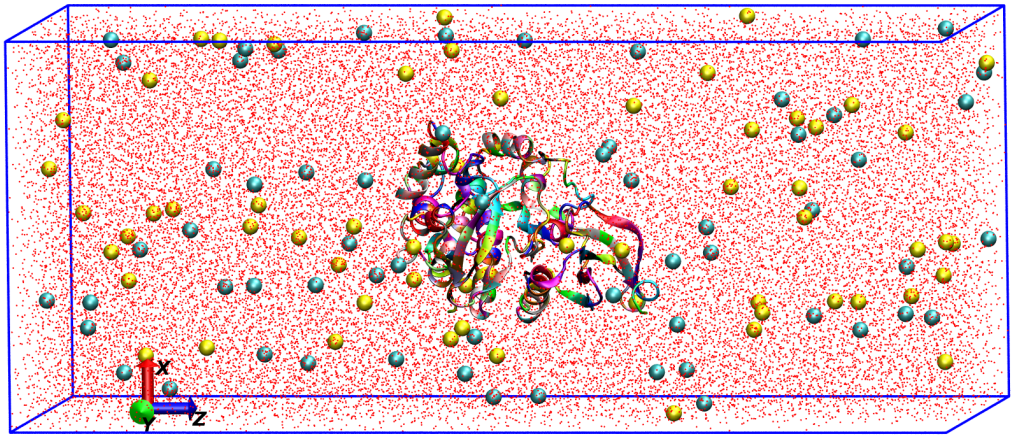
dzięki czemu można domniemywać, że białko osiągnęło strukturę równowagową. Dysponując ograniczonym czasem na wykonanie obliczeń trzeba polegać na założeniu wiarygodności tych danych, którymi się dysponuje, gdyż nigdy nie wiadomo jakim zmianom strukturalnym podlegałoby białko w dalszym czasie.

W wyniku równowagowania bez usuwania rotacji wokół środka ciężkości białka uległo ono obróceniu. Przy symulacji rozciągania jest to niekorzystne, ponieważ należałoby zastosować pudełko sześcienne, które przez zwiększoną objętość powodowałoby zwiększenie czasu obliczeń. Aby tego uniknąć stosuje się pudełko w formie prostopadłościanu o wymiarze zwiększonym tylko w kierunku rozciągania tak, aby pomieścić rozwijane białko.

W tym celu w dalszym etapie należy białko obrócić tak, by zorientować N i C-koniec białka wzdłuż jednej z osi układu współrzędnych i stworzyć nowe pudełko.

Służy do tego skrypt napisany w środowisku programistycznym Python (rot.py), który wczytuje plik `.gro` oraz oblicza kąty pomiędzy wektorem utworzonym pomiędzy dwoma zadanymi atomami, a osiami układu współrzędnych. W przypadku 1TKI atomami tymi były atomy węgla alfa Ile 138 oraz Lys 18. Mając już wartości kątów, korzysta się z polecenia `editconf` z parametrem `-rotate`, po którym podaje się kąty w stopniach, o które należy obrócić cząsteczkę wokół osi X Y Z tak, by wybrane wcześniej atomy znalazły się wzdłuż osi Z, po której białko ulega rozciągnięciu.

Po obrocie pudełka usunięto rozpuszczalnik oraz jony z układu zarówno w pliku `gro` jak i pliku z topologią. Następnie stworzono nowe pudełko o wymiarach 7.8 x 8.4 x 18.6 nm, dodano rozpuszczalnika oraz jonów według procedury opisanej poprzednio, co w efekcie dało 119587 atomów w układzie. Ponownie równowagowano układ najpierw z ograniczeniami nałożonymi na atomy białka, a następnie bez ograniczeń przez 100 ps. Czas obliczeń na 16 rdzeniach wyniósł odpowiednio 27min 10s oraz 28min 6s. Dzięki tej procedurze równowagowanie białka mogło być przeprowadzone w mniejszym pudełku, a co za tym idzie obliczenia trwały krócej niż miałoby to miejsce przy równowagowaniu przez 7.5 ns przy wymiarach pudełka niezbędnych do symulacji rozciągania.



Rys. 4.4: Wizualizacja symulowanego układu z białkiem, atomami tlenu od wody w kolorze czerwonym, jonami sodu w kolorze żółtym oraz jonami chloru w kolorze jasnoniebieskim.

Posiadając już zrównowagowany układ można było przystąpić do właściwych symulacji rozciągania.

4.4 Rozciąganie

Rozciąganie polega na przyłożeniu wirtualnej sprężyny do środka ciężkości grupy atomów lub tak jak w tym przypadku, do pojedynczych atomów i odsuwaniu jej ze stałą szybkością. Wraz z przemieszczeniem sprężyna o zadanej stałej sprężystości k powoduje powstawanie siły pomiędzy wirtualnym końcem, a danym atomem, co prowadzi do jego przemieszczenia.

Aby przystąpić do rozciągania należało zdefiniować grupy atomów, do których miała zostać przyłożona siła. Służy do tego polecenie `make_ndx`, w którym utworzono grupy `pull` oraz `stop` z poprzednio opisanych atomów węgla alfa dwóch końcowych aminokwasów. Program `make_ndx` posiada własną wewnętrzną składnię. Potrzebne grupy utworzono poleceniami `r 18 & a CA` oraz `r 338 & a CA`. Przed zapisaniem

należało nadać nowo utworzonym grupom nazwy. W wyniku działania tego programu otrzymuje się plik z rozszerzeniem `.ndx`, który należy uwzględnić przy uruchamianiu `grompp` opcją `-n`.

Przy rozciąganiu najdogodniej jest stosować zespół NVE, czyli stałą liczbę cząsteczek, objętość oraz stałą energię układu. Dzięki temu mamy zagwarantowaną stałą energię układu. Wymaga to wyłączenia barostatu oraz termostatu. W tym celu potrzebna jest podwójna precyzja obliczeń. Ponadto aby zapobiec zbytniemu wzrostowi temperatury układu w wyniku obecności zewnętrznej siły, a co za tym idzie pracy, należy zastosować większe pudełko. Powyższe zmiany prowadziłyby do znacznego wydłużenia czasu obliczeń i nie zostały uwzględnione w omawianych symulacjach.

Stosując anizotropowy barostat należy uniemożliwić jego działanie w wymiarze, w którym prowadzi się rozciąganie. Bez tego pudełko ulega odbiegającym od rzeczywistości odkształceniom, mianowicie kurczy się w wymiarze, w którym badana cząsteczka ulega rozciągnięciu.

Za rozciąganie odpowiada sekcja w pliku `.mdp` zwana *pull code*. Jej parametry przedstawiają się następująco:

- **pull** — Odpowiada za włączenie kodu rozciągania. Opcja `constant_force` sprawia, że rozciąganie odbywa się przy stałej sile. W tym przypadku zastosowano opcję `umbrella`, co pozwala na stosowanie stałej prędkości.

- **pull_geometry** — `distance` powoduje rozciąganie wzdłuż wektora łączącego dwie grupy. Opcja `direction` umożliwia rozciąganie wzdłuż zdefiniowanego wektora. Opcja ta nakłada ograniczenie powodujące, że odległości pomiędzy dwoma rozciąganymi grupami atomów nie mogą przekroczyć połowy wymiaru, w którym zachodzi rozciąganie. Aby uniknąć tego ograniczenia stosuje się opcję `direction_periodic`.

- **pull_dim** — Określa które z wymiarów wektora siły mają być zapisywane. W tym przypadku jest to wartość siły dla osi Z układu współrzędnych.

- **pull_group** — Zawiera nazwy grup atomów z pliku `ndx`, które mają zostać poddane rozciąganiu. `pull_group0` zawiera definicję grupy odniesienia. W przypadku braku tej definicji za punkt odniesienia bierze się początek układu współrzędnych.

- **pull_vec** — Definiuje wektor, wzdłuż którego białko ulega rozciąganiu odpowiednio dla każdej z grup poza grupą zerową.

- **pull_rate** — Określa dla każdej z grup prędkość rozciągania. Podawana jest w pm/ns.

- **pull_k** — Określa stałą siłową dla każdej z grup w kJ mol^{-1} nm^{-2}. W opisywanych symulacjach wynosi 500 kJ mol^{-1} nm^{-2}.

Symulacje rozciągania dla wszystkich prędkości prowadzone były na 16 rdzeniach, a każda z trajektorii była zapisywana w oddzielnym folderze. Podczas symulacji aktywny był barostat oraz termostat uprzednio używany podczas równowagowania. Rozciąganie odbywało się wzdłuż N-końca oraz C-końca białka. Jako wynik oprócz standardowych plików otrzymano pliki z wartościami siły, `pullf.xvg`.

Prędkość [nm/ns]	Czas symulacji [ns]	Czas obliczeń
50	0.2	1h15:55
10	1	4h48:31
5	2	11h57:22
2	5	21h50:17
0.8	10	2d7h56:25
0.4	20	3d15h07:03

Tablica 4.2: Czasy symulacji dla różnych prędkości rozciągania.

Listing 4.1: Polecenia Gromacsa używane podczas symulacji podane w kolejności wykonywania

```
1 pdb2gmx -f hadded.pdb -o tki.gro -ignh -his
```

```
editconf -f tki.gro -o box.gro -box 8.8 7.8 7.6 -c
genbox -cp box.gro -cs spc216.gro -o water.gro -p topol.top

grompp -f em -o ions -p topol -c water
genion -s ions -o ions.gro -p topol.top -nn 29 -np 27
grompp -f em -o em -c ions.gro -p topol
mdrun -v -deffnm em

grompp -f npt_res -o npt_res -c em.gro -p topol
mpirun -np 16 -hostfile ./hosts mdrun_mpi -v -deffnm npt_res
grompp -f npt -o npt -c npt_res.gro -p topol -e npt_res.edr -t
    npt_res.trr
mpirun -np 16 -hostfile ./hosts mdrun_mpi -v -deffnm npt

#### obrot czasteczki tak, by konce znalazly sie na osi Z
python rot.py
editconf -f npt_rot.gro -o npt_rot.gro -rotate 0 65.662 0
python rot.py
editconf -f npt_rot.gro -o npt_rot.gro -rotate -62.469 0 0
vmd npt_rot.gro
editconf -f npt_rot.gro -o npt_rot.gro -rotate 90 0 0
trjconv -f npt_rot.gro -o npt_large_box.gro -s npt

#### usuniecie z topol.top linii dot. wody i jonow
editconf -f npt_large_box.gro -o npt_large_box.gro -box 7.8 8.4 18.
    6 -c
genbox -cp npt_large_box.gro -cs spc216.gro -o npt_large_water.gro
    -p topol.top
grompp -f em -o npt_large_ions -p topol -c npt_large_water
genion -s npt_large_ions -o npt_large_ions.gro -p topol.top -
    neutral -c 0.086

grompp -f npt_large_res -o npt_large_res -c npt_large_ions.gro -p
    topol
mpirun -np 16 -hostfile ./hosts mdrun_mpi -v -deffnm npt_large_res
```

```
32 grompp -f npt_large -o npt_large -c npt_large_res.gro -p topol -e
      npt_large_res.edr -t npt_large_res.trr
33 mpirun -np 16 -hostfile ./hosts mdrun_mpi -v -deffnm npt_large
34
35 make_ndx -f npt_large -o pull.ndx
36
37 cd p50
38 grompp -f pull.mdp -o pull.tpr -p ../topol.top -c ../npt_large.gro
      -t ../npt_large -e ../npt_large -n ../pull.ndx -maxwarn
39 mpirun -np 16  -machinefile ../hosts mdrun_mpi -v -s pull -dlb yes
```

Rozdział 5

Analiza wyników

Posiadając już dane na temat zależności siły w czasie dla różnych prędkości rozciągania można było przystąpić do analizy wyników.

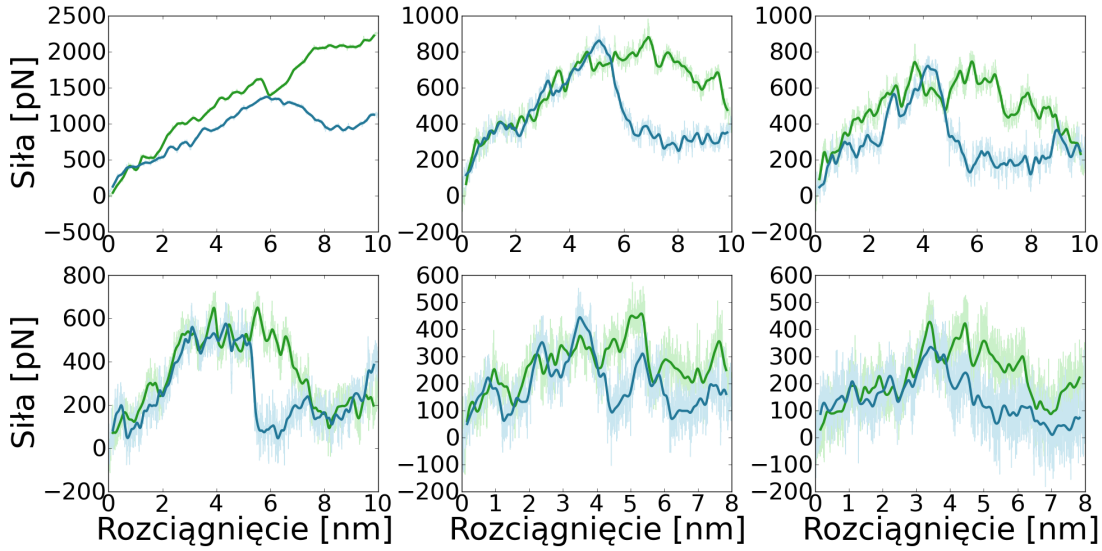
5.1 Znajdowanie siły rozplatania

Pierwszym etapem był wybór z zestawu danych maksymalnej siły rozplatania. Metoda najprostsza, czyli branie jako siły rozplatania maksymalnej siły z surowych wykresów siły od rozciągnięcia ma te zalety, że jest proste i najmniej czasochłonne, niestety wynik ten byłby zakłamany w wyniku drgań termicznych, które powodują powstawanie znacznych szumów na wykresie siły od rozciągnięcia. W celu eliminacji tego niepożądanego zjawiska zastosowano wygładzanie.

Aby uzyskać wygładzone krzywe napisano skrypt w języku Python (smooth.py) w wersji 2.6.5 na systemie Ubuntu 10.04 w wersji serwerowej. Skrypt ten odpowiedzialny był za wygładzanie wykresów metodą średniej kroczącej Gaussa z zadaną szerokością ramki. Jego główną część stanowi algorytm opisany na stronie [21].

Zasada jego działania polega na pobraniu liczby punktów odpowiadających szerokości ramki wokół danego punktu na wykresie niewygładzonym, a następnie wyliczeniu średniej ważonej, gdzie wagi odpowiadają rozkładowi Gaussa dla liczby punktów odpowiadających szerokości ramki.

Szerokość ramki użyta w publikacji [8] wynosiła 0.16 nm i została użyta również w tym przypadku. Przy wygładzaniu należało uwzględnić fakt, że używana stała szerokość w nm odpowiadała różnej liczbie punktów na wykresie. Znając szybkość rozciągania oraz krok zapisu danych obliczenie liczby potrzebnych punktów stało się trywialne.



Rys. 5.1: Wygładzone krzywe zależności siły od rozciągnięcia dla 6 prędkości rozciągania dla końców białka.

Posiadając już wygładzone wykresy siły od rozciągnięcia jako siły rozplatania bierze się maksymalne siły na wygładzonym wykresie siły od rozciągnięcia. Dla porównania z wynikami literaturowymi odczytano siły rozplatania z wykresu w publikacji [8].

Prędkość [nm/ns]	Maksymalna siła [pN]			
	Z symulacji		Z publikacji	
	N-koniec	C-koniec	N-koniec	C-koniec
50	1372	1618	1323	1376
10	815	822	796	710
5	720	745	645	526
2	576	650	366	559
0.8	443	459	495	667
0.4	334	429	452	538

Tablica 5.1: Maksymalne siły rozplatania dla różnych prędkości rozciągania dla symulacji oraz z danych literaturowych.

5.2 Dopasowanie krzywej teoretycznej do wyników

5.2.1 Mechanizm postępowania

Korzystając z modelu teoretycznego Hummera–Szabo omówionego we wstępie na podstawie publikacji [17] oraz metodologii prezentowanej w publikacji [22] napisano skrypt w środowisku programistycznym Python, który miał za zadanie poprzez optymalizację parametrów kinetycznych znaleźć jak najlepsze dopasowanie krzywej teoretycznej dla otrzymanych wyników.

Do obliczenia teoretycznej zależności średniej siły rozplatania od prędkości rozciągania służy wzór 2.17 przedstawiony we wprowadzeniu.

Do optymalizacji parametrów $\mathbf{p} = \{k_0, x_b, \kappa_m\}$ służy funkcja najmniejszych kwadratów $S(\mathbf{p})$:

$$S(\mathbf{p}) = \sum_{i=1}^{n} r_i^2$$

$$r_i = F_{s,i} - \overline{F}(v_i, \mathbf{p})$$

Gdzie $F_{s,i}$ oznacza i-tą siłę rozplatana wziętą z symulacji dla prędkości v_i, a $\overline{F}(v_i, \mathbf{p})$ oznacza teoretyczną wartość siły obliczoną dla i-tej prędkości i parametrów

p.

Mając tak zdefiniowaną funkcję najmniejszych kwadratów minimalizuje się ją wewnętrzną funkcją `fmin()` pakietu SciPy [23] języka Python ze względu na parametry **p**. Jest to optymalizacja metodą Simplex. Wbudowana w Pythona metoda najmniejszych kwadratów jest szybciej zbieżna niż optymalizacja metodą Simplex, jednak przy znacznej różnicy parametrów początkowych od optymalnych parametrów powoduje powstawanie błędów. Kod minimalizowanej funkcji błędu przedstawiony jest w 5.1 .

Listing 5.1: Minimalizowana funkcja błędu.

```
1 def err(p_l):
2         out=0
3         fs=F(p_l,V)
4         out=((fs - f)**2).sum()
5         return out
6 p1=fmin(err, p, xtol=1e-5, maxiter=1000000, maxfun=1000000)
```

Funkcja ta pobiera parametry do optymalizacji jako lokalne `p_l`. Na ich podstawie oraz na podstawie prędkości `V` liczona jest teoretyczna wartość siły `fs`. Suma kwadratów różnic siły teoretycznej `fs` oraz globalnej wartości siły `f` obliczana jest w linii nr 4. Do uzyskania wartości teoretycznych użyta jest funkcja `F()`, która zdefiniowana jest następująco:

Listing 5.2: Znajdowanie wartości funkcji teoretycznej dla zadanych parametrów i prędkości.

```
1
2 s = lambda t, vel_l, p_l: exp(-p_l[0] * exp(-0.5*ks*p_l[1]**2)/(
    vel_l*ks*p_l[1]*(p_l[2]/(p_l[2]+ks))**(1.5))*(exp(ks*vel_l*p_l[1
    ]*t - (0.5*(ks*vel_l*t)**2)/(ks+p_l[2]))-1))
3
4 def F(p_l, x_l):
5         k0_l, xb_l, km_l = p_l[0], p_l[1], p_l[2]
6         D_l=k0_l * sqrt(2*pi) / (km_l**(1.5)*xb_l*exp(-0.5*km_l*
```

```
            xb_l**2))
7           b=D_l*(km_l+ks)
8           out=[]
9           for i in range(len(x_l)):
10                  vel_l = x_l [i]
11                  a=(xb_l*D_l*(km_l+ks)**2/(ks*vel_l)+1)
12                  tau=(lambertW(-exp(-a))+a)/b
13                  out.append(-ks*300*kb*(xb_l-vel_l*quad(s,0,tau,
                        args=(vel_l, p_l), epsabs=1e-16)[0]))
14          return array(out)
```

Wartości $\overline{F}(v_i, \mathbf{p})$ stanowiące wynik równania 2.17 obliczane są w linii nr 13. Równania tego nie da się rozwiązać analitycznie, przez co występująca w nim całka oznaczona liczona jest numerycznie. Całka ta liczona jest dla funkcji `s` zdefiniowanej w linii nr 1, która stanowi równocześnie bezpośrednie odzwierciedlenie funkcji 2.16 z wprowadzenia. Granice całkowania zdefiniowane są od `0` do `tau`, które znajdywane jest na podstawie równania 2.10. Równanie to nie może być rozwiązane analitycznie, ponieważ otrzymujemy równanie typu:

$$a = b\tau + e^{-b\tau} \tag{5.1}$$

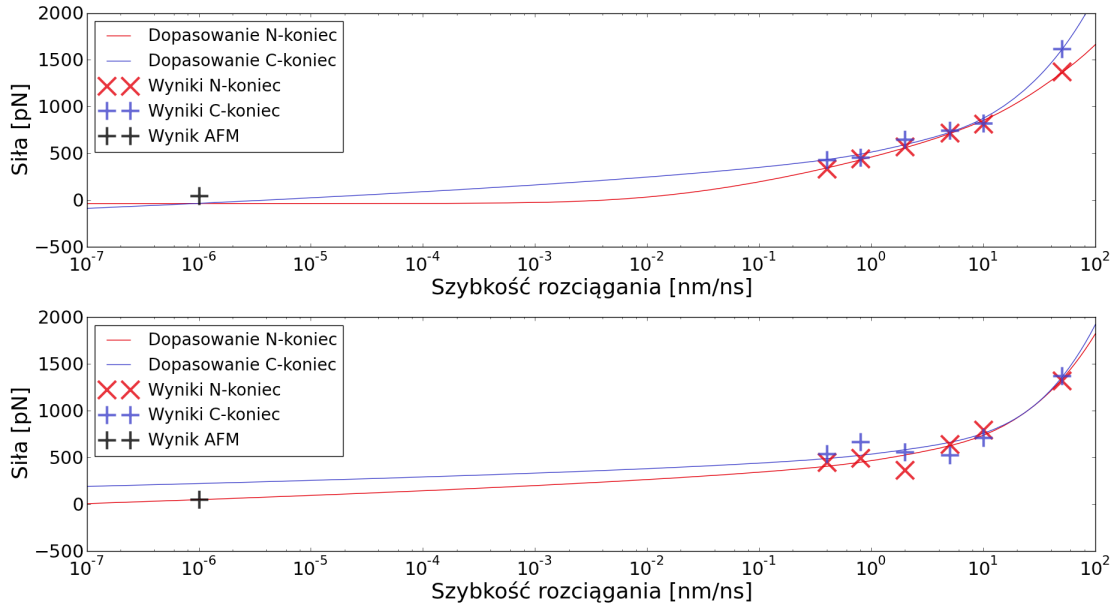
Gdzie `a` i `b` zdefiniowane są w kodzie. Do rozwiązania tego równania użyta została funkcja Ω Lamberta, która zdefiniowana jest jako funkcja odwrotna do We^W [24]. Rozwiązanie równania 5.1 znalezione zostało przy pomocy programu Wolfram Alpha [18] i przedstawione zostało w linii nr 12 kodu. Wartości funkcji Lamberta liczone są według procedury `lambertw()`, która została zaimportowana z pakietu SciPy w wersji 0.8 [23].

5.2.2 Analiza metody dopasowania

Mając maksymalne siły rozplatania z symulacji oraz dysponując danymi rozplatania w tych samych warunkach tej samej cząsteczki z publikacji [8] oraz danymi ekspery-

mentalnymi AFM z publikacji [7], gdzie znaleziona średnia siła rozplatania dla 1TKI wyniosła około 50 pN dla prędkości 1 μm/s można przystąpić do analizy wyników. Odbywa się ona poprzez dopasowanie parametrów k_0, x_b i κ_m do danych z symulacji oraz obliczenie na podstawie tych parametrów siły rozplatania dla prędkości 1 μm/s.

Zgodnie z teorią mikroskopową przedstawioną w publikacji [17] krzywa teoretyczna osiąga wartości ujemne poniżej prędkości równych iloczynowi k_0 i x_b, wobec czego jeśli $k_0 x_b > 1\mu$m/s odzyskanie siły rozplatania jest niemożliwe.



Rys. 5.2: Wykresy siły rozplatania w zależności od szybkości rozciągania dla danych z symulacji (górny panel) oraz dla danych literaturowych (dolny panel) wraz z dopasowanymi krzywymi teoretycznymi.

Z tabeli 5.2 wynika, że jedynie w przypadku parametrów odzyskanych z symulacji rozciągania dla N-końca dla publikacji [8] sensowna jest próba odzyskania siły rozplatania dla prędkości 1μm/s. Dla tych parametrów wynosi ona 48.9 pN, co jest dość dobrym przybliżeniem wobec 50 pN eksperymentalnych.

Koniec	k_0 [s^{-1}]	x_b [nm]	$k_bT\kappa_m$ [N/m]	x_bk_0 [m/s]
Symulacje				
N	7.944×10^{7}	0.04287	38.36	3.41×10^{-3}
C	2.787×10^{3}	0.1960	3.669	4.56×10^{-7}
Praca [8]				
N	3.970×10^{-1}	0.347	1.804	1.38×10^{-10}
C	1.219×10^{-26}	1.101	0.5820	1.34×10^{-35}

Tablica 5.2: Parametry kinetyczne k_0, x_b i κ_m dla N i C-końca 1TKI z symulacji i z publikacji [8]

Zbadano również zależność uzyskiwanych parametrów kinetycznych od wyboru punktu startowego. Analizę tę wykonano dla danych pochodzących z symulacji dla C-końca.

Parametr								
k_0 1.6× [s^{-1}]	10^{-17}	10^{-16}	10^{-12}	10^{-2}	10^{2}	10^{8}	10^{10}	10^{12}
x_b $\times10^{-10}$[m]	0.5	1	2	4	6	8	10	20
κ_m $\times10^{20}$	0.5	1	2	3	4	4×10^{22}	4×10^{23}	4×10^{24}

Tablica 5.3: Startowe parametry kinetyczne k_0, x_b i κ_m dla N-końca. Na czerwono zaznaczono te parametry, które generowały błąd.

W tabeli 5.3 umieszczono przykładowe parametry startowe. Na czerwono zaznaczono te, dla których optymalizacja nie powiodła się. Parametry startowe zaznaczone na zielono doprowadziły w wyniku optymalizacji do tych samych wartości, co przedstawione w tabeli 5.2.

Do dopasowania brane były wyniki pojedynczych symulacji, co przy dopasowaniu do wykresu **średnich** sił zerwania mogło prowadzić do dużych błędów w otrzymanych parametrach kinetycznych. Aby sprawdzić wpływ błędów wyników symulacji zbadano wpływ ±10% wartości siły rozplatania dla najwyższej i najwolniejszej szybkości rozplatania dla danych z symulacji dla C-końca na wartości odzyskanych parametrów kinetycznych.

	k_0 [s^{-1}]	x_b [nm]	$k_bT\kappa_m$ [N/m]	x_bk_0 [m/s]	$\sum r^2$
Symulacja	2.787×10^{3}	0.1960	3.669	4.56×10^{-7}	7.604×10^{-21}
v =50 m/s					
+10%	1.700×10^{-5}	0.4914	1.224	8.35×10^{-15}	1.165×10^{-20}
–10%	3.500×10^{4}	0.1499	5.492	5.25×10^{-6}	5.863×10^{-21}
v =0.4 m/s					
+10%	2.152	0.2929	2.340	6.30×10^{-10}	8.00×10^{-21}
–10%	8.289×10^{4}	0.1503	4.987	1.25×10^{-5}	8.91×10^{-21}

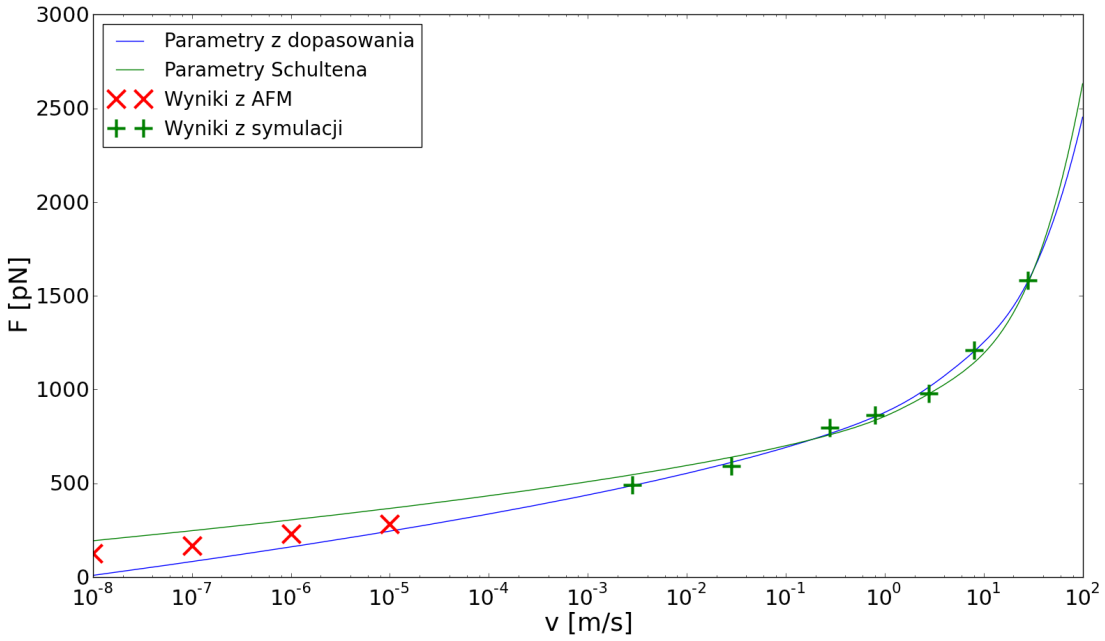
Tablica 5.4: Zmiany wyznaczonych parametrów kinetycznych w zależności od zmiany wartości maksymalnych sił rozplatania.

Z tabeli 5.4 wynika, że zmiana wyników o 10% prowadzi do zmiany k_0 o kilka rzędów wielkości.

Sprawdzono również tę metodę dla wyników symulacji rozciągania innego białka, modułu tytyny I91 [25], 1TIT w bazie PDB. Wyniki te były wzięte z pracy [22], gdzie parametry kinetyczne były wybierane nie na drodze numerycznej optymalizacji, ale za pomocą analizy dopasowania różnych zestawów parametrów. W przypadku tego białka dysponowano również zestawem parametrów kinetycznych k_0 i x_b z eksperymentów AFM z pracy [26], przy czym przy braku κ_m uzyskane ono zostało z dopasowania do danych eksperymentalnych i z symulacji przy stałych pozostałych parametrach.

	k_0 [s^{-1}]	x_b [nm]	$k_bT\kappa_m$ [N/m]	x_bk_0 [m/s]	$\sum r^2$
Dopasowanie	7.7×10^{-1}	0.172	7.52	1.32×10^{-10}	2.65×10^{-21}
Praca [22]	1.6×10^{-11}	0.385	2.86	6.16×10^{-21}	1.17×10^{-20}
AFM [26]	3.3×10^{-4}	0.250	4.54	8.25×10^{-14}	9.27×10^{-21}

Tablica 5.5: Dopasowanie parametrów kinetycznych do danych z pracy [22], przedstawienie parametrów z [22] oraz parametrów kinetycznych z eksperymentu AFM [26]



Rys. 5.3: Wykres krzywej teoretycznej z parametrami z pracy Schultena oraz po dopasowaniu z naniesionymi danymi z eksperymentu AFM.

	F [pN]		
v [nm/ms]	AFM	Praca [22]	Z dopasowania
0.01	125	193	8.6
0.1	166	247	82
1	230	304	160
10	283	366	244

Tablica 5.6: Przedstawienie sił rozplatania wyliczonych z dopasowania do danych z symulacji w porównaniu do rzeczywistych średnich sił rozplatania zmierzonych w eksperymencie AFM.

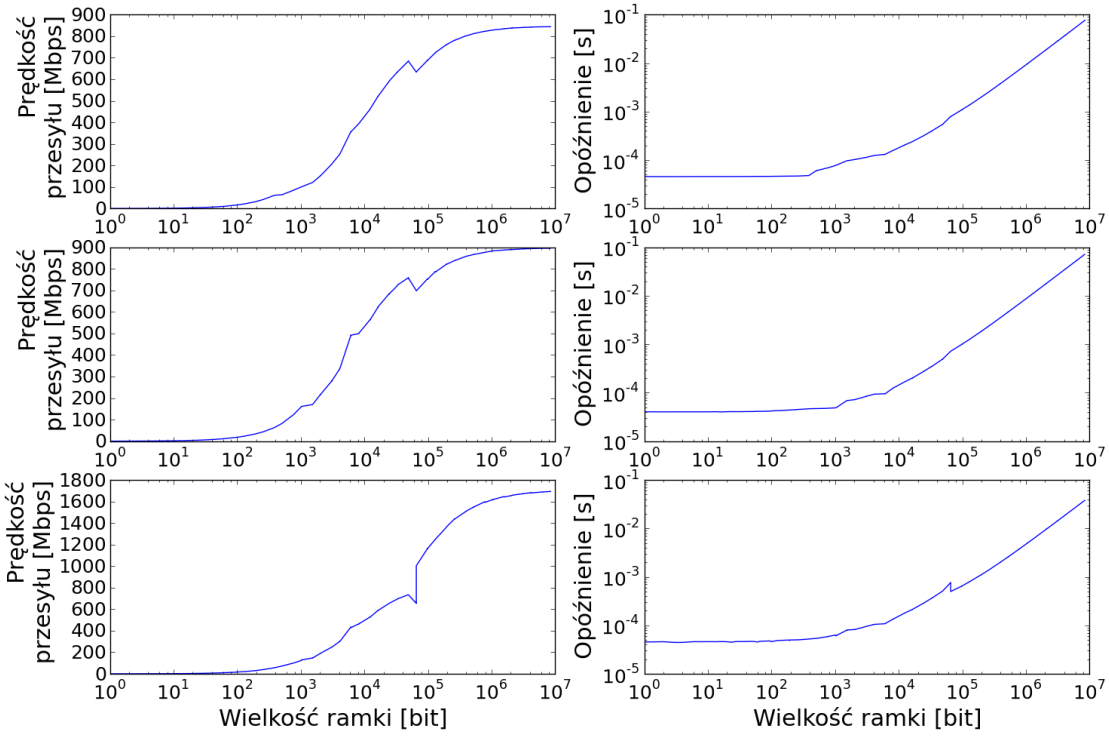
Przy czym zmiana średniej siły zerwania dla v=2.8 m/s z 997 pN na 978 pN czyli o 1.9% spowodowała zmianę dopasowania k_0 o rząd wielkości i zmianę wyznaczanej siły dla prędkości 0.01 nm/ms z 2×10^{-2} pN do 8.6 pN.

5.3 Analiza wydajności klastra

Pierwszym etapem w analizie wydajności było zbadanie jakości połączenia realizowanego pomiędzy dwoma komputerami. Komunikacja pomiędzy nimi odbywała się poprzez sieć 1 Gbps, jednak komputery te zostały wyposażone w 2 karty, także teoretyczna przepustowość powinna wynosić 2 Gbps.

Do analizy wydajności sieci służy narzędzie NetPIPE [27].

Analizę przepustowości przeprowadzono zarówno na pojedynczych kartach jak i na obu kartach działających razem.



Rys. 5.4: Wykresy przepustowości oraz opóźnienia dla poszczególnych kart sieciowych oraz dla kart sieciowych działających razem w zależności od wielkości przesyłanych danych.

Na podstawie wykresów można odczytać, że przepustowość komunikacji MPI przez pierwszą jak i przez drugą kartę sieciową wynosi odpowiednio 830 Mbps oraz

890 Mbps. Po włączeniu obu kart sieciowych naraz przepustowość komunikacji rośnie do 1680 Mbps. Opóźnienia dla małych rozmiarów pakietów wynoszą odpowiednio 42 μs oraz 36 μs. Dla kart działających razem opóźnienie to wynosi około 45 μs.

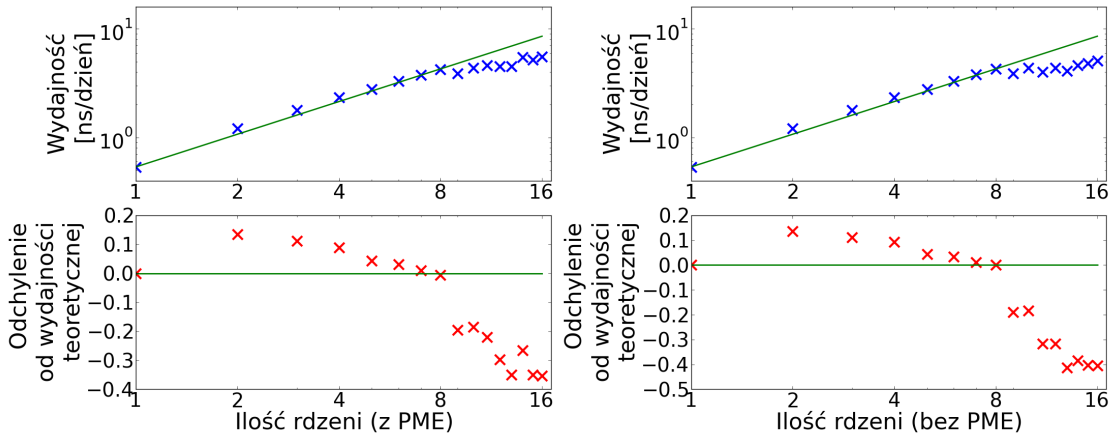
Wyniki te świadczą o wydajnym połączeniu pomiędzy dwoma komputerami, jednak opóźnienia rzędu kilkudziesięciu μs nijak mają się do tych osiąganych przez sieci typu Infiniband czy też Myrinet, które są o rząd wielkości mniejsze, co pozwala na zmniejszenie opóźnień w komunikacji pomiędzy procesami i w efekcie prowadzi do lepszej skalowalności.

Analiza wydajności klastra polegała na wykonaniu tej samej powtórzonej symulacji na różnej liczby rdzeni, a następnie zbadaniu czasu potrzebnego na obliczenia w zależności od liczby wykorzystywanych rdzeni. Dobrym wyznacznikiem jest liczba ns obliczonych w określonej jednostce czasu, na przykład na dzień. W idealnym przypadku liczba ta powinna rosnąć liniowo wraz ze wzrostem liczby wykorzystywanych rdzeni.

Do symulacji użyto tego samego układu, który był rozciągany przy prędkości 50 nm/ns. Symulację powtarzano 16 razy idąc od obliczeń na 1 rdzeniu aż do obliczeń na 16 rdzeniach prowadzonych równolegle. Obliczenia te wykonano zarówno z użyciem przydziału osobnych rdzeni na obliczenia PME jak i bez. Powyżej 8 rdzeni komunikacja między komputerami odbywała się poprzez dwie karty 1Gbps.

Na tej podstawie sporządzono wykresy wydajności liczonych ns/dzień oraz porównano jak obliczona wydajność ma się w stosunku do teoretycznego liniowego wzrostu wydajności.

Z powyższych wykresów można wnioskować, że skalowalność klastra zachowuje się liniowo aż do bariery 8 rdzeni. Ponadto wydajność obliczeń powyżej 1 rdzenia rośnie powyżej zależności teoretycznej, co świadczy o tym, że obliczenia tylko na jednym rdzeniu są wolniejsze. Po przekroczeniu granicy obliczeń prowadzonych na 8 rdzeniach wydajność w najgorszym przypadku jest o około 40% mniejsza niż wynika-



Rys. 5.5: Wykres wydajności obliczeń w zależności od liczby użytych rdzeni oraz ich odchyleń od wartości teoretycznych.

łoby to z linowego wzrostu wydajności. Podobne wyniki dają obliczenia prowadzone przy automatycznym wyborze liczby rdzeni przeznaczonych tylko na obliczenia PME jak i obliczenia prowadzone bez przydziału osobnych rdzeni na obliczenia PME poprzez zastosowanie opcji `-nopme` w programie `mdrun`.

Wydajność obliczeń w przypadku zastosowania automatycznego przydziału rdzeni na obliczenia PME jest jednak wyraźnie większa przy wykorzystaniu 16 rdzeni niż bez tego przydziału.

Rozdział 6

Wnioski

Na podstawie przeprowadzonych obliczeń można wywnioskować, że symulacje rozplatania białek o znanej strukturze mogą być pomocne przy odzyskiwaniu parametrów kinetycznych. Należy jednak zawsze pamiętać, że metoda ta może być tylko wspomaganiem eksperymentu. Ostatecznym potwierdzeniem zawsze musi być emipryczny, dający się powtórzyć wynik doświadczalny, a nie tylko teoretyczny.

Dokładność uzyskanych parametrów kinetycznych w ogromnej mierze zależy od ilości oraz jakości uzyskanych wyników symulacji. Stosując dużą próbę statystyczną uzyskujemy lepsze wyniki. Pojedyncze wartości sił rozplatania czy też zerwania stanowią zbyt słabą przesłankę co do prawdziwości odzyskanych parametrów kinetycznych. Ponadto w celu poprawnego dopasowania ogromny wpływ ma także duża rozpiętość badanych szybkości. Im więcej, tym lepiej. Badana zależność sił rozplatania od prędkości rozciągania ma jednak charakter logarytmiczny. Pociąga to za sobą wykładniczy wzrost czasu potrzebnego do symulacji, a co za tym idzie czasu obliczeń. Z tego powodu zachodzi potrzeba zakupu coraz to wydajniejszego sprzętu.

Pojedyncze wyniki, tak jak w tym przypadku powodują powstawanie dużego błędu z powodu małej liczebności grupy. Oznaczane parametry, jak na przykład stała rozplatania k_0 charakteryzują się dużą czułością i nawet błąd kilkudziesięciu pN w oznaczaniu siły rozplatania może oznaczać różnice wyznaczonej stałej szybkości sięgającą kilku rzędów wielkości. Nie tyle rozwiązaniem, co ograniczeniem tego

problemu jest wspomniane wcześniej zwiększenie próby, co pociąga za sobą jednak zwiększenie czasu obliczeń. Za pocieszenie może jednak służyć fakt, że rozrzut sił rozplatania maleje wraz ze spadkiem prędkości. Z drugiej strony jednak dokładność uzyskanych parametrów kinetycznych zależy przede wszystkim od wartości dla mniejszych prędkości.

Przy odzyskanych parametrach kinetycznych obliczone na ich podstawie siły dla eksperymentalnego zakresu prędkości znacząco odbiegają od tych wyznaczonych doświadczalnie. Wynika to z czułości algorytmu dopasowania na wyniki symulacji. Ponadto ekstrapolacja sił rozplatania dla zakresu eksperymentalnego jest bardzo czuła na zmiany parametrów kinetycznych, które dla zmiany sił rozplatania o 10% mogą przyjmować wartości różniące się od siebie nawet o kilka rzędów wielkości. Dlatego też metoda ta wymaga dalszego doskonalenia. Przy większej liczbie wyników symulacji dla danej szybkości pomocne może być uwzględnienie odchyleń standardowych przy metodzie najmniejszych kwadratów poprzez stosowanie ważenia. Wymaga to jednak większej ilości wartości sił rozplatania dla danej szybkości.

Budowa klastra na zwykłej sieci o przepustowości 1 Gbps opartej na zwykłym switchu przy maszynach wieloprocesorowych, kiedy to na jednej maszynie może być uruchomionych aż 8 niezależnych procesów jest nieefektywna. Ze względu na niewielki wzrost wydajności przy użyciu wszystkich 16 wątków rozsądniejsze wydaje się prowadzenie obliczeń równolegle na dwóch maszynach. Wynika to z prostego faktu, że komunikacja pomiędzy procesami uruchomionymi na jednej płycie głównej jest nieporównywalnie szybsza niż pomiędzy tymi samymi procesami połączonymi przez sieć. W zastosowaniu znajdują się specjalne szybsze łącza typu Infiniband czy Myrinet, które pozwalają na uzyskanie lepszej skalowalności.

Nie można także zapominać o wiedzy potrzebnej do prawidłowej obsługi programów do symulacji dynamiki molekularnej. W przypadku programu Gromacs opanowanie parametrów służących do kontroli symulacji zajęło około rok bazując przede wszystkim na znalezionych w internecie samouczkach oraz listach mailingowych służących do wymiany myśli pomiędzy użytkownikami. Dlatego też dobór takich opcji

jak czas równowagowania, sprotonowanie, zespół, typ barostatu i termostatu ma kluczowy wpływ na wynik. Nieznajomość parametrów, a co za tym idzie nieumiejętność ich kontroli może prowadzić do sytuacji, kiedy to wynik symulacji nie ma nic wspólnego z rzeczywistością. Nawet będąc pewnym swoich umiejętności zawsze należy podchodzić do wyniku symulacji z dużą rezerwą, ponieważ jest to tylko wynik teoretyczny.

Symulacje dynamiki molekularnej stanowią ogromną wartość jako narzędzie do badania i wizualizacji procesów odbywających się w skali pojedynczej cząsteczki. Dzięki temu mamy możliwość wglądu na zachowanie oraz rolę poszczególnych atomów w trakcie danego procesu.

Bibliografia

[1] D. van der Spoel, E. Lindahl, B. Hess, A. R. van Buuren, E. Apol, P. J. Meulenhoff, D. P. Tieleman, A. L. T. M. Sijbers, K. A. Feenstra, R. van Drunen, H. J. C. Berendsen. Gromacs user manual version 4.5. *www.gromacs.org*, V 2011.

[2] M. Waligórski. Spektroskopia korelacji fluorescencji: symulacje monte carlo fluorescencji dyfundujących fluoroforów. praca magisterska. *Uniwersytet im. A.Mickiewicza, Wydział Chemii*, 2007.

[3] H. Lu, B. Isralewitz, A. Krammer, V. Vogel, K. Schulten. Unfolding of titin immunoglobulin domains by steered molecular dynamics simulation. *Biophys Journal*, 75:662–671, 1998.

[4] K. Maruyama. Connectin, giant elastic protein of muscle. *The FASEB Journal*, 11:341–345, 1997.

[5] S. Labeit, B. Kolmerer. Titins: Giant proteins in charge of muscle ultrastructure and elasticity. *Science*, 270:293–296, 1995.

[6] O. Mayans, P. van der Ven, M. Wilm, A. Mues, P. Young, D. O. Furst, M. Wilmanns, M. Gautel. Structural basis for activation of the titin kinase domain during myofibrillogenesis. *Nature*, 395:863 – 869, 1998.

[7] E. M. Puchner, A. Alexandrovich, A. L. Kho, U. Hensen, L. V. Schäfer, B. Brandmeier, F. Gräter, H. Grubmüller, H. E. Gaub, M. Gautel. Mecha-

noenzymatics of titin kinase. *Proceedings of the National Academy of Sciences*, 105:13385–13390, 2008.

[8] F. Gräter, J. Shen, H. Jiang, M. Gautel, H. Grubmüller. Mechanically induced titin kinase activation studied by force-probe molecular dynamics simulations. *Biophysical Journal*, 88:790 – 804, 2005.

[9] Stephan Lange, Fengqing Xiang, Andrey Yakovenko, Anna Vihola, Peter Hackman, Elena Rostkova, Jakob Kristensen, Birgit Brandmeier, Gereon Franzen, Birgitta Hedberg, Lars Gunnar Gunnarsson, Simon M. Hughes, Sylvie Marchand, Thomas Sejersen, Isabelle Richard, Lars Edström, Elisabeth Ehler, Bjarne Udd, Mathias Gautel. The kinase domain of titin controls muscle gene expression and protein turnover. *Science*, 308:1599–1603, 2005.

[10] A. Wiita, R. Perez-Jimenez, K. A. Walther, F. Grater, B. J. Berne, A. Holmgren, J. M. Sanchez-Ruiz, J. M. Fernandez. Probing the chemistry of thioredoxin catalysis with force. *Nature*, 450:124 – 127, 2007.

[11] M. Carrion-Vazquez, A. F. Oberhauser, S. B. Fowler, P. E. Marszalek, S. E. Broedel, J. Clarke, J. M. Fernandez. Mechanical and chemical unfolding of a single protein: A comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 96:3694–3699, 1999.

[12] T. E. Fisher, A. F. Oberhauser, M. Carrion-Vazquez, P. E. Marszalek, J. M. Fernandez. The study of protein mechanics with the atomic force microscope. *Trends in Biochemical Sciences*, 24:379 – 384, 1999.

[13] J. Lemkul. Gromacs tutorials. `http://www.bevanlab.biochem.vt.edu/Pages/Personal/justin/gmx-tutorials/index.html`, V 2011.

[14] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, K. Schulten. Scalable molecular dynamics with namd. *Journal of Computational Chemistry*, 26:1781–1802, 2005.

[15] Namd tutorials. http://www.ks.uiuc.edu/Research/namd/, V 2011.

[16] B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl. Gromacs 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation. *Journal of Chemical Theory and Computation*, 4:435–447, 2008.

[17] G. Hummer, A. Szabo. Kinetics from nonequilibrium single-molecule pulling experiments. *Biophysical Journal*, 85:5–15, 2003.

[18] Wolfram alpha. www.wolframalpha.com, V 2011.

[19] Rcsb protein data bank. http://www.rcsb.org/, V 2011.

[20] Whatif internet server. http://swift.cmbi.ru.nl, V 2011.

[21] S. Harden. Linear data smoothing in python. http://www.swharden.com/blog/2008-11-17-linear-data-smoothing-in-python/, V 2011.

[22] E. H. Lee, J. Hsin, M. Sotomayor, G. Comellas, K. Schulten. Discovery through the computational microscope. *Structure*, 17:1295 – 1306, 2009.

[23] Scipy. http://www.scipy.org/, V 2011.

[24] Definicja, właściwości funkcji lamberta w. http://en.wikipedia.org/wiki/Lambert_W_function, V 2011.

[25] S. Improta, A. S. Politou, A. Pastore. Immunoglobulin-like modules from titin i-band: extensible components of muscle elasticity. *Structure*, 4:323 – 337, 1996.

[26] M. Carrion-Vazquez, A. F. Oberhauser, S. B. Fowler, P. E. Marszalek, S. E. Broedel, J. Clarke, J. M. Fernandez. Mechanical and chemical unfolding of a single protein: A comparison. *Proceedings of the National Academy of Sciences*, 96:3694–3699, 1999.

[27] Netpipe. http://www.scl.ameslab.gov/netpipe/, V 2011.

Dodatek A. Spis zawartości dołączonej płyty CD

```
Katalog główny
├── praca.pdf - Praca magisterska
├── smooth.py - Skrypt Pythona odpowiedzialny za wygładzanie
├── fit_polaczone.py - Skrypt Pythona odpowiedzialny za dopasowanie modelu
│   Hummera-Szabo do danych
├── README - Plik z opisem zawartości płyty
└── pliki_Gromacs - Katalog z plikami potrzebnymi do przeprowadzenia symulacji
    ├── polecenia - Spis poleceń Gromacsa używanych przy symulacji
    ├── 1TKI.pdb - Oryginalny plik PDB z bazy danych
    ├── hadded.pdb - Plik PDB po operacjach programem WHATIF
    ├── rot.py - Skrypt Pythona do obliczania kątów przy obracaniu białka
    ├── hosts - Plik z adresami hostów dla OpenMPI
    ├── topol.top - Plik topologii
    ├── *.mdp - Pliki z parametrami dla równowagowania
    ├── p50 - parametry i podsumowanie dla v=50 nm/ns
    │   ├── pull.mdp - Parametry dla rozciągania
    │   └── md.log - Informacje na temat symulacji
    ├── p10 - parametry i podsumowanie dla v=10 nm/ns
    ├── p5 - parametry i podsumowanie dla v=5 nm/ns
    ├── p2 - parametry i podsumowanie dla v=2 nm/ns
    ├── p0.8 - parametry i podsumowanie dla v=0.8 nm/ns
    └── p0.4 - parametry i podsumowanie dla v=0.4 nm/ns
```

Poznań, dnia 21 czerwca 2011

OŚWIADCZENIE

Ja, niżej podpisany **Mateusz Najsztub**, student Wydziału Chemii Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt.:

. . . . . .

. . . . . .

napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób.

Oświadczam również, że egzemplarz pracy dyplomowej w formie wydruku komputerowego jest zgodny z egzemplarzem pracy dyplomowej w formie elektronicznej.

Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, decyzja o wydaniu dyplomu zostanie cofnięta.

Prof. dr hab. Andrzej Molski	. . . . .
/nazwisko kierownika pracy magisterskiej/	/podpis studenta/
. . . . .	87102302630
/podpis kierownika pracy magisterskiej/	/PESEL studenta/