RAFAEL RISNIK ROMEIRO

Proposta de um campo de forças *coarse-grained* para a previsão da estrutura nativa de baixa resolução de proteínas

São Paulo

2017

RAFAEL RISNIK ROMEIRO

Proposta de um campo de forças *coarse-grained* para a previsão da estrutura nativa de baixa resolução de proteínas

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências.

Orientador: Prof. Dr. Pedro de Alcântara Pessoa Filho

São Paulo

2017

RAFAEL RISNIK ROMEIRO

Proposta de um campo de forças *coarse-grained* para a previsão da estrutura nativa de baixa resolução de proteínas

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências.

Área de Concentração: Engenharia Química

Orientador: Prof. Dr. Pedro de Alcântara Pessôa Filho

São Paulo

2017

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo,_____de_____de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Romeiro, Rafael Risnik
Proposta de um campo de forças coarse-grained para a previsão da estrutura nativa de baixa resolução de proteínas / R. R. Romeiro -- versão corr. - São Paulo, 2017.
151 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Química.

1.Proteína 2.Dinâmica Molecular 3.Termodinâmica I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Química II.t.

À milha família por todo amor e suporte

AGRADECIMENTOS

A realização deste trabalho não seria possível sem a presença de algumas pessoas e instituições. Deixo aqui, então, meu agradecimento:

- Ao meu orientador Prof. Dr. Pedro de Alcântara Pessôa Filho pela orientação, apoio intelectual e emocional, companheirismo e dedicação desde os anos da graduação;
- Aos meus pais pelo suporte, amor, carinho e dedicação sempre apoiando e incentivando o caminho que escolhi;
- Ao meu irmão pela companhia no árduo caminho da academia, pelas risadas e pela sempre motivadora presença;
- A minha querida avó, Cecília, por todas as conversas, apoios, risadas e pela companhia durante toda minha jornada.
- A minha tia, Ana Paula, por todo apoio e carinho que sempre demonstrou.
- A Simone Satie Kudo, que apesar de não ser da família consanguínea, faz parte da família do coração, pela amizade, pela companhia, pelo apoio emocional, pelas risadas, pelo companheirismo e pela presença durante esta jornada;
- Ao Dr. Luis Fernando Mercier Franco pela amizade, companheirismo, orientação, apoio intelectual e emocional mesmo estando do outro lado do mundo;
- Aos amigos Sara Isabel Borges Cardoso, Luan Merida de Medeiros, David Éber Sanches de Menezes, Guilherme Fabri Pereira e Matheus Mlot Palma pela companhia, pelas risadas, pela atenção e pelo companheirismo de todos os dias;
- A todo corpo de docentes e funcionários da Escola Politécnica da USP pelo apoio e dedicação e;
- Ao CNPq, pela contribuição financeira à pesquisa.

RESUMO

A capacidade de prever a estrutura nativa de uma proteína é um problema ainda sem solução. A predição da estrutura final ou nativa de uma proteína -, ou seja, partir da estrutura primária (sequência de aminoácidos linear) de um polipeptídeo tentar prever qual será a estrutura terciária (arranjo de hélices alfa, folhas beta e grampos) - tem sido um desafio para diversos pesquisadores desde o século passado. Atualmente existem diversos modelos que se propõem a executar essa tarefa, mas poucos que de fato partem de princípios físicos básicos para realizá-la. A grande maioria baseia-se em estruturas já conhecidas de proteínas com sequenciamento ou função similares para prever a estrutura terciária. Neste trabalho, no entanto, propõe-se o desenvolvimento de um campo de forças *coarse-grained* para aplicação em simulações de dinâmica molecular a fim de prever a estrutura de proteínas sem que seja necessária a comparação com estruturas já conhecidas. O fator de forma é um importante indicativo da estrutura de uma molécula em solução. Apesar de se tratar de uma grandeza que fornece informações de baixa resolução, ou seja, não fornece pormenores a respeito da posição dos átomos na estrutura da molécula, é uma estimativa inicial do espaço ocupado pela molécula e também da maneira com a qual ela interage com outras moléculas em solução. Isso é decorrente das operações matemáticas necessárias para que o fator de forma seja acessado a partir de dados de experimentos de espalhamento de raios X. Os resultados mostram que o método consegue prever uma estrutura condizente com os dados de espalhamento de raios X das estruturas cristalográficas e com os dados experimentais utilizados.

Palavras-chave: Fator de forma. Espalhamento de raios X. Campo de forças. Simulação molecular. Proteínas.

ABSTRACT

The prediction of the final structure of a protein (also called native structure) has been addressed by many research groups since the last century. This problem can be understood as how to predict the tertiary structure that a protein molecule assumes after the folding process from just the primary structure (the sequence of amino acids). Nowadays there are several models aiming at solving this problem, but very few of them are based on physical principles. Most of these models are template-based ones that search for similar amino acids sequences or analogous biological functions to predict the native structure. In the present work, however, we propose the development of a force field predict the form factor of proteins that does not entail the knowledge of any other model or template to do so. The form factor is an important aspect of the structure of a molecule in solution. Despite being a low-resolution method of analysis, in the sense that it does not provide details about the exact positions of the atoms inside the molecular structure (because of the mathematical operations needed to retrieve informations from the scattering data), it is an initial estimative of the volume occupied by this molecule and also a good initial path for uncovering how these molecules interact to each other in solution. The results show that the method presented here can predict a structure that agrees with the scattering data of the crystallographic structure and with available experimental data of x-ray scattering of proteins in solution.

Keywords: Form factor. Small-angle X ray scattering. Force field. Molecular simulation. Proteins.

ÍNDICE DE FIGURAS

FIGURA 1 - LIGAÇÃO AMÍDICA (PEPTÍDICA) DESTACADA EM VERMELHO ENTRE DOIS AMINOÁCIDOS ARBITRÁRIOS. A LIGAÇÃO OCORRE POR MEIO DA REAÇÃO QUÍMICA ENTRE UM GRUPO AMINA E UM GRUPO ÁCIDO CARBOXÍLICO. 3

FIGURA 2 - ILUSTRAÇÃO DA ESTRUTURA PRIMÁRIA DE UM POLIPEPTÍDIO. A ESTRUTURA PRIMÁRIA É CARACTERIZADA PELA SEQUÊNCIA LINEAR DE AMINOÁCIDOS. NA FIGURA SÃO COLOCADOS 4 AMINOÁCIDOS CONECTADOS POR MEIO DE 3 LIGAÇÕES PEPTÍDICAS. 4

FIGURA 3 - ILUSTRAÇÃO COM O DESENHO DE DUAS ESTRUTURAS SECUNDÁRIAS FORMADAS A PARTIR DO ESTABELECIMENTO DE LIGAÇÕES DE HIDROGÊNIO. A ESQUERDA A ESTRUTURA COMUMENTE CHAMADA DE ALFA HÉLICE E A DIREITA A ESTRUTURA CHAMADA FOLHA BETA.....5

FIGURA 4 - FIGURA REPRESENTANDO A ESTRUTURA TERCIÁRIA DE UMA PROTEÍNA. É POSSÍVEL OBSERVAR A PRESENÇA DAS ESTRUTURAS SECUNDÁRIAS, ALFA HÉLICES E FOLHAS BETA DOBRADAS UMAS SOBRE AS OUTRAS, CARACTERIZANDO A ESTRUTURA TERCIÁRIA DE UMA PROTEÍNA. NA FIGURA TRATA-SE DA PROTEÍNA BA2930 SINTETIZADA PELO MICRORGANISMO *BACILLUS ANTHRACIS.* 6

FIGURA 5 - ILUSTRAÇÃO REPRESENTANDO A ESTRUTURA QUATERNÁRIA DE UM POLIPEPTÍDIO. É POSSÍVEL OBSERVAR NA FIGURA 4 ESTRUTURAS TERCIÁRIAS CONECTADAS. UMA EM AZUL, UMA EM AMARELO, UMA EM VERMELHO E UMA EM ROSA. A CONEXÃO ENTRE ESTRUTURAS TERCIÁRIAS FORMA O COMPLEXO PROTEICO DE ESTRUTURA QUATERNÁRIA. A ILUSTRAÇÃO APRESENTA A ESTRUTURA QUATERNÁRIA DA HEMOGLOBINA. 7

FIGURA 6 - POSSÍVEL CONFIGURAÇÃO DA FUNÇÃO ENERGÉTICA DO PROCESSO DE *FOLDING* DE UMA PROTEÍNA. O EIXO À ESQUERDA, E, REPRESENTA A ENERGIA MÉDIA E O EIXO À DIREITA, Q, REPRESENTA A FRAÇÃO DE INTERAÇÕES DA ESTRUTURA NATIVA E DESCREVE A POSIÇÃO DE UM CONJUNTO DE ESTADOS NA FUNÇÃO ENERGÉTICA.. AS FLUTUAÇÕES ΔE E O INTERVALO DE ESTABILIDADE ΔE_S ENTRE UM ESTADO NÃO NATIVO, UM ESTADO GLOBULAR E O ESTADO NATIVO SÃO FUNÇÕES DE E E Q. APESAR DA EVOLUÇÃO TER LEVADO A UM AFUNILAMENTO DA FUNÇÃO ENERGÉTICA, EM DECORRÊNCIA DA INFLUÊNCIA CINÉTICA NO PROCESSO EXISTEM, NO FUNDO, ARMADILHAS CINÉTICAS (*DISCRETE TRAPS)*, ...10

FIGURA 7 - ESQUEMA DO EXPERIMENTO DE ESPALHAMENTO DE RADIAÇÃO DE BAIXO ÂNGULO. NA FIGURA K0 É O NÚMERO DE ONDA DA ONDA INCIDENTE, K É O NÚMERO DE ONDA DA ONDA ESPALHADA, Q É O VETOR DE ESPALHAMENTO E 2Θ O ÂNGULO DE ESPALHAMENTO. 14
FIGURA 8 - ESQUEMA DO CAMPO DE ESPALHAMENTO. NA FIGURA, L É O PONTO DE OBSERVAÇÃO E O É O CHAMADO CENTRO DE ESPALHAMENTO. O VETOR R INDICA A POSIÇÃO DE L EM RELAÇÃO A O E O VETOR R' INDICA A POSIÇÃO DE UMA CARGA NEGATIVA DA AMOSTRA. O CAMPO POTENCIAL ΦR REPRESENTA, NO CASO DO ESPALHAMENTO DE RAIOS X, A DENSIDADE ELETRÔNICA DE CARGAS NEGATIVAS DO MATERIAL. 16
FIGURA 9 - ÂNGULO DIÉDRICO (Ξ) INDICADO POR MEIO DA PROJEÇÃO CONSTRUÍDA NA FIGURA. A POSIÇÃO DOS ÁTOMOS É INDICADA PELOS QUATRO VÉRTICES PRESENTES NA FIGURA. A POSIÇÃO RELATIVA ENTRE OS QUATRO FORMA O ÂNGULO DIÉDRICO E É CONSIDERADA PARA O CÁLCULO DO POTENCIAL DIÉDRICO. 35
FIGURA 10 - EXEMPLO DE ARQUIVO *INPUT* GERADO PELO CÓDIGO GENINP. NA FIGURA OBSERVA-SE O TÍTULO GERADO PELO CÓDIGO NA PRIMEIRA LINHA, A QUANTIDADE DE RESÍDUOS DE AMINOÁCIDO, O TAMANHO DA ARESTA DA CAIXA QUE SERÁ UTILIZADA NA SIMULAÇÃO E O PH QUE SERÁ CONSIDERADO NA SIMULAÇÃO. ABAIXO DA SEGUNDA LINHA A SEQUÊNCIA DE AMINOÁCIDOS DE ACORDO COM A ABREVIAÇÃO EM TRÊS LETRAS DE CADA UM. 52
FIGURA 11 - ESTRUTURAS INICIAIS DA MOLÉCULA DE PROTEÍNA PROPOSTAS PARA AS SIMULAÇÕES DE DINÂMICA MOLECULAR. DA ESQUERDA PARA A DIREITA: HELICOIDAL, INTESTINO E ZIG-ZAG. AS ESTRUTURAS PROPOSTAS NÃO TEM RELAÇÃO FÍSICA COM A ESTRUTURA PRIMÁRIA DA PROTEÍNA, SÃO ESTRUTURAS PROPOSTAS COM O OBJETIVO DE CABEREM NA CAIXA DE SIMULAÇÃO SEM QUE A CAIXA SEJA GRANDE DEMAIS A PONTO DE ENCOLHER BRUSCAMENTE NO INÍCIO DA SIMULAÇÃO E CAUSAR UMA PARADA NO ALGORITMO DE DINÂMICA MOLECULAR. 54
FIGURA 12 - FLUXOGRAMA REPRESENTANDO A SEQUÊNCIA DE CÓDIGOS DE PROGRAMAÇÃO UTILIZADOS PARA A PREPARAÇÃO DOS ARQUIVOS NECESSÁRIOS PARA A SIMULAÇÃO DE DINÂMICA MOLECULAR. NO FLUXOGRAMA SÃO APRESENTADAS AS INFORMAÇÕES REQUERIDAS PELOS CÓDIGOS E AS INFORMAÇÕES FORNECIDAS PRO ELES. 55
FIGURA 13 - ESTRUTURA AO FINAL DA SIMULAÇÃO MOLECULAR DA LISOZIMA SEM LEVAR EM CONSIDERAÇÃO A MODIFICAÇÃO PARA AS FORÇAS HIDROFÓBICAS INDICADA NA

EQUAÇÃO 3.1.13. O FATOR H CONSIDERADO PARA AS FORÇAS HIDROFÓBICAS REDUZ O PARÂMETRO ENERGÉTICO DO POTENCIAL DE MIE, DESTE MODO DIMINUINDO A INTENSIDADE DA INTERAÇÃO ENTRE OS RESÍDUOS HIDROFÓBICOS E A ÁGUA. 60
FIGURA 14 - ESTRUTURA AO FINAL DA SIMULAÇÃO MOLECULAR DA LISOZIMA LEVANDO EM CONSIDERAÇÃO A MODIFICAÇÃO PARA AS FORÇAS HIDROFÓBICAS INDICADA NA EQUAÇÃO 3.1.13. O FATOR H CONSIDERADO PARA AS FORÇAS HIDROFÓBICAS REDUZ O PARÂMETRO ENERGÉTICO DO POTENCIAL DE MIE, DESTE MODO DIMINUINDO A INTENSIDADE DA INTERAÇÃO ENTRE OS RESÍDUOS HIDROFÓBICOS E A ÁGUA. 62
FIGURA 15 – DENSIDADE DA CAIXA DE SIMULAÇÃO AO LONGO DA SIMULAÇÃO DE DINÂMICA MOLECULAR PARA A LISOZIMA. AS SIMULAÇÕES FORAM FEITAS DURANTE 10,0 NANOSSEGUNDOS COM PASSOS DE 2,0 PICOSSEGUNDOS NO *ENSEMBLE* NPT A TEMPERATURA DE 300 K E PRESSÃO DE 1,0 BAR. 64
FIGURA 16 - EVOLUÇÃO DAS DENSIDADES PARA AS SIMULAÇÕES DE DINÂMICA MOLECULAR COM A LISOZIMA. SÃO APRESENTADOS OS PRIMEIROS PICOSSEGUNDOS DA SIMULAÇÃO PARA AVALIAÇÃO DO PERÍODO TRANSIENTE DA SIMULAÇÃO. AS SIMULAÇÕES PARTINDO DAS TRÊS ESTRUTURAS INICIAIS PROPOSTAS SÃO EXIBIDAS. 65
FIGURA 17 - DENSIDADE DA CAIXA DE SIMULAÇÃO AO LONGO DA SIMULAÇÃO DE DINÂMICA MOLECULAR PARA A ALBUMINA SÉRICA BOVINA. AS SIMULAÇÕES FORAM FEITAS DURANTE 10,0 NANOSSEGUNDOS COM PASSOS DE 2,0 PICOSSEGUNDOS NO *ENSEMBLE* NPT A TEMPERATURA DE 300 K E PRESSÃO DE 1,0 BAR. 66
FIGURA 18 - EVOLUÇÃO DAS DENSIDADES PARA AS SIMULAÇÕES DE DINÂMICA MOLECULAR COM A LISOZIMA. SÃO APRESENTADOS OS PRIMEIROS PICOSSEGUNDOS DA SIMULAÇÃO PARA AVALIAÇÃO DO PERÍODO TRANSIENTE DA SIMULAÇÃO. AS SIMULAÇÕES PARTINDO DAS TRÊS ESTRUTURAS INICIAIS PROPOSTAS SÃO EXIBIDAS. 67
FIGURA 19 - ESTRUTURAS AO FINAL DAS SIMULAÇÕES MOLECULARES PARA A LISOZIMA. NESSAS SIMULAÇÕES VARIOU-SE A ESTRUTURA INICIAL DA BIOMOLÉCULA PARA QUE FOSSE AVALIADA A INFLUÊNCIA DESSA ESTRUTURA INICIAL, UMA VEZ QUE É SABIDO EXISTIR INFLUÊNCIA CINÉTICA NO PROCESSO DE *FOLDING* DE UMA BIOMOLÉCULA. RESPECTIVAMENTE, DA ESQUERDA PARA DIREITA: HELICOIDAL, INTESTINO E ZIG ZAG..69
FIGURA 20 - EVOLUÇÃO DO RAIO DE GIRO PARA A LISOZIMA NAS TRÊS SIMULAÇÕES DE DINÂMICA MOLECULAR PARTINDO CADA VEZ DE UMA ESTRUTURA INICIAL DIFERENTE. O RAIO DE GIRO É UMA PROPRIEDADE GEOMÉTRICA QUE TRADUZ O QUÃO COMPACTA É A

ESTRUTURA. O RAIO DE GIRO DA LISOZIMA MEDIDO EXPERIMENTALMENTE VALE A 1,43 NM. 70
FIGURA 21 - ESTRUTURAS AO FINAL DAS SIMULAÇÕES MOLECULARES PARA A ALBUMINA SÉRICA BOVINA. NESSAS SIMULAÇÕES VARIOU-SE A ESTRUTURA INICIAL DA BIOMOLÉCULA PARA QUE FOSSE AVALIADA A INFLUÊNCIA DESSA ESTRUTURA INICIAL, UMA VEZ QUE É SABIDO EXISTIR INFLUÊNCIA CINÉTICA NO PROCESSO DE *FOLDING* DE UMA BIOMOLÉCULA. RESPECTIVAMENTE, DA ESQUERDA PARA DIREITA: HELICOIDAL, INTESTINO E ZIG ZAG. 71
FIGURA 22 - EVOLUÇÃO DO RAIO DE GIRO PARA A ALBUMINA SÉRICA BOVINA NAS TRÊS SIMULAÇÕES DE DINÂMICA MOLECULAR PARTINDO CADA VEZ DE UMA ESTRUTURA INICIAL DIFERENTE. O RAIO DE GIRO É UMA PROPRIEDADE GEOMÉTRICA QUE TRADUZ O QUÃO COMPACTA É A ESTRUTURA. O RAIO DE GIRO DA ALBUMINA MEDIDO EXPERIMENTALMENTE VALE A 2,98 NM. 72
FIGURA 23 – CURVA DE INTENSIDADE DE ESPALHAMENTO DE RAIOS-X PARA A MOLÉCULA DE PROTEINA LISOZIMA. SÃO APRESENTADAS NO GRÁFICO AS TRÊS CURVAS CORRESPONDENTES ÀS SIMULAÇÕES DE DINÂMICA MOLECULAR SENDO INICIADAS POR CADA ESTRUTURA INICIAL PROPOSTA, A CURVA DA ESTRUTURA CRISTALOGRÁFICA E A CURVA DE INTENSIDADE DE ESPALHAMENTO CALCULADA A PARTIR DA ESTRUTURA PREVISTA PELO MÉTODO CABS. 74
FIGURA 24 - CURVA DE INTENSIDADE DE ESPALHAMENTO DE RAIOS-X PARA A MOLÉCULA DE PROTEINA ALBUMINA SÉRICA BOVINA. SÃO APRESENTADAS NO GRÁFICO AS TRÊS CURVAS CORRESPONDENTES ÀS SIMULAÇÕES DE DINÂMICA MOLECULAR SENDO INICIADAS POR CADA ESTRUTURA INICIAL PROPOSTA, A CURVA DA ESTRUTURA CRISTALOGRÁFICA, OS DADOS EXPERIMENTAIS DE FRANCO (FRANCO ET AL., 2015) E A CURVA DE INTENSIDADE DE ESPALHAMENTO CALCULADA A PARTIR DA ESTRUTURA PREVISTA PELO MÉTODO CABS. 76

ÍNDICE DE TABELAS

TABELA 1 - PARÂMETROS DE INTERAÇÃO DE VAN DER WAALS E ELETROSTÁTICOS PARA A PRIMEIRA VERSÃO DO CAMPO DE FORÇAS. OS PARÂMETROS DA PARTÍCULA DE ÁGUA *COARSE-GRAINED* SÃO DO TRABALHO DE SHINODA ET AL. (SHINODA ET AL., 2006), OS PARÂMETROS DOS ÍONS SÃO DO CAMPO DE FORÇA OPLS (JORGENSEN ET AL., 1996) E OS PARÂMETROS DOS RESÍDUOS DE AMINOÁCIDO FORAM AJUSTADOS PARA QUE O RAIO DE GIRO DA ALBUMINA SÉRICA BOVINA CORRESPONDESSE AO EXPERIMENTAL DE CERCA DE 2,9 NM. 40

TABELA 2 - PARÂMETROS PARA AS INTERAÇÕES QUE CONSIDERAM O POTENCIAL DE LIGAÇÃO QUÍMICA E O POTENCIAL ÂNGULAR. NA TABELA SÃO APRESENTADAS AS CONSTANTES DESTES POTENCIAIS RETIRADAS DO TRABALHO DE ANGELESCU ET AL. E VOICESCU E IONESCU (ANGELESCU ET AL., 2007; VOICESCU & IONESCU, 2012). 42

TABELA 3 – VOLUME MOLECULAR E CONSTANTE DE LENNARD-JONES DOS RESÍDUOS DE AMINO ÁCIDO UTILIZADOS NAS FUNÇÕES QUE CARACTERIZAM O CAMPO DE FORÇAS. OS RESÍDUOS SÃO ASSUMIDOS COMO ESFERAS E O PARÂMETRO Σ DE LENNARD-JONES É ENTÃO O DIÂMETRO DA ESFERA. 43

TABELA 4 - VALORES DE pK_A PARA OS TERMINAIS CARBOXILA E AMINA DOS AMINO ÁCIDOS E PARA AS CADEIAS LATERAIS, QUANDO IONIZÁVEIS QUE FORAM UTILIZADOS COM A EQUAÇÃO DE HENDERSSON-HASSELBACH NO CÁLCULO DA CARGA DE CADA RESÍDUO NA MOLÉCULA DE PROTEÍNA. 47

TABELA 5 - HIDROPATIA DOS AMINO ÁCIDOS BASEADAS NA ESCALA CONSTRUÍDA POR KYTE E DOOLITTLE (KYTE & DOOLITTLE, 1982). AS HIDROPATIAS SÃO CONSIDERADAS NOS MODELOS A FIM DE MIMETIZAR AS FORÇAS HIDROFÓBICAS. UM PARÂMETRO FOI ADICIONADO NA INTERAÇÃO DE VAN DER WAALS COM BASE NA HIDROPATIA DE CADA AMINOÁCIDO. 48

TABELA 6 - RAIO DE GIRO DA LISOZIMA CALCULADO A PARTIR DA MÉDIA DOS ÚLTIMOS 8500 PICOSSEGUNDOS DAS SIMULAÇÕES DE DINÂMICA MOLECULAR PARA AS TRÊS ESTRUTURAS INICIAIS DIFERENTES. O VALOR RELATADO NA LITERATURA PARA O RAIO DE GIRO DA LISOZIMA É DE 1,43 NM. O DESVIO PERCENTUAL É CALCULADO SOBRE ESSE VALOR. 71

TABELA 7 - RAIO DE GIRO DA ALBUMINA SÉRICA BOVINA CALCULADO A PARTIR DA MÉDIA DOS ÚLTIMOS 8500 PICOSSEGUNDOS DAS SIMULAÇÕES DE DINÂMICA MOLECULAR PARA AS

TRÊS ESTRUTURAS INICIAIS DIFERENTES. O VALOR RELATADO NA LITERATURA PARA O RAIO DE GIRO DA ALBUMINA É DE 2,98 NM. O DESVIO PERCENTUAL É CALCULADO SOBRE ESSE VALOR. 72
TABELA 8 - ERRO QUADRÁTICO CALCULADO PARA AS TRÊS CURVAS DE INTENSIDADE DE ESPALHAMENTO PARA A PROTEÍNA LISOZIMA PREVISTAS PELO MÉTODO AQUI PROPOSTO E O ERRO CALCULADO PARA A ESTRUTURA PREVISTA PELO MÉTODO CABS. CADA CURVA DE CADA SIMULAÇÃO DE DINÂMICA MOLECULAR COMEÇANDO DE UMA ESTRUTURA DISTINTA FOI COMPARADA COM A CURVA CALCULADA PARA A ESTRUTURA CRISTALOGRÁFICA. O ERRO É CALCULADO DE ACORDO COM A FÓRMULA 4.3.1.....75
TABELA 9 - ERRO QUADRÁTICO CALCULADO PARA AS TRÊS CURVAS DE INTENSIDADE DE ESPALHAMENTO PARA A PROTEÍNA ALBUMINA SÉRICA BOVINA PREVISTAS PELO MÉTODO AQUI PROPOSTO E O ERRO CALCULADO PARA A ESTRUTURA PREVISTA PELO MÉTODO CABS. CADA CURVA DE CADA SIMULAÇÃO DE DINÂMICA MOLECULAR COMEÇANDO DE UMA ESTRUTURA DISTINTA FOI COMPARADA COM A CURVA CALCULADA PARA A ESTRUTURA CRISTALOGRÁFICA. O ERRO É CALCULADO DE ACORDO COM A FÓRMULA 4.3.1 77

LISTA DE SÍMBOLOS

$\vec{k}_0$	Vetor/número da onda incidente no experimento de espalhamento de raios-X
$\vec{k}$	Vetor/número da onda espalhada no experimento de espalhamento de raios-X
θ	Metade do ângulo de espalhamento
$\vec{q}$	Vetor de espalhamento
π	Perímetro da circunferência dividido pelo seu diâmetro
λ	Comprimento de onda
A_0	Amplitude da onda incidente no expermiento de espalhamento de raios-X
e	Constante de Euler
i	Unidade imaginária
$\vec{r}$	Vetor posição no espaço real
b, A(q)	Amplitude de espalhamento
∇^2	Operador Laplaciano
v	Parâmetro que especifica a intensidade de interação com o campo de espalhamento
φ	Campo escalar de espalhamento

ψ	Onda espalhada
W	Fluxo de partículas
Ω	Ângulo sólido
$I(q)$	Intensidade de espalhamento
$f(q)$	Comprimento de espalhamento
N	Número de partículas
Φ	Ângulo entre os vetores de espalhamento e de posição
v_i	Volume da região i
V_0	Volume total
v	Volume específico
p_{ij}	Função de probabilidade
ξ	Função de probabilidade normalizada
$P(q)$	Fator de forma
$S(q)$	Fator de estrutura
K	Constante de proporcionalidade
$p(r)$	Função de distribuição de distâncias
A_a	Amplitude de espalhamento da partícula no vácuo

A_s	Amplitude de espalhamento do volume excluído
A_b	Amplitude de espalhamento da camada de hidratação
ρ_s	Densidade do solvente
$\delta\rho_b$	Diferença entre a densidade do solvente e a primeira camda de hidratação
d	Distância máxima de resolução do experimento de espalhamento
$\vec{F}$	Vetor força
$\vec{p}$	Vetor momento
∇	Operador gradiente
U	Função potencial
U_{el}	Potencial eletrostático
q_i	Carga da partícula i
ϵ_0	Permissividade do vácuo
ϵ_r	Permissividade relativa
U_{LJ}	Potencial de Lennard-Jones
ε	Parâmetro energético do potencial de Mie
σ	Parâmetro geométrico do potencial de Mie
U_B	Potencial de Buckingham

A, B e C	Parâmetros do potencial de Buckingham
$U_{ligação}$	Potencial de ligação química
k_r	Constante harmônica do potencial de ligação química
$\vec{r}_0$	Vetor posição de equilíbrio
U_θ	Potencial angular
k_θ	Constante harmônica do potencial angular
θ_{ijk}	Ângulo formado entre as partículas i,j e k
$\theta_{0,ijk}$	Ângulo de equilíbrio entre as partículas i,j e k
U_{Mie}	Potencial de Mie
K_a	Constante de equilíbrio da dissociação ácida
K_b	Constante de equilíbrio da ionização da base
[ ]	Concentração
pK_a	Cologaritmo da constante de equilíbrio de dissociação ácida
pK_b	Cologaritmo da constante de equilíbrio de ionização da base
pH	Cologaritmo da concentração de íons H^+
pOH	Cologaritimo da concentração de íons OH^-
v_j	Parâmetro da equação de Henderson-Hasselbach

α_j	Fração de grupos ionizados na equação de Henderson-Hasselbach
H_{ij}	Fator H de hidrofobia
h_i	Hidropatia do resíduo i
u	Velocidade da partícula
m	Massa da partícula
k	Constante de Boltzmann
T	Temperatura
R_G	Raio de giro
N_B	Número disponível de dados
$\vec{r}_{cm}$	Vetor posição do centro de massa
Er	Erro quadrático
κ_{CG}	Dado da proteína *coarse-grained*
κ_{CR}	Dado da proteína cristalográfica

SUMÁRIO

1. INTRODUÇÃO 1

1.1. JUSTIFICATIVA 1

1.2. OBJETIVOS 2

1.3. ESTRUTURA DO TRABALHO 2

2. REVISÃO BIBLIOGRÁFICA 3

2.1. ESTRUTURA PROTEICA 3

2.2. ESPALHAMENTO DE BAIXO ÂNGULO DE RAIOS-X 13

2.3. SIMULAÇÃO MOLECULAR 25

2.4. CAMPOS DE FORÇA 31

3. MATERIAIS E MÉTODOS 37

3.1. DESENVOLVIMENTO TEÓRICO 37

3.2. MÉTODO PARA AS SIMULAÇÕES 51

4. RESULTADOS 59

4.1. HIDROFOBICIDADE 60

4.2. INFLUÊNCIA DA ESTRUTURA INICIAL 63

4.3. CÁLCULO DA INTENSIDADE DE ESPALHAMENTO DE RAIOS-X 73

4.4. SUGESTÕES PARA TRABALHOS FUTUROS 78

BIBLIOGRAFIA 79

APÊNDICE A – CÓDIGO GENINP 90

APÊNDICE B – CÓDIGO CF_LRP 96

APÊNDICE C – CÓDIGO STR 116

APÊNDICE D – CÓDIGO TRJ 124

1. INTRODUÇÃO

1.1. Justificativa

Um dos problemas que mais desafiou grupos de pesquisa no último século foi o de prever a estrutura terciária de proteínas a partir de sua estrutura primária (Zhang, 2008). Apesar da técnica que permite o acesso à estrutura primária (sequenciamento de aminoácidos) de um polipeptídio já estar bastante dominada, o processo de *folding* (ou dobramento) dessa biomolécula ainda guarda muitas lacunas para serem estudadas e desvendadas.

Neste âmbito, um primeiro passo, apesar de ainda apresentar baixa resolução, seria o de conseguir prever o fator de forma dessas moléculas em solução. O objetivo em um experimento de espalhamento de raios X é, a partir dos dados de intensidade de espalhamento no espaço recíproco, recuperar informações do espaço real. Por conta de o experimento ser realizado em um domínio limitado e da transformada entre o espaço real e o recíproco ser tomada de 0 a infinito, existe uma perda de resolução. Não obstante, o fator de forma não só fornece informações a respeito da forma, mas também da maneira com que todas as moléculas presentes na solução interagem umas com as outras (Svergun & Koch, 2003). Neste sentido, há diversas tentativas e modelagens para executar a tarefa oposta, ou seja, a partir de dados de espalhamento de raios-X, ou do fator de forma, os algoritmos recriam uma estrutura que seja condizente com aqueles dados. Dessa forma, a motivação deste trabalho foi propor um método que possuísse algum embasamento físico para que essa previsão da estrutura fosse realizada e os dados de espalhamento funcionassem como um passo verificador, não como entrada do algoritmo.

1.2. Objetivos

O objetivo deste trabalho é propor a construção de um campo de forças *coarse-grained* que traduza como os aminoácidos interagem entre eles com base em princípios físicos, e verificar, por meio de simulações moleculares, a eficácia deste campo na predição da estrutura de proteínas em solução.

1.3. Estrutura do trabalho

O presente trabalho foi dividido de forma a abranger os fundamentos da estruturação de proteínas, do fator de forma e da utilização da simulação molecular para a previsão desejada.

O trabalho está dividido da seguinte maneira:

1. Revisão bibliográfica: será apresentado o panorama atual dos avanços científico tecnológicos a respeito do tema;
2. Materiais e métodos: será apresentado todo desenvolvimento teórico do campo de forças proposto justificando as escolhas e decisões tomadas ao longo do processo a partir de resultados preliminares;
3. Resultados e discussão: serão apresentados os resultados obtidos com a aplicação do campo de forças desenvolvido e as conclusões de sua aplicação;

2. Revisão Bibliográfica

2.1. Estrutura proteica

Proteínas são cadeias polipeptídicas formadas por resíduos de aminoácidos que podem ou não possuir modificações pós traducionais, como a glicosilação e a formação de pontes dissulfeto. Diz-se resíduo, pois, ao formar a ligação amídica, como é exibido na Figura 1, é produzida uma molécula de água, alterando sutilmente os aminoácidos que se conectaram. Portanto, neste panorama, chamamos então cada aminoácido dentro da cadeia proteica de resíduo.

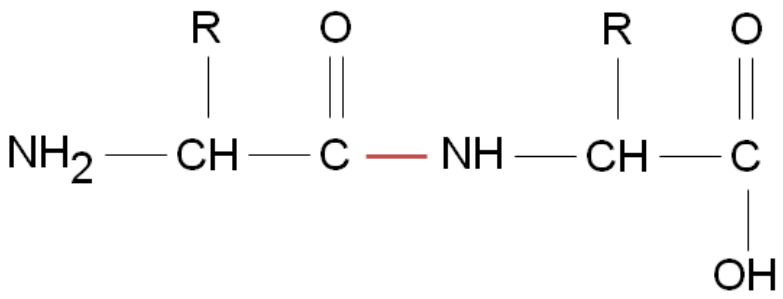

Figura 1 - Ligação amídica (peptídica) destacada em vermelho entre dois aminoácidos arbitrários. A ligação ocorre por meio da reação química entre um grupo amina e um grupo ácido carboxílico.

Não é mais um mistério que a sequência de aminoácidos de uma proteína é determinada pelo gene que codifica esta biomolécula no DNA. Apesar de ainda restarem algumas questões a respeito da degeneração do processo traducional, uma vez que um mesmo aminoácido pode ser identificado por mais de um códon (sequência de três bases nitrogenadas), o sequenciamento de uma proteína já é obtido mais facilmente. Um fato interessante que se especula

a respeito da degeneração é que talvez ela possua relação com o processo subsequente, nomeado *folding* ou dobramento da proteína (Hu et al., 2013). A este processo será dada especial atenção por conta dos objetivos aqui propostos.

A estruturação, ou *folding* ou dobramento de uma proteína é didaticamente caracterizada por quatro fases. Essas fases são, por sua vez, identificadas a partir das conformações assumidas pela cadeia polipeptídica:

- Estrutura primária: sequência linear ordenada de aminoácidos traduzida a partir de um gene.

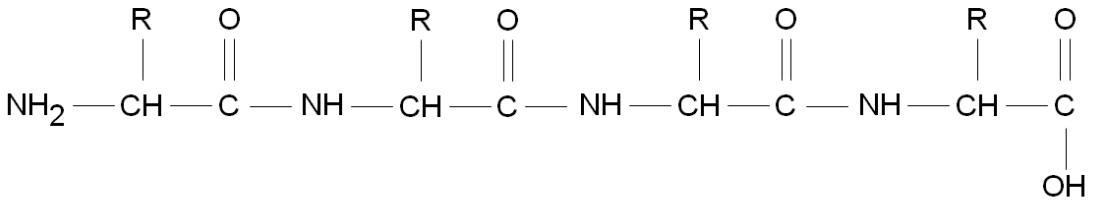

Figura 2 - Ilustração da estrutura primária de um polipeptídio. A estrutura primária é caracterizada pela sequência linear de aminoácidos. Na figura são colocados 4 aminoácidos conectados por meio de 3 ligações peptídicas.

- Estrutura secundária: a cadeia da biomolécula assume estruturas características como alfa hélices e folhas beta (Pauling & Corey, 1951).

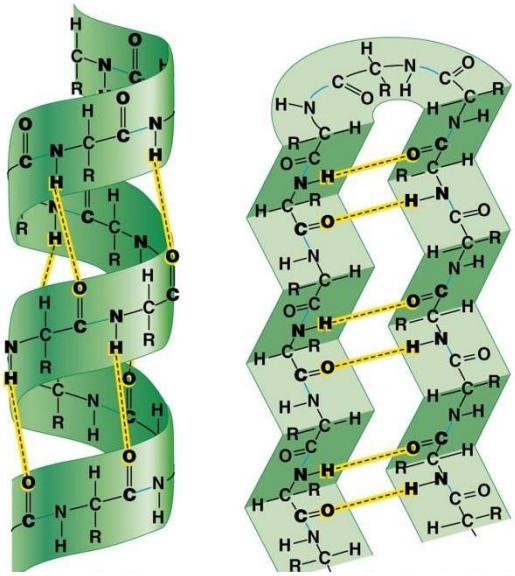

Figura 3 - Ilustração com o desenho de duas estruturas secundárias formadas a partir do estabelecimento de ligações de hidrogênio. A esquerda a estrutura comumente chamada de alfa hélice e a direita a estrutura chamada folha beta.

Fonte: *FREEMAN, S.; HARRINGTON, M.; SHARP, J. Biological Science. Pearson Education Canada, 2ª ed., 2012.*

- Estrutura terciária: a cadeia proteica já com as estruturas secundárias formadas passa a se dobrar sobre si mesma

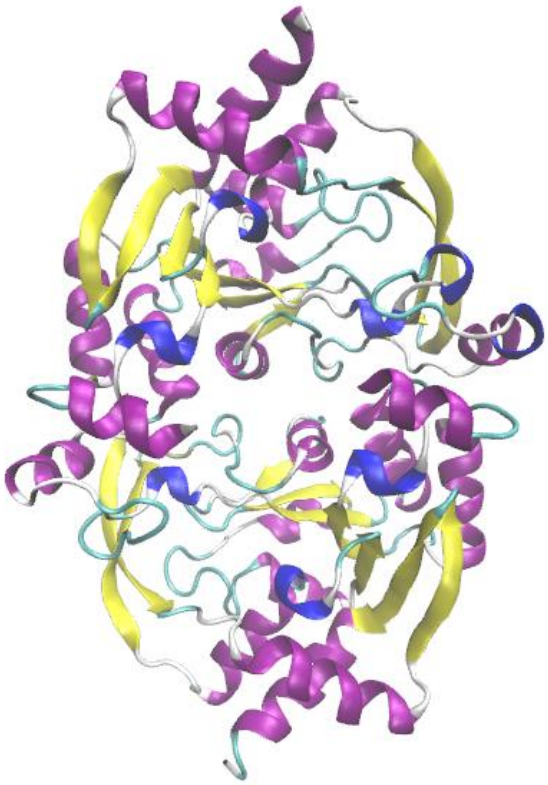

Figura 4 - Figura representando a estrutura terciária de uma proteína. É possível observar a presença das estruturas secundárias, alfa hélices e folhas beta dobradas umas sobre as outras, caracterizando a estrutura terciária de uma proteína. Na figura trata-se da proteína BA2930 sintetizada pelo microrganismo *Bacillus anthracis.*

Fonte: KLIMECKA, M. M.; CHRUSZCZ, M.; FONT, J.; SKARINA, T.; SHUMILIN, I.; ONOPRYIENKO, O.; POREBSKI, P. J.; CYMBOROWSKI, M.; ZIMMERMAN, M. D.; HASSEMAN, J.; GLOMSKI, I. J.; LEBIODA, L.; SAVCHENKO, A.; EDWARDS, A.; MINOR, W. Structural analysis of a putative aminoglycoside N-acetyltransferase from Bacillus anthracis. Journal of Molecular Biology, v. 410, p. 411-423, 2011.

- Estrutura quaternária: união de estruturas terciárias para formar complexos proteicos. Um exemplo de estrutura quaternária é a hemoglobina, proteína responsável pelo transporte de oxigênio no corpo humano (Lukin et al., 2003).

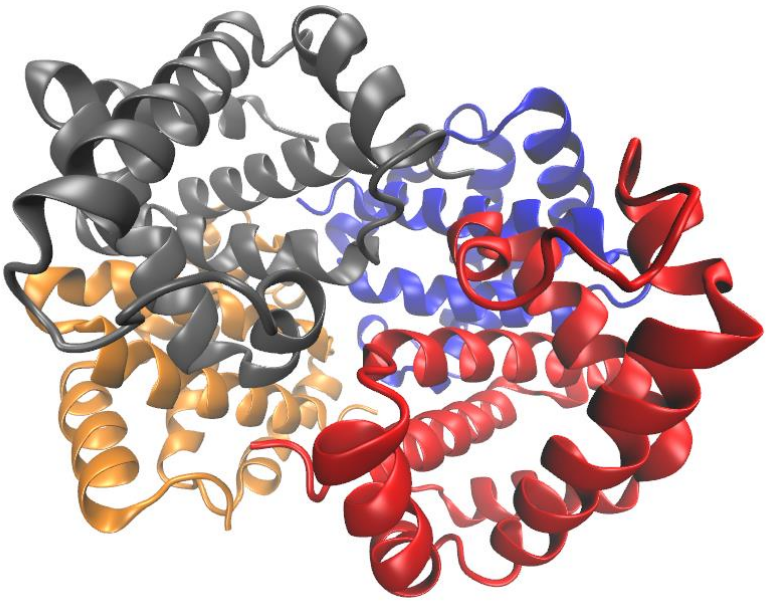

Figura 5 - Ilustração representando a estrutura quaternária de um polipeptídio. É possível observar na figura 4 estruturas terciárias conectadas. Uma em azul, uma em amarelo, uma em vermelho e uma em rosa. A conexão entre estruturas terciárias forma o complexo proteico de estrutura quaternária. A ilustração apresenta a estrutura quaternária da hemoglobina.

Fonte: SAVINO, C.; MIELE, A. E.; DRAGHI, F.; JOHNSON, K. A.; SCIARA, G.; BRUNORI, M.; VALLONE, B. Pattern of cavities in globins: The case of human hemoglobina. *Biopolymers,* v. 91, p. 1097-1107, 2009.

.

A despeito da fina caracterização estrutural, o processo em si e como ele ocorre ainda se mantêm nebulosos quanto à completa descrição e investigação de alguns porquês. Por conta de algumas doenças já serem reconhecidamente causadas por falhas no processo de *folding* de algumas proteínas, muito se estuda a respeito da estruturação dessas biomoléculas visando a entender o motivo para essas falhas ocorrerem (Agorogiannis, 2004). Ademais, no âmbito da engenharia, conhecer este processo fornecerá ferramentas para que se

consiga controlá-lo e manipulá-lo de acordo com as demandas tecnológico científicas, permitindo que avanços sejam feitos no campo da biotecnologia.

Por conta da busca por essa compreensão, durante o século XX diversos grupos de pesquisa envolveram-se com este problema (Zhang, 2008; Clementi, 2008; Levitt & Warshel, 1975). Na primeira metade daquele século, muito era discutido a respeito da função das pontes dissulfeto, estruturas formadas entre cisteínas, que provocam o dobramento de algumas proteínas. No entanto, quando Anfinsen, no ano de 1973, publicou sua hipótese termodinâmica a respeito do *folding*, o foco dessas investigações mudou da bioquímica para a física de polímeros em conjunto com a computação (Compiani & Capriotti, 2013).

A hipótese é de simples compreensão. Anfinsen afirmou que, dadas as condições fisiológicas de atuação de uma proteína, ou seja, pH, solvente, força iônica, presença de outros componentes, temperatura e outros, a forma nativa desta proteína é caracterizada pela estrutura que minimiza a energia do sistema, ou seja, a conformação final é determinada pelas interações entre os átomos presentes na cadeia polipeptídica e, portanto, pela sequência de aminoácidos que caracterizam a biomolécula (Anfinsen, 1973).

Com isso, abriu-se um novo campo para a investigação do dobramento de proteínas e, neste novo campo, espaço para a inserção das técnicas computacionais baseadas na termodinâmica, ou seja, passou-se a, a partir de uma função da energia, simular o processo virtualmente para que o mínimo de energia fosse atingido.

No entanto, quatro anos antes da hipótese termodinâmica ser formulada e publicada, foi formulado o que hoje é chamado de paradoxo de Levinthal (Levinthal, 1969). Seria estranho imaginar que a superfície do espaço configuracional do processo de *folding*, espaço onde se caracterizam os estados energéticos da proteína ao longo do processo de dobramento, fosse composta de apenas um único mínimo global e que ele fosse atingido puramente pelo controle termodinâmico do processo, ou seja, como a hipótese

termodinâmica apontaria. Imaginou-se, portanto, que alguma influência cinética existiria também.

Se for considerada, por exemplo, uma proteína com 4000 átomos, teríamos 12000 graus de liberdade, um para cada direção do espaço tridimensional caso não existissem restrições. Conquanto, sabe-se que há limitações para o movimento desses átomos. É estimado que, para um polipeptídio que contém 150 resíduos, existam 450 graus de liberdade, 300 devido a rotações e 150 devido a ângulos relativos de ligação (Levinthal, 1969). Isto posto, a desconfiança de um controle e/ou influência cinética torna-se ainda mais provável, uma vez que é possível imaginar, dada a grande quantidade de graus de liberdade, que existam diversos mínimos locais ao longo do espaço configuracional.

Dessa forma, a hipótese termodinâmica e o paradoxo de Levinthal parecem ser conflitantes, já que propõem diferentes controles para o mesmo processo. O controle cinético determinaria que, durante o processo de estruturação da proteína, a sequência de estados ou o caminho percorrido, e nesse sentido faz-se referência ao caminho trilhado no espaço configuracional, seria determinante para atingir o estado nativo. Por outro lado, o controle termodinâmico determinaria que apenas os estados inicial e final seriam determinantes e o caminho percorrido entre os dois não seria decisivo para que o estado nativo fosse atingido. Contudo, de acordo com a teoria de *folding* proteico, esses dois controles podem existir concomitantemente neste processo, resolvendo assim a aparente contradição (Onuchic & Wolynes, 2004; Onuchic et al., 1997; Bryngelson et al., 1995).

O processo de *folding* de uma proteína é, sem dúvida, complexo. É esperado, por exemplo, que este processo envolva uma complexa rede de reações elementares, no entanto, relações simples, como uma relação linear, já foram observadas empiricamente na modelagem do processo cinético. Essa simplicidade é característica não de qualquer polímero com qualquer sequência de aminoácido, mas um resultado do processo de evolução (Onuchic & Wolynes, 2004).

É colocada a hipótese de que, baseada na teoria da evolução das espécies, ocorreu um afunilamento da função matemática que caracteriza a energia das diferentes conformações que a molécula de proteína pode assumir. A diferença energética associada com o diferente posicionamento dos resíduos de aminoácido ou pelo posicionamento do solvente permite que algumas estruturas sejam mais estáveis do que outras. No entanto, apesar do afunilamento, a especificidade da estrutura em relação à energia ainda é muito baixa, ou seja, existe uma grande variedade de estruturas que também são mínimos da função da energia. Portanto, essas estruturas agem como armadilhas cinéticas, trazendo ao processo uma complexidade cinética muito grande. Para exemplificar isto, observemos na Figura 6 o gráfico simplificado da função energética do processo de *folding* proteico.

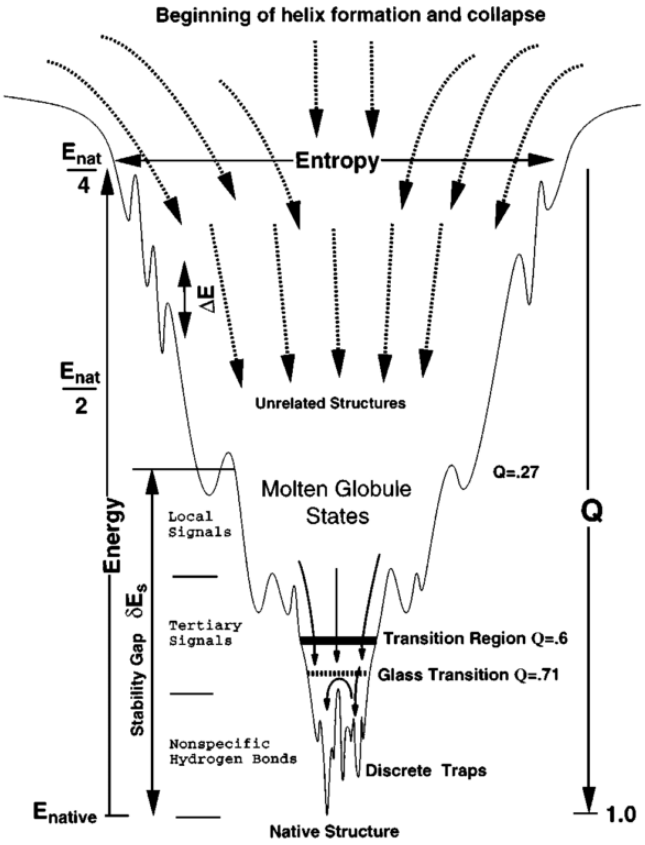

Figura 6 - Possível configuração da função energética do processo de *folding* de uma proteína. O eixo à esquerda, E, representa a energia média e o eixo à direita, Q, representa a fração de interações da estrutura nativa e descreve a posição de um conjunto de estados na função energética. As flutuações ΔE e o intervalo de estabilidade δE_S entre um estado não nativo, um estado globular e o estado nativo são funções de E e Q. Apesar da evolução ter levado a um afunilamento da função energética, em decorrência da influência cinética no processo existem, no fundo, armadilhas cinéticas (*discrete traps)*,

Fonte: Onuchic et al., 1997.

Nota-se que, apesar da possibilidade de existência de um mínimo global, existem mínimos locais próximos, as chamadas armadilhas cinéticas. Este fato levou a uma complicação bastante grande para a modelagem teórica do processo, afinal, torna-se uma tarefa bastante difícil modelar um processo com armadilhas cinéticas.

Diferentemente dos processos e sistemas mais comuns e menos complexos que atingem o equilíbrio termodinâmico a partir da minimização da energia, sistemas proteicos não podem ser definidos com essa simplicidade, uma vez que o processo de *folding* não apresenta controle puramente termodinâmico nem puramente cinético. É postulado que a energia relacionada ao contato entre dois resíduos que, na forma nativa, não estão em contato na estrutura molecular da proteína ou a energia de um resíduo e uma estrutura secundária que não está presente na forma nativa podem ser tomadas como variáveis aleatórias, ou seja, as interações que não fazem parte da forma nativa da proteína remetem a uma estrutura aleatória de um heteropolímero. Em um segundo postulado, afirma-se que quando uma estrutura que ocorre na forma nativa é formada, a energia dessa estrutura deve contribuir para a estabilidade da molécula. (Bryngelson et al., 1995). A partir destes dois postulados, Bryngelsol e Wolynes formularam o princípio da mínima frustração, que afirma que proteínas, por meio da evolução, foram aprimoradas de modo a minimizar a energia de suas formas nativas e também minimizar a quantidade de armadilhas cinéticas. Este princípio auxiliou na melhoria das funções construídas para traduzir perfil energético do processo de *folding* (Onuchic & Wolynes, 2004).

A minimização da frustração, ou teoria da mínima frustração, fornece um meio de modelar o processo através de algoritmos e também ajuda a melhorar as funções de energia para o processo de *folding.*

Com este aparente conflito fora de questão, passa-se ao desafio de formular e implementar métodos que consigam abarcar esse fenômeno para reproduzi-lo,

torná-lo previsível e, talvez, controlável. Em decorrência do alto número de graus de liberdade e da grande complexidade que um sistema proteico apresenta, a primeira, e arrisca-se dizer, única opção para tal tarefa, é que se direcionem os esforços para ferramentas computacionais. Neste sentido, com o objetivo de observar fenômenos mecânicos, ou seja, o dobramento de uma molécula, espera-se que entre em cena a ferramenta da dinâmica molecular (MD na sigla em inglês). Apesar disso, além das técnicas de simulação por MD, surgiram algumas tentativas de máquinas de aprendizado ou inteligência artificial também.

Dentre os modelos teóricos desenvolvidos pode-se citar: o colapso hidrofóbico (HC na sigla em inglês) (Dill, 1985); mecanismos hierárquicos, como o modelo de difusão-colisão (DC na sigla em inglês) (Panchenko et al., 1996) e o posterior modificado de *foldon* difusão-colisão (FDC na sigla em inglês) (Compiani et al., 2004); mecanismo de nucleação-condensação (NC na sigla em inglês) (Itzhaki et al., 1995); modelo de núcleo estendido (EN na sigla em inglês) (Fersht, 2000); entre outros.

Já para aplicação computacional, podem-se encontrar dois grandes grupos de métodos para a previsão da estrutura proteica: os métodos que utilizam estruturas já conhecidas para servir de moldes para as ainda não conhecidas, chamados de modelos de *template*, e os métodos que utilizam princípios físicos somente para tentar prever a estrutura proteica partindo apenas da sequência de aminoácidos do polipeptídio, nomeados de modelagem livre (Zhang, 2008). Em tempo: nos modelos de *template* encontramos ainda uma subdivisão entre os que utilizam a função biológica da molécula como um possível indicador da estrutura nativa e os que não utilizam. Chamada de busca homóloga, baseia-se na conservação evolutiva da estrutura proteica atrelada a sua função no organismo. Apesar de a evolução alterar as espécies e suas características, as proteínas de espécies próximas parecem manter a estrutura molecular similar, uma vez que a função que desempenham está intimamente conectada a esta estrutura e, portanto, podem ser uma boa forma para estimativas iniciais (Zhang, 2008).

Aqui, no entanto, propõe-se um campo de forças para ser utilizado em simulações de dinâmica molecular, com o objetivo de prever a estrutura de proteínas e, a partir desta estrutura, calcular o fator de forma, uma grandeza que caracteriza algumas particularidades estruturais dessas moléculas, sem serem necessários *templates*.

A vantagem aqui é a não necessidade da existência de uma proteína homóloga com sua estrutura cristalina já determinada, ou seja, o método pode ser aplicado independentemente da existência ou não de proteínas homólogas já estudadas. Ademais, por se tratar de um método que utiliza simulação de dinâmica molecular, foram propostos potenciais de interação entre as moléculas presentes em solução, fornecendo assim um embasamento físico para o processo que pode, eventualmente, ser refinado a partir de modelos mais complexos e que representem de maneira mais satisfatória essas interações intermoleculares.

2.2. Espalhamento de baixo ângulo de raios-X

O fator de forma é uma grandeza que surge da teoria de espalhamento de radiação de baixo ângulo. Dessa forma, retomemos brevemente os fundamentos deste fenômeno. Todas as equações e fundamentos aqui apresentados estão baseados nos textos apresentados por Guinier e Feigin (Guinier & Fournet, 1955; Feigin & Svergun, 1987).

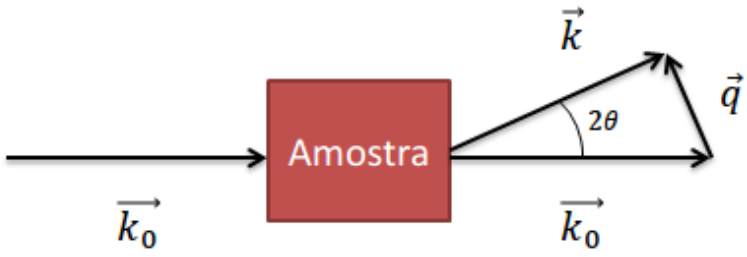

Figura 7 - Esquema do experimento de espalhamento de radiação de baixo ângulo. Na figura $\vec{k}_0$ é o número de onda da onda incidente, $\vec{k}$ é o número de onda da onda espalhada, $\vec{q}$ é o vetor de espalhamento e 2θ o ângulo de espalhamento.

Na Figura 7 podemos identificar algumas grandezas importantes para a modelagem. São elas: $\vec{k}$, $\vec{k}_0$, $\vec{q}$ e θ. k e k_0 são os chamados números de onda para a onda espalhada e para a onda incidente respectivamente, $|\vec{q}|$ é o módulo do vetor de espalhamento e θ é metade do ângulo de espalhamento. Define-se o vetor $\vec{q}$ da seguinte maneira:

$$\vec{q}=\vec{k}-\vec{k}_0 \tag{2.2.1}$$

Sabe-se que, por definição, o número de onda, k, é escrito como:

$$|\vec{k}|=\frac{2\pi}{\lambda} \tag{2.2.2}$$

em que λ é o comprimento de onda. Aplicando a lei dos cossenos e algumas modificações algébricas, consegue-se obter a conhecida equação para o módulo do vetor espalhamento:

$$|\vec{q}|=\frac{4\pi}{\lambda}\text{sen}(\theta) \tag{2.2.3}$$

Dadas as definições preliminares acima, pode-se prosseguir para a avaliação propriamente dita do espalhamento de onda. Para simplicidade das equações,

considerou-se, sem perda de generalidade, a dispersão de uma onda plana monocromática.

Antes de prosseguir para o equacionamento, é importante que entendamos o que ocorre na esfera microscópica do fenômeno de espalhamento de onda.

A onda incidente, ao atingir a matéria, interage com partículas carregadas (prótons ou elétrons). Essas partículas, por sua vez, passam a vibrar com a mesma frequência da onda incidente, tornando-se, assim, fontes secundárias de ondas esféricas. A interferência entre a onda incidente e as ondas esféricas geradas pelas partículas carregadas constrói um padrão característico. É esperado, claro, que as ondas secundárias também interajam com partículas carregadas, criando um efeito dominó na excitação dessas partículas e um padrão de interferência bastante complexo. Apesar disso, para simplicidade, considera-se para o equacionamento apenas a interferência entre a onda incidente e as ondas secundárias, essa aproximação é válida para fracas dispersões (Landau & Lifshitz, 1985).

Retomando o equacionamento, considera-se uma onda plana monocromática e a aproximação acima, em que apenas a onda incidente e a secundária são responsáveis pelo padrão de interferência, pode-se escrever que a onda resultante é traduzida pela seguinte expressão:

$$\psi = A_0 e^{i\vec{k}_0\vec{r}} + \left(\frac{A_0 b}{|\vec{r}|}\right) e^{i\vec{k}\vec{r}} \qquad (2.2.4)$$

em que A_0 e A_0b/r são as amplitudes das ondas incidente e secundária, respectivamente, r é a grandeza espacial, i é a unidade imaginária e b é uma constante nomeada amplitude de espalhamento. Quanto mais forte for a interação entre a onda e o material, maior será a constante b.

O cálculo da onda espalhada, ou seja, a soma da superposição das várias ondas esféricas geradas pela formação de fontes secundárias de onda no material, é baseada na teoria de perturbações. A onda espalhada $\psi(\vec{r})$ é a solução da equação de onda seguinte:

$$\left[\nabla^2+\left|\vec{k}_0\right|^2+v\varphi(\vec{r})\right]\psi(\vec{r})=0 \qquad (2.2.5)$$

em que ∇^2 é o operador laplaciano, v é um parâmetro que especifica a intensidade da interação com o campo φ e $\varphi(r)$ é um campo escalar definido como densidade de dispersão. Essa função, um campo potencial, é determinada dentro de um espaço finito e em alguns casos pode também depender do tempo. No caso do espalhamento de raios X, por exemplo, esse campo representa a densidade de distribuição de cargas elétricas negativamente carregadas no material, conforme exibido na Figura 8.

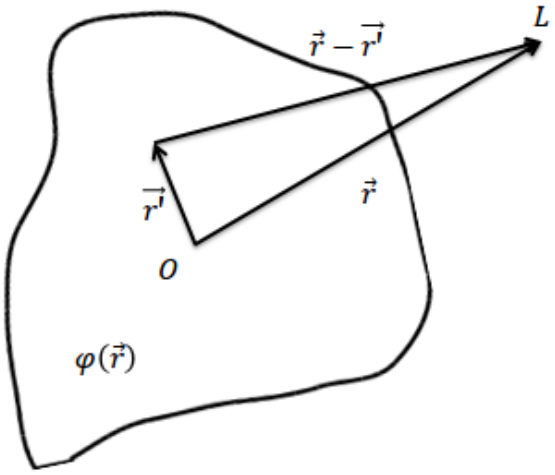

Figura 8 - Esquema do campo de espalhamento. Na figura, L é o ponto de observação e O é o chamado centro de espalhamento. O vetor $\vec{r}$ indica a posição de L em relação a O e o vetor $\vec{r'}$ indica a posição de uma carga negativa da amostra. O campo potencial $\varphi(\vec{r})$ representa, no caso do espalhamento de raios X, a densidade eletrônica de cargas negativas do material.

A solução da equação 2.2.5 é feita por meio de uma série de potências em v. O termo independente desta série é a onda incidente e o termo subsequente é a primeira aproximação da onda dispersa, considerando apenas uma interação com as partículas carregadas do material. Quando restringimos a solução à

primeira aproximação, a amplitude em um ponto L é igual à soma das duas componentes:

$$\psi^{(0)}(\vec{r})+\psi^{(1)}(\vec{r})=A_0 e^{i\vec{k}_0\vec{r}}+\frac{A_0 v}{4\pi}\int \frac{e^{i\vec{k}\left(\vec{r}-\vec{r}'\right)}}{\left|\vec{r}-\vec{r}'\right|}\varphi\left(\vec{r}'\right)e^{i\vec{k}_0\vec{r}'}d\vec{r}' \qquad (2.2.6)$$

O ponto L de observação, mostrado na Figura 8, é usualmente posicionado de maneira que $\left|\vec{r}-\vec{r}'\right|$ seja grande quando comparado com as dimensões da área dispersada. Este é o caso da difração de Fraunhofer, chamada de espalhamento de baixo ângulo. Com esta hipótese, conseguimos simplificar a equação 2.2.6 para obtermos:

$$\psi^{(0)}(\vec{r})+\psi^{(1)}(\vec{r})=A_0 e^{i\vec{k}_0\vec{r}}+\frac{A_0 v e^{i\vec{k}\vec{r}}}{4\pi|\vec{r}|}\int \varphi\left(\vec{r}'\right)e^{-i\vec{q}\vec{r}'}d\vec{r}' \qquad (2.2.7)$$

Comparando então as equações 2.2.7 e 2.2.4 conclui-se que a constante b, anteriormente chamada de amplitude de espalhamento, pode ser então calculada da seguinte forma:

$$b=A(\vec{q})=\frac{v}{4\pi}\int \varphi(\vec{r})e^{-i\vec{q}\vec{r}}d\vec{r} \qquad (2.2.8)$$

Sendo assim, $A(\vec{q})$ é a amplitude de espalhamento provocada pelo campo $\varphi(\vec{r})$. Trata-se da primeira aproximação de Born (Born, 1926), em outras palavras, a aproximação de um único espalhamento.

Em um experimento de dispersão de radiação, entretanto, consegue-se medir apenas o fluxo de partículas dispersadas (fótons ou nêutrons) por unidade de ângulo sólido. Escrevemos então, que a intensidade do espalhamento é proporcional ao quadrado do módulo da amplitude de espalhamento:

$$\frac{dW}{d\Omega}=I(\vec{q})=\left|A(\vec{q})\right|^2 \qquad (2.2.9)$$

em que W é o fluxo de partículas, Ω é o ângulo sólido.

Contudo, quando tratamos de uma solução, como é o caso de experimentos com proteínas, deve-se descrever um conjunto de átomos em fase líquida, ou seja, sem disposição ordenada de partículas. Começamos modelando um sistema que consiste de N moléculas idênticas com comprimento de espalhamento $f(\vec{q})$. Podemos, portanto, escrever que a intensidade de espalhamento pode ser calculada da seguinte forma:

$$I(\vec{q}) = \sum_{i=1}^{N} \sum_{j=1}^{N} f_i(\vec{q}) f_j(\vec{q})\, e^{i\vec{q}\vec{r}_{ij}} \qquad (2.2.10)$$

em que $\vec{r}_{ij}$ é a distância entre as moléculas i e j. Separando os termos nos quais i = j e tomando a média configuracional sobre todas as moléculas, temos:

$$\overline{I(\vec{q})} = \overline{\sum_{i=1}^{N} f_i^2(\vec{q})} + \overline{\sum \sum_{i\neq j} f_i(\vec{q})\, f_j(\vec{q}) \cos\left(\vec{q}\vec{r}_{ij}\right)} = \overline{N}\,\overline{f^2(\vec{q})} + \overline{f(\vec{q})}^2\, \overline{\sum \sum_{i\neq j} \cos\left(\vec{q}\vec{r}_{ij}\right)} \qquad (2.2.11)$$

O valor médio de N é escrito na expressão porque no caso geral, o número de moléculas pode variar. Para que consigamos calcular o valor da soma dupla na equação 2.2.11, uma função de probabilidade, p, é conveniente. A função dará a probabilidade de encontrar uma partícula em um volume dv_i na posição $\vec{r}_i$ e, simultaneamente outra partícula em um volume dv_j na posição $\vec{r}_j$. Tomando a média da função cosseno sob todas as orientações produz:

$$\overline{\cos(\vec{q}\vec{r})} = \int_0^{\pi} \cos[qr\cos(\phi)] \frac{\mathrm{sen}(\phi)}{2} d\phi = \frac{\mathrm{sen}(qr)}{qr} \qquad (2.2.12)$$

em que ϕ é o ângulo entre os vetores $\vec{q}$ e $\vec{r}$ e $\frac{\mathrm{sen}(\phi)d\phi}{2}$ representa a probabilidade do ângulo estar no intervalo entre ϕ e $\phi+d\phi$. Logo obtemos que:

$$\overline{\sum \sum_{i\neq j} \cos\left(\vec{q}\vec{r}_{ij}\right)} = \iint_{V_0} \frac{\mathrm{sen}(qr_{ij})}{qr_{ij}} p_{ij} dv_i dv_j \qquad (2.2.13)$$

em que V_0 é o volume total da amostra sendo irradiada.

É mais conveniente utilizar em lugar de p uma função de probabilidade normalizada $\xi = p/v^2$, em que $v = V_0/N$ é o volume molecular médio por partícula. Substituindo na expressão, tem-se:

$$\overline{I(\vec{q})}=\bar{N}\overline{f^2(\vec{q})}+\overline{f(\vec{q})}^2\iint_{V_0}\frac{\mathrm{sen}(qr_{ij})}{qr_{ij}}\frac{dv_i}{v}\frac{dv_j}{v}-\overline{f(\vec{q})}^2\iint_{V_0}\left[1-\xi\left(r_{ij}\right)\right]\frac{\mathrm{sen}(qr_{ij})}{qr_{ij}}\frac{dv_i}{v}\frac{dv_j}{v} \qquad (2.2.14)$$

Examinando a expressão 2.2.14 é possível notar que o primeiro termo corresponde ao espalhamento da partícula com volume V_0 e densidade constante. Este termo é negligenciável quando $q > 2\pi/D_0$, situação em que $D_0 \approx V^{\frac{1}{3}}$. Com respeito a segunda integral deve-se perceber que por conta da normalização, $\xi(r_{ij})$ tende para a unidade quando r_{ij} tende para o infinito. Portanto, podemos escrever:

$$\int_{V_0}\frac{dv_j}{v}\int_{V_0}\left[1-\xi(r_{ij})\right]\frac{\mathrm{sen}(qr_{ij})}{qr_{ij}}\frac{dv_i}{v}\approx\bar{N}\int_0^{\infty}\left[1-\xi(r)\right]\frac{\mathrm{sen}(qr)}{qr}\frac{4\pi r^2}{v}dr \qquad (2.2.15)$$

em que a primeira integral é V_0/v e coordenadas esféricas foram usadas na segunda.

Se ainda o comprimento de dispersão da molécula for considerado esfericamente simétrico, então $\overline{f(\vec{q})}^2=\overline{f^2(\vec{q})}$, portanto escrevemos:

$$I(q)=\bar{N}F^2(q)\left\{1-\frac{1}{v}\int_0^{\infty}\left[1-\xi(r)\right]\frac{\mathrm{sen}(qr)}{qr}\frac{4\pi r^2}{v}dr\right\} \qquad (2.2.16)$$

em que $I(q)$ e $F^2(q)$ representam as intensidades médias de espalhamento do sistema e de cada molécula respectivamente. A expressão 2.2.16 representa a intensidade de espalhamento em função do fator de forma de uma partícula, assim como da função estatística que governa o arranjo das partículas na amostra, o chamado fator de estrutura. Ela é frequentemente escrita da seguinte forma:

$$I(\vec{q})=KP(\vec{q})S(\vec{q}) \qquad (2.2.17)$$

em que $P(q)$ é o chamado fator de forma e $S(q)$ o fator de estrutura.

De maneira simplificada, o fator de forma caracteriza o formato de uma única partícula, enquanto o fator de estrutura caracteriza a distribuição dessas

partículas em solução. Quando unimos estes dois conceitos ao aparato teórico desenvolvido na termodinâmica estatística, observamos que é possível estudar os potenciais de interação entre moléculas a partir de informações sobre a intensidade de espalhamento pelo conjunto de partículas. (Svergun & Koch, 2003).

Diversos parâmetros característicos podem ser obtidos diretamente da curva de intensidade de espalhamento como: massa molecular, volume excluído de partícula, que é o volume inacessível a outras partículas por conta da presença de uma partícula, e raio de giro. Este último é o mais clássico e conhecido parâmetro extraído diretamente dos dados de espalhamento. O raio de giro fornece uma medida do tamanho da proteína e do grau de compactação dos resíduos de aminoácido ao redor do centro de massa da proteína, dependendo de como o raio de giro é calculado (Kikhney & Svergun, 2015).

Este parâmetro pode ser estimado a partir da aproximação de Guinier, que considera uma aproximação para o gráfico do logaritmo da intensidade de espalhamento contra q^2. Essa aproximação consiste em dizer que para valores baixos de q, a intensidade depende unicamente de q e do raio de giro. Neste mesmo gráfico ainda é possível calcular a intensidade de espalhamento para q=0. Este valor, frequentemente caracterizado por $I(0)$ é proporcional à massa molecular da proteína e à concentração de proteína na solução (Kikhney & Svergun, 2015).

Ademais, por meio da transformada inversa de Fourier, conseguimos calcular a função de distribuição de distâncias, ou a comumente chamada $p(r)$. Essa função é um histograma das distâncias entre todos os possíveis pares de partículas em relação a uma partícula. A função de distribuição se relaciona com a intensidade de espalhamento por meio da seguinte equação (Kikhney & Svergun, 2015):

$$p(r)=\frac{r^2}{2\pi^2}\int_0^{\infty}\frac{q^2 I(q)\,\mathrm{sen}(qr)}{qr}\,dq \qquad (2.2.18)$$

Por definição essa função é identicamente nula em q=0 e é não negativa.

O cálculo da intensidade de espalhamento de radiação para soluções proteicas foi proposto segundo diversas modelagens. Os primeiros modelos usavam metodologias de tentativa e erro, comparando diferentes padrões de dispersão de radiação com dados experimentais. Os modelos sempre tentavam associar a forma das moléculas com formas geométricas regulares como prismas, elipsoides ou então com conjuntos de contas (pequenas esferas empacotadas de maneira regular).

De maneira cronológica, o primeiro modelo que avançou de maneira significativa em relação à tentativa e erro foi o modelo desenvolvido por Stuhrmann (Stuhrmann, 1970). Nele, o formato da molécula era representado por uma função que desempenhava o papel de um envelope. Este, por sua vez, descrevia os limites da forma das partículas em coordenadas esféricas. A este modelo dá-se o nome de modelo envelope e, pode-se dizer que ele contribuiu significativamente para o progresso dos métodos que buscavam solucionar a interpretação dos dados gerados pela dispersão de radiação (Svergun & Koch, 2003).

O uso deste método, no entanto, era limitado a formas geométricas simples. Avançando na linha do tempo nos deparamos com os modelos de contas (Chacón et al., 1998; Svergun, 1999), que se utilizam do grande poder de processamento dos computadores para reviver as ideias de tentativa e erro. Neste método, usualmente um volume esférico com um diâmetro máximo estipulado era preenchido com pequenas contas (esferas muito pequenas com raio fixo). A partir de uma configuração inicial, a disposição das pequenas esferas era alterada por meio de simulações Monte Carlo, que são estocásticas e, portanto, pouco se considera a respeito dos aspectos físicos envolvidos no processo, uma vez que não é considerado um potencial de interação, mas uma função estocástica.

A principal limitação para este método era a hipótese de que todas as partículas teriam a mesma densidade. Um modelo mais versátil e recente é o dos pseudo-resíduos (Svergun et al., 2001). Neste, a proteína pode ser representada por um conjunto de pseudo-resíduos. O número de resíduos é

conhecido, já que o sequenciamento de um polipeptídio é uma técnica dominada atualmente. A tarefa, portanto, é encontrar as coordenadas destes resíduos para que os dados de dispersão de radiação estejam de acordo com as posições de cada um.

Os pseudo-resíduos são alocados de maneira aleatória e são reposicionados por meio de algoritmos genéticos, até que estejam de acordo com os dados experimentais. Novamente, por conta da falta de informações físicas e da multiplicidade de possíveis respostas dadas pelas simulações Monte Carlo, seria interessante adicionar conceitos físicos aos modelos para que tenhamos algum respaldo teórico em relação aos resultados obtidos. A proposta neste trabalho é justamente construir um modelo teórico que leve em conta alguns conceitos físicos para a previsão do fator de forma de proteínas.

Mais recentemente, Zhang e colaboradores (Zhang et al., 2011) uniram a modelagem *coarse-grained* ao cálculo da intensidade de espalhamento de raios X. No trabalho, os autores se baseiam em estruturas cristalográficas do banco de dados PDB e, a partir de simulações de flexibilidade, calculam a intensidade de espalhamento de raios X para melhorar a etapa de *threading* dos algoritmos de seleção de *templates*.

No entanto, os modelos até aqui construídos dependem de dados existentes de espalhamento de raios X para então calcularem uma estrutura condizente com estes dados. A vantagem do método aqui proposto é a necessidade apenas do modelo de interação entre os aminoácidos e da sequência primária da proteína. A partir destas duas informações propusemos um campo de força *coarse-grained* para ser utilizado em simulações de dinâmica molecular clássica para a previsão da estrutura terciária de uma proteína. No modelo proposto, os dados de espalhamento de raios-X são utilizados como verificação da forma atingida pela biomolécula na simulação de dinâmica molecular, diferente dos algoritmos citados anteriormente que se baseiam nos dados de espalhamento para então calcularem uma estrutura condizente com esses dados.

Vale ressaltar que o campo de forças proposto utiliza modelos não estocásticos, ou seja, não se trata de uma reorganização dos resíduos a partir de uma função estocástica, mas sim da tentativa de, a partir da sequência primária, atingir estrutura terciária com base em um modelo de interação intermolecular e intramolecular.

Dessa forma unimos o poder de amostragem computacional atual com princípios físicos, explorou-se a técnica de simulação de dinâmica molecular em conjunto com o *software* CRYSOL (Svergun et al., 1995), que calcula a intensidade de espalhamento com base na configuração molecular atingida ao final da simulação (Svergun & Koch, 2003).

O cálculo do espalhamento de radiação poderia ser conduzido utilizando-se apenas a fórmula de Debye:

$$I(q)=\sum_{i=1}^{N}\sum_{j=1}^{N} f_i f_j \frac{\operatorname{sen}(\vec{q}\vec{r}_{ij})}{\vec{q}\vec{r}_{ij}} \qquad (2.2.19)$$

em que f é o comprimento de espalhamento, caso utilizássemos a estrutura cristalográfica da proteína. No entanto, deseja-se chegar a essa estrutura a partir da estrutura primária por meio de simulação molecular.

Em um experimento de SAXS, considerando uma solução diluída com proteína, sal e solvente, a intensidade de espalhamento é medida em função do ângulo de espalhamento. Por conta das diversas configurações, posições e orientações das partículas, a intensidade é isotrópica e proporcional à média da intensidade de espalhamento de uma única partícula sobre todas as orientações possíveis. Pode-se escrever então o seguinte (Svergun et al., 2001):

$$I(q) = \overline{\left|A_a(\vec{q}) + \rho_s A_s(\vec{q}) + \delta\rho_b A_b(\vec{q})\right|^2} \qquad (2.2.20)$$

em que A_a é a amplitude de espalhamento da partícula no vácuo, A_s é a amplitude de espalhamento do volume excluído e A_b é a amplitude de espalhamento da camada de hidratação. Como a densidade eletrônica da

camada de hidratação pode ser diferente da densidade eletrônica do solvente, mesmo após subtrair o espalhamento causado pelo solvente é importante considerar essa diferença entre as densidades.

A principal vantagem em utilizar o CRYSOL, neste caso, é a possibilidade de estabelecer uma densidade diferente para a camada de solvatação próxima ao soluto que se deseja estudar (Svergun et al., 1995) tornando o cálculo do espalhamento mais acurado.

Por fim, é importante ressaltar que a resolução é inversamente proporcional ao módulo do vetor $\vec{q}$. Escrevemos, pois, que a mínima distância analisada por meio de um experimento de SAXS é (Svergun et al., 2001):

$$d=\frac{2\pi}{q} \qquad (2.2.21)$$

em que d é a distância e q o módulo do vetor transferência de momento. Ademais, como é medida a intensidade de espalhamento, para acessar a função de probabilidade que rege as interações no sistema ou a densidade eletrônica da molécula estudada são necessárias operações matemáticas que fazem com que essa medição perca resolução.

Por conta disso, a técnica de espalhamento atraiu menor atenção para o campo da previsão de estrutura proteica, principalmente por conta da baixa resolução fornecida por um experimento de SAXS (10-50 Å). A técnica fornece apenas informações sobre a forma de moléculas de proteína através de histogramas de distância, sem conseguir fornecer informações a respeito da posição exata dos átomos na molécula (Zhang et al., 2011).

2.3. Simulação molecular

Quando tratamos das ciências da natureza clássicas, grande parte de seus escopos e estudos embasam-se na chamada hipótese do contínuo. Essa hipótese trata de estabelecer uma maneira para que interpretemos, nestes domínios, como o próprio nome sugere, a matéria como um contínuo, sem espaços vazios.

N ão obstante, desde que grandes nomes da ciência como Daniel Bernoulli, Rudolf Clausius, James Clerk Maxwell, Ludwig Boltzmann, Josiah Willard Gibbs dentre outros, passaram a estudar a ciência que procurava compreender os aspectos dinâmicos e, porque não dizer, comportamentais da matéria, avançou-se para a descri ção atômica e não mais contínua (Pathria & Beale, 1996).

Em 1738, Daniel Bernoulli publicou *Hydrodynamica*, que estabeleceu a base para o desenvolvimento da teoria cinética dos gases. Em 1859, após a leitura sobre difusão molecular de Rudolf Clausius, Maxwell formulou a distribuição de velocidades moleculares de Maxwell, que é considerada a primeira lei da mecânica estatística (Basil, 2002). Nos anos de 1870, Boltzmann inicia os trabalhos e avança consideravelmente a mecânica estatística com o teorema H, a teoria de transporte, o equilíbrio térmico e a equação de estado de gases. Gibbs, em 1884, cunha o termo mec ânica estatística (Gibbs, 1884) e em 1902 a formaliza com seu livro *Elementary Principles in Statistical Mechanics*.

A partir da abordagem estat ística da mecânica e da teoria de *ensembles*, o problema dos muitos corpos torna-se trabalhável do ponto de vista computacional, permitindo que possamos explorá-lo com base no comportamento atômico. Posto isto, conseguimos traduzir, dessa forma, propriedades observadas no contínuo como consequência de observações e interações entre átomos, ou seja, no discreto. (Alder & Wainwright, 1959).

Na então fusão entre o aspecto estatístico colocado por Gibbs na teoria de *ensembles* (Gibbs, 1902), consolidando assim, a mecânica estatística, com os computadores, nasce a modelagem de simulação de dinâmica molecular. Em particular, no escopo deste trabalho, fez-se uso da técnica de simulação molecular clássica, uma vez que é a única opção, em dinâmica molecular, capaz de representar sistemas proteicos (Meller, 2001).

Os problemas que podem ser enfrentados com essa ferramenta computacional são diversos. Dentre eles pode-se citar o estudo da dinâmica de moléculas, o estudo de estabilidade com ligantes (Alavi et al., 2017; Walker et al., 2017), o estudo de possíveis ligantes (Suganya et al., 2017), o desenvolvimento de novas moléculas ou fármacos (Nandy et al., 2017) e o estudo a respeito da energia relacionada com ligações ou metaestados configuracionais (Matubayasi, 2017). É notável a expansão do nível exploratório que se atingiu com dinâmica molecular. Em essência, alterou-se a relação entre o experimental e o teórico, apesar de os computadores executarem os cálculos com base em um modelo, sendo assim um método teórico, grandezas físicas podem ser medidas, justificando o uso da expressão “experimento computacional” (Meller, 2001).

A técnica de simulação molecular consiste na idealização ou na tentativa de reproduzir o *bulk* do material que se deseja estudar. Apesar disso, já existem aplicações que procuram reproduzir interfaces e sistemas em confinamento (Franco et al., 2016). Em suma, estabelece-se um volume limite, que pode ser cúbico, dodecaédrico, ortorrômbico e etc. e, dentro deste volume, são colocados modelos de moléculas e/ou partículas que interagem segundo um campo de forças. Com o objetivo de garantir que se reproduza o *bulk*, as condições de contorno do volume idealizado, usualmente, são consideradas ou periódicas ou refletivas, ou seja, no primeiro caso, idealizam-se outros volumes ao redor do principal que simulam o mesmo sistema. Logo, quando uma molécula ultrapassa a barreira do volume central, a mesma molécula entrará no domínio central de maneira diametralmente oposta, por conta dos volumes secundários que rodeiam o principal. No segundo caso, os limites funcionam como anteparos de colisões perfeitamente elásticas ou, fazendo uma analogia

com a radiação, como espelhos perfeitos. Desse modo, moléculas que se aproximam dos limites são rebatidas de modo perfeito (Allen & Tildesley, 2009).

A técnica clássica, por sua vez, fundamenta-se na resolução, por meio de métodos numéricos, das equações de movimento de Newton equa ção 2.3.1 (Allen & Tildesley, 2009). A cada iteração, uma nova disposição dos átomos na simulação é calculada a partir da configuração anterior e da avaliação das forças a que estes átomos estão submetidos. Essas forças, por sua vez, são calculadas por meio de um potencial de interação, também chamado de campo de forças (Berendsen et al., 1995).

$$\vec{F}=\frac{d\vec{p}}{dt} \qquad (2.3.1)$$

em que $\vec{F}$ é a força, $\vec{p}$ o momento linear e t o tempo.

Este campo de for ças, por sua vez, é uma função potencial conservativa que permite o cálculo da força ($\vec{F}$) a qual est á submetida cada partícula na simulação (Berendsen et al., 1995). Em decorrência da propriedade conservativa deste campo, podem-se calcular as forças de maneira imediata a partir do gradiente desta função potencial, como mostra a equação 2.3.2:

$$\vec{F}=-\vec{\nabla}U \qquad (2.3.2)$$

Dentre as situa ções que podem ser tratadas com essa técnica, encontram-se o cálculo de propriedades termodinâmicas e de equilíbrio da estrutura molecular. Com o desenvolvimento da mecânica estatística e a teoria dos *ensembles*, notou-se que essas propriedades eram médias estatísticas independentes da dinâmica do sistema, logo, poderiam ser calculadas por qualquer método de amostragem, dentre eles a simula ção de dinâmica molecular ou a simulação Monte Carlo (Meller, 2001). Podemos também citar o estudo de mudanças na estrutura proteica ao redor do domínio da estrutura nativa, mudanças que podem explicar, por exemplo, a forma com que um ligante opera em uma mol écula proteica, ou a razão de um ligante causar a disfun ção do polipeptídio, como quando uma molécula de monóxido de

carbono liga-se à hemoglobina (Kuczera, 1996). Ademais, abre-se o campo exploratório a respeito do processo de *folding* dessas biomoléculas, uma vez que, a partir de um potencial de interação, pode-se estruturar a simulação para que a proteína parta da estrutura primária e dobre-se com base nas intera ções propostas pelo modelo de interação (Brooks, 1998). Em tempo, conseguiu-se uma maior compreensão do problema de ancoramento (em inglês, *the docking problem*), ou seja, encontrar a geometria e a afinidade do ligante (Rosenfeld et al., 1995).

Tratando de simulação de proteínas, no entanto, cujo tamanho pode variar muito (encontram-se proteínas com dezenas a centenas de aminoácidos) (Bradley et al., 2005), deve-se tomar cuidado com a carga computacional exigida. Não apenas isso, mas é importante ter em mente que devemos ser capazes de reproduzir a realidade: portanto, n ão teremos apenas a proteína, mas também algum solvente a envolvendo, fazendo com que a quantidade final de partículas seja muito maior do que apenas a quantidade de partículas que representa a biomolécula. Por esta razão, é interessante ter em mãos técnicas que possibilitem a modelagem computacional destes compostos, dado que a capacidade de processamento pode vir a ser uma limitação, mesmo nos dias de hoje (Blanco et al., 2013).

Essa carga computacional é medida não só em número de partículas contidas na simulação, mas também na escala de tempo do fenômeno que se deseja estudar (Blanco et al., 2013). Quando a capacidade computacional é um fator limitante, um modo de reduzir a demanda computacional, mas ainda assim conseguir reproduzir a realidade, é substituir o modelo atomístico (AA – *all atom*), onde cada átomo na simulação é representado por uma partícula, por modelos *coarse-grained* (CG) (Blanco et al., 2013; Takada, 2012; Tozzini, 2005). Nestes modelos convertemos, por exemplo, um res íduo de aminoácido em apenas uma partícula (Saunders & Voth, 2013; Takada, 2012; Tozzini, 2005). Desse modo, reduzimos a quantidade de partículas e, portanto, a quantidade de equações de movimento que o simulador deve calcular a cada passo numérico. É importante ressaltar, no entanto, que mesmo para simulações que envolvem prote ínas, ou seja, moléculas que contém um

número grande de átomos, existem problemas que já são enfrentados com simulação molecular atomística, ou seja, considerando cada partícula como um átomo.

Com a simula ção de dinâmica molecular, portanto, possuímos uma ferramenta que, a partir de um potencial de interação, consegue amostrar diferentes configurações da proteína. É importante notar que, diferentemente das simulações Monte Carlo sem restrições, que conseguem alterar a posição dos resí duos sem respeitar a estrutura primária da proteína, assim gerando múltiplas configurações que conseguem traduzir os dados de dispersão de radiação (Svergun & Koch, 2003), a simulação de dinâmica molecular mantém as ligações estabelecidas na estrutura linear do polipeptídio. Logo, em conjunto com o *software* CRYSOL (Svergun et al., 1995), gera-se configurações de equilíbrio diversas baseadas no potencial de interação para o cálculo do fator de forma que respeitam as ligações primárias da biomolécula. Assim sendo, conseguimos introduzir diversos aspectos físicos para o método que queremos desenvolver para a previsão do fator de forma de proteínas.

Diferentemente dos métodos citados na seção teórica sobre espalhamento de raios-X, em que dados de espalhamento são utilizados para o cálculo de estruturas proteicas, tarefa oposta do m étodo proposto neste trabalho, há m étodos que se utilizam da simplificação de campos de força *coarse-grained* para prever a estrutura de prote ínas. Apesar destes métodos não necessariamente utilizarem a técnica de espalhamento de raios-X para a comparação de seus resultados, como será utilizado neste trabalho, estes métodos são apropriados para que consigamos comparar os resultados obtidos a partir do método aqui proposto.

Dentre os m étodos que se utilizam de simulação computacional com campos de for ça *coarse-grained* para a previs ão de estruturas de proteínas, podemos citar o CABS (na sigla, em ingl ês, significa carbono alfa – carbono beta – modelo de grupo funcional lateral) (Kolinski, 2004; Kolinski et al., 2013), que utiliza-se de simula ções Monte Carlo e a representação *coarse-grained* para prever a estrutura de prote ínas; e o UNRES (na sigla, em ingl ês, significa

resíduos unidos) (Liwo et al., 2005; Liwo et al., 2010) que utiliza simula ções de din âmica molecular e otimiza ções por meio do m étodo *simulated annealing* para encontrar o m ínimo da função de energia do processo de *folding* proteico para prever a estrutura de prote ínas.

A vantagem do m étodo aqui proposto em relação ao CABS é a utilização de simulações de dinâmica molecular, ou seja, um método que se baseia em potenciais de interação e parâmetros com significado físico ao invés de simulações estocásticas, além de o CABS possuir a limitação de boas previsões p ara proteínas com n úmero menor ou igual a 120 resíduos de aminoácidos, enquanto o método aqui proposto, a priori, não possui limitações neste sentido. Em relação ao UNRES, propusemos um campo de forças *coarse-grained* com um grau de simplifica ção maior, que será explicitado e caracterizado na seção do desenvolvimento teórico do campo. Deste modo, propusemos um campo de forças para simulações de dinâmica molecular com menos partículas na simulação o que acarreta em menor carga computacional.

Pela facilidade de acesso aos servidores do CABS, utilizaremos os resultados da previsão dele para comparação com os resultados obtidos pelo método proposto neste trabalho.

Já possuímos, desse modo, a ferramenta de amostragem que restringe e respeita a estrutura primária dessas moléculas e dois m étodos já consolidados na literatura científica para que consigamos estabelecer comparações com o método aqui proposto. Falta estudarmos de que maneira se dá a interação delas em fase líquida para que consigamos reproduzir da maneira mais próxima do real possível a estrutura final desses polipeptídios. Para tanto passaremos agora a estudar os possíveis campos de forças para estes sistemas.

2.4. Campos de força

Antes de abordarmos os campos de for ça focados na aplicação de dinâmica molecular, é interessante que entendamos a teoria e o embasamento físico destas ferramentas.

A din âmica, ou mais geralmente a mecânica, é a parte da física que estuda o comportamento dos corpos e seus movimentos. Existiram diversas abordagens para descrever este comportamento, desde o tempo dos grandes fil ósofos gregos (Heilbron, 2016). No entanto, as abordagens que desejamos atingir aqui para que o conceito de campo de força torne-se mais claro são as formulações gerais da mecânica clássica.

Dessa forma, é de bom grado revisarmos alguns conceitos da chamada mecânica Newtoniana e entender os motivos que levaram ao desenvolvimento das formulações gerais.

Na modelagem Newtoniana temos o estabelecimento das três leis de Newton e, por conta do aparato matemático envolvido, uma abordagem dependente do sistema de coordenadas adotado e da identificação das forças envolvidas na situação que se deseja estudar (Lemos, 2007).

Portanto, torna-se interessante, para que as resoluções façam-se mais gerais, que se determinem o que são as hoje chamadas de coordenadas generalizadas. Com elas, o problema fica independente do sistema de coordenadas. Com isso não é mais necessário que se pré-estabeleça um sistema de orientação, basta adotar um que seja capaz representar de maneira unívoca o vetor posição de cada partícula e que consiga satisfazer identicamente todos os vínculos presentes no problema (Lemos, 2007).

Unindo as coordenadas generalizadas ao princípio de D’Alembert, consegue-se chegar a duas novas abordagens da mecânica clássica que não são mais dependentes do sistema de orientação nem da identificação das forças

envolvidas no problema: a mecânica Lagrangeana e a mecânica Hamiltoniana. De modo sucinto, a principal alteração de modelagem entre a mecânica newtoniana e as duas formulações gerais citadas é a não mais dependência na identificação das forças, mas a percepção da existência de uma função potencial de interação que origina as forças. Nesse sentido, basta que sejam apontadas as interações para que as forças surjam na modelagem do problema (Lemos, 2007).

Com essas formulações gerais, portanto, não é mais necessário que saibamos calcular e identificar cada força envolvida, basta que sejam apontados os potenciais de interação. À vista disso, a modelagem de sistemas atômicos, por exemplo, torna-se mais simples e geral. É necessário apenas que identifiquemos quais são as interações que ocorrem entre os átomos para conseguirmos estudar a dinâmica destes sistemas.

Podemos ent ão considerar, por exemplo, um sistema contendo N partículas interagentes. Neste caso, de maneira simplificada e geral, assumimos que as interações se deem entre conjuntos de átomos no sistema, de forma que podemos escrever a equa ção 2.4.1:

$$U = \sum U_1(\vec{r}_l) + \sum\sum U_2(\vec{r}_l,\vec{r}_j) + \sum\sum\sum U_3(\vec{r}_l,\vec{r}_j,\vec{r}_k) + \ldots \qquad (2.4.1)$$

em que s ão considerados potenciais de interação que agem em uma partícula, por exemplo a ação de um campo externo, pares de partículas, trios de partículas e assim em diante (Allen & Tildesley, 2009).

Apesar de a equa ção 2.4.1 seguir infinitamente até que sejam considerados todos os possíveis arranjos de partículas que se traduzem em um potencial, o termo mais importante desta expressão é o potencial de pares de partículas. Este potencial depende apenas da distância entre os átomos considerados no par (Allen & Tildesley, 2009). De maneira geral, ele pode ser separado em três parcelas: o potencial eletrostático, o potencial de van der Waals e o potencial de ligação.

O potencial eletrostático é resultado do campo elétrico gerado pelas partículas eletricamente carregadas. De maneira geral seguimos o modelo de Coulomb (Jorgensen et al., 1996) para representar esta parcela. A equa ção 2.4.2 abaixo denota este modelo:

$$U_{el}=\frac{q_i q_j}{4\pi\epsilon_0\epsilon_r|\vec{r}|} \tag{2.4.2}$$

em que q é a carga da partícula, ϵ_0 é a permissividade do vácuo e ϵ_r é a constante dielétrica relativa ao vácuo.

O potencial de van der Waals, mais comumente chamado de força de van der Waals, traduz as interações de dipolo. Este é geralmente estudado com experimentos envolvendo argônio em decorrência da ausência de carga líquida nos átomos, assim anulando a contribuição eletrostática. Desse modo, diversos modelos já foram propostos para traduzir essa interação. Os dois mais comuns são o potencial de Lennard-Jones 12-6 (Jones, 1924) e o potencial de Buckingham (Buckingham, 1938), ambos traduzidos pelas expressões apresentadas abaixo:

$$U_{LJ}=4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{|\vec{r}_{i,j}|}\right)^{12}-\left(\frac{\sigma_{ij}}{|\vec{r}_{i,j}|}\right)^{6}\right] \tag{2.4.3}$$

$$U_B=Ae^{-B|\vec{r}_{i,j}|}-\frac{C}{|\vec{r}_{i,j}|^6} \tag{2.4.4}$$

No potencial de Lennard-Jones, σ é a distância para a qual o potencial é nulo e ε traduz a profundidade do po ço de energia. É possível identificar essas duas informações a partir da interpretação gráfica do potencial de Lennard-Jones. Já para o potencial de Buckingham A, B e C são constantes ajustáveis sem significado físico.

Ainda na parcela que considera os potenciais de pares, resta o potencial de liga ção qu ímica. Existem diversos modelos que tentam reproduzir o comportamento de uma ligação (van Gunsteren et al., 1996; Morse, 1929; Ferguson, 1995; Warner, 1972). Dentre eles, o mais comum é o potencial

harmônico, seguindo a clássica lei de Hooke. A equação 2.4.5 a seguir mostra este modelo:

$$U_{ligação} = \sum \frac{k_r}{2} \left(\left| \vec{r}_{I,j} \right| - \left| \vec{r}_{0I,j} \right| \right)^2 \qquad (2.4.5)$$

em que k_r é uma constante arbitrária, $\vec{r}_0$ é a posição de equilíbrio e $\vec{r}$ é a posição atual da partícula.

Seguindo adiante na equação 2.4.1 encontra-se o termo que considera triplas de átomos para o potencial. Nele é contemplado o chamado potencial angular, ou seja, dados três átomos ligados em sequência, o ângulo formado entre eles apresenta um potencial de interação que é traduzido pelo potencial de ângulo. Assim como o potencial de ligação química, o potencial angular também possui diversos modelos para descrevê-lo, não obstante o mais comum é também um potencial de harmônico (Bulacu et al., 2013; Brooks et al., 1983). A equação 2.4.6 a seguir denota este modelo:

$$U_\theta = \sum \frac{k_\theta}{2} \left(\theta_{ijk} - \theta_{0,ijk} \right)^2 \qquad (2.4.6)$$

em que k_θ é uma constante arbitrária, θ_0 é o ângulo de equilíbrio e θ o ângulo formado entre as três partículas.

Por fim, apesar da equação 2.4.1 não se limitar apenas até o termo de quadruplas, o último termo comumente considerado é a parcela que leva em conta a interação entre quatro átomos, também chamada de potencial diédrico (Bulacu & van der Giessen, 2013). Trata-se da interação angular que é formada entre esses quatro átomos, como mostra a Figura 9. À imagem dos outros dois potenciais anteriormente apresentados, existem diversos modelos que tentam produzir bons resultados para essa interação.

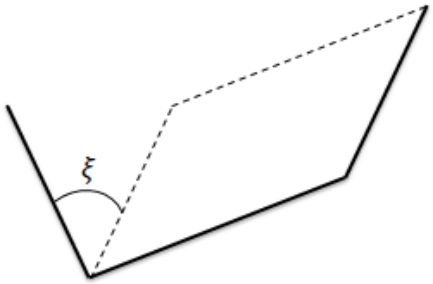

Figura 9 - Ângulo diédrico (**ξ**) indicado por meio da projeção construída na figura. A posição dos átomos é indicada pelos quatro vértices presentes na figura. A posição relativa entre os quatro forma o ângulo diédrico e é considerada para o cálculo do potencial diédrico.

A união de todos estes potenciais até agora apresentados formam o que chamamos de campo de força. Existem diversos exemplos na literatura de campos de força: AMBER (Sorin & Pande, 2005; DePaul et al., 2010), CHARMM (Vanommeslaeghe et al., 2010), Gromos (Oostenbrink et al., 2004), OPLS (Jorgensen et al., 1996) dentre outros. Todos estes citados compartilham uma mesma característica: todos são campos de força atomísticos. Isso significa que estes campos de força foram modelados considerando que cada partícula na simulação corresponde a um átomo.

Contudo, existem limitações operacionais, ou seja, simulações que ultrapassam a capacidade computacional atual, tanto em decorrência do grande número de partículas na simulação quanto por conta da extensão do tempo de simulação. Desse modo, algumas medidas podem ser tomadas para tornar essas simulações possíveis (Blanco et al., 2013; Takada, 2012; Tozzini, 2005).

Conforme apresentado anteriormente, simulações moleculares que envolvem moléculas proteicas são fortes candidatas à utilização dessas medidas, uma vez que envolvem um grande número de partículas. Uma das possíveis medidas é justamente alterar a forma de representação das moléculas na simulação. Essa alteração é, como já apresentado, o chamado campo de

forças *coarse-grained*. Nesses campos de força substituímos a representação atomística, onde cada partícula representa um átomo, pela representação em contas, onde cada partícula representa um grupo de átomos (Clementi, 2007).

Neste aspecto, existem diversos níveis de agrupamento possíveis. O menor nível é o que considera de 4 a 6 contas para representação de um aminoácido. A cadeia lateral é representada por uma conta enquanto os átomos pesados da estrutura principal (*backbone*) são representados cada um por contas individuais (Smith & Hall, 2001; Urban et al., 2004; Nguyen et al., 2004). Avançando em relação à aglutinação de átomos em uma única conta existem os modelos de duas contas. Nestes, a cadeia lateral é representada por uma enquanto a estrutura principal (*backbone*) é representada por outra (Bahar & Jernigan, 1997). E por último existe o modelo da rede elástica (ENM na sigla em inglês) onde a proteína é substituída por um arranjo de contas conectadas por molas e, usualmente, cada conta representa um aminoácido (Kundu et al., 2004; Wu et al., 2003; Burioni et al., 2004).

Por fim, um último questionamento válido para campos de força que utilizam a representação *coarse-grained* é se a capacidade de mimetizar a realidade é mantida. Uma vez que mesmo modelos atomísticos são modelos, ou seja, tentativas de reproduzir a realidade, qual seria o limite para os campos que se utilizam da simplificação *coarse-grained*? Diversos testes e desenvolvimentos já foram realizados, e comparações com resultados experimentais comprovaram que essa modelagem é capaz de representar o comportamento de alguns sistemas mesmo com as diversas simplificações que são consideradas (Clementi, 2007).

Deste modo, a proposta é parametrizar um campo de forças *coarse-grained* nos moldes ENM, ou seja, com apenas uma conta por aminoácido, que seja capaz de representar a estrutura nativa da proteína a partir de sua estrutura primária de forma a permitir o cálculo do fator de forma dessas biomoléculas.

3. MATERIAIS E MÉTODOS

3.1. Desenvolvimento teórico

O objetivo deste trabalho, como dito, é propor um campo de forças para utilização em simulações de dinâmica molecular com o intuito de predizer a estrutura de proteínas. Com o propósito de tornar a experiência da leitura deste desenvolvimento teórico menos árida e mais proveitosa, tomou-se a liberdade de apresentar as ideias de maneira cronológica, da mais simples até a final, considerada mais complexa.

Para entendimento do panorama, deseja-se, por meio de simulações de dinâmica molecular, aplicar um campo de forças, ou seja, um potencial de interação entre partículas, com a finalidade de prever, a partir da sequência de aminoácidos de uma proteína, sua estrutura nativa. E comparar essa estrutura nativa de baixa resolução, por conta do campo de forças desenvolvido ser *coarse-grained*, por meio de curvas de espalhamento de raios X. Para isso, são organizadas configurações iniciais do sistema com a sequência de aminoácidos. Estas configurações iniciais são compostas de diversas partículas e, ao longo do desenvolvimento do campo de forças, a caracterização das partículas alterou-se. No entanto, de modo geral, tem-se a molécula de proteína, caracterizada pela sequência de aminoácidos, partículas *coarse-grained* de água, de modo a representar o solvente, e partículas representando íons. Nas simulações realizadas foram utilizados cátions de sódio ou ânions de cloro. Esses íons atuam como neutralizantes da carga da molécula de proteína na simulação molecular.

Inicialmente estudou-se a escolha entre formular um campo de forças atomístico, ou seja, considerando cada átomo sendo representado por uma partícula, ou um campo de forças *coarse-grained*, onde a decisão a respeito da

aglutinação seria ainda um segundo passo, ou seja, quantos e quais grupos de átomos seriam representados por uma partícula e como esses grupos seriam representados. Diversos autores (Blanco, 2013; Tozzini, 2005; Meller, 2001) afirmam a impossibilidade da capacidade computacional atual de executar simulações moleculares atomísticas quando se tratam de biomoléculas. Não só moléculas proteicas são grandes, mas também não podem ser simuladas isoladas, essas cadeias polipeptídicas estão sempre em solução em condições normais, necessitando uma grande quantidade de moléculas de solvente e, por vezes, de contra íons, uma vez que a carga líquida da simulação deve ser sempre nula em decorrência do algoritmo de cálculo das forças eletrostáticas de Coulomb (Darden et al., 1993). Ademais, o tempo requerido de simulação para que a proteína processe o *folding* tornou a escolha do campo de forças atomístico impraticável. Em decorrência dessas razões, optou-se por seguir o caminho de um campo de forças *coarse-grained*.

Posto isso, passou-se para um segundo momento em que deveria ser tomada a decisão a respeito do grau de aglutinação dos átomos em partículas. Existem diversas formas de realizar a transformação de representação atomística para a representação *coarse-grained* (Tozzini, 2005). Para a primeira idealização teórica, considerou-se que cada aminoácido seria representado por uma partícula e existiriam três tipos de partículas para a representação dos aminoácidos. Esses três tipos seriam distinguidos pelas cargas que apresentariam, a saber: um positivamente carregado, nomeado de HIS como menção ao resíduo histidina, que geralmente é positivamente carregado, um negativamente carregado, nomeado GLU como menção ao resíduo ácido glutâmico, que geralmente é negativamente carregado e um neutro, nomeado MET como menção ao resíduo metionina, que geralmente não possui carga líquida.

Com essa configuração do campo de forças, ainda simplificada, os parâmetros da expressão de Lennard-Jones para as interações de van der Waals foram escolhidos com base na grandeza que permitiria uma melhor predição do fator de forma. Por conta da relação entre o fator de forma e o raio de giro de uma molécula (Svergun & Koch, 2003), esta grandeza foi escolhida para servir de

guia na parametrização das constantes σ e ε que aparecem na expressão 2.4.3. A priori escolheu-se como base para a parametrização a proteína albumina sérica bovina (BSA na sigla em inglês). Essa escolha deveu-se a grande quantidade de dados já medidos e estudos já realizados para esta biomolécula, permitindo a validação do campo de forças com evidências experimentais sólidas.

Ainda no aspecto da interação não ligada, ou seja, que não envolve caracterização de ligações químicas, optou-se por utilizar, além de uma representação *coarse-grained* para a biomolécula, fazer uso desse mesmo tipo de representação molecular para a água que solvata a cadeia polipeptídica. Dessa forma, adotou-se o modelo de água desenvolvido por Shinoda et al. (Shinoda et al., 2006). No modelo desenvolvido por estes autores, cada partícula representa três moléculas de água. É importante pontuar que houve a preocupação em desenvolver um modelo proteico compatível com um modelo de água, visando, assim, aplicações futuras para, por exemplo, o estudo da interação entre duas moléculas de proteína.

Por conta da conclusão dos autores responsáveis pelo modelo de água utilizado, em um primeiro momento, alterou-se a forma da interação de van der Waals no campo de forças desenvolvido. Como foi percebido que a interação de Mie 12-4 representava melhor as propriedades da água, essa forma da expressão foi considerada. A expressão de Mie 12-4 é exibida na equação 3.3.1:

$$U_{Mie} = \frac{3\sqrt{3}}{2}\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{|\vec{r}_{I,j}|}\right)^{12} - \left(\frac{\sigma_{ij}}{|\vec{r}_{I,j}|}\right)^{4}\right] \qquad (3.1.1)$$

Considerando então as cargas, parâmetros da interação eletrostática, e as constantes da expressão de Mie 12-4, calcularam-se as constantes de todas as possíveis interações entre todas as partículas presentes na simulação: HIS, MET, GLU, W (água), NA+ (cátion de sódio) e CL- (ânion cloreto). Os parâmetros dos contra íons considerados para as simulações foram retirados do já consolidado campo de forças atomístico OPLS (Jorgensen et al., 1996).

Apesar da dinâmica de partículas atomísticas ser diferente da dinâmica de partículas *coarse-grained*, existem trabalhos já publicados com simulações mistas (Rzepiela et al., 2011; Shi et al., 2006) e não foram observados desvios nos resultados obtidos em decorrência deste fato, inclusive, por combinar a maior acurácia do modelo atomístico com a rapidez do modelo *coarse-grained* algumas vantagens em relação a velocidade computacional são observadas. Não obstante, a mistura dos dois modelos em simulação deve ser feita com cautela. *O* cálculo dos parâmetros de interação não ligada foram feitos com base nas regras de mistura de Lorentz-Berthelot (Reis et al., 2007) exibidas abaixo nas equações 3.3.2 e 3.3.3:

$$\sigma_{ij}=\frac{\sigma_i+\sigma_j}{2} \tag{3.1.2}$$

$$\varepsilon_{ij}=\sqrt{\varepsilon_i\varepsilon_j} \tag{3.1.3}$$

Deste modo, a partir dos parâmetros para os compostos puros foi possível calcular os parâmetros para as interações entre estes compostos. Os parâmetros das partículas de água foram retirados do trabalho de Shinoda (Shinoda et al., 2006), os parâmetros dos íons presentes na simulação foram retirados do campo de forças atomístico OPLS (Jorgensen et al., 1996) e os parâmetros iniciais dos resíduos de aminoácido foram ajustados para que o raio de giro da albumina sérica bovina correspondesse ao resultado experimental de cerca de 2,9 nm. Os resultados para estes valores são apresentados abaixo na Tabela 1 para cada par presente na simulação:

Tabela 1 - Parâmetros de interação de van der Waals e eletrostáticos para a primeira versão do campo de forças. Os parâmetros da partícula de água *coarse-grained* são do trabalho de Shinoda et al. (Shinoda et al., 2006), os parâmetros dos íons são do campo de força OPLS (Jorgensen et al., 1996) e os parâmetros dos resíduos de aminoácido foram ajustados para que o raio de giro da albumina sérica bovina correspondesse ao experimental de cerca de 2,9 nm.

Partícula i	Partícula j	σ_{ij} (nm)	ε_{ij} (kJ/mol)	q_iq_j (e^2)
W	W	0,4371	3,745	0,0
HIS	HIS	0,5000	2,479	+1,0

GLU	GLU	0,5000	2,479	+1,0
MET	MET	0,5000	2,479	0,0
NA+	NA+	0,3333	0,012	+1,0
CL-	CL-	0,4417	0,493	+1,0
W	HIS	0,4686	3,047	0,0
W	GLU	0,4686	3,047	0,0
W	MET	0,4686	1,523	0,0
W	NA+	0,3852	0,208	0,0
W	CL-	0,4394	1,358	0,0
HIS	GLU	0,5000	2,479	-1,0
HIS	MET	0,5000	2,479	0,0
HIS	NA+	0,4167	0,170	+1,0
HIS	CL-	0,4709	1,105	-1,0
GLU	MET	0,5000	2,479	0,0
GLU	NA+	0,4167	0170	-1,0
GLU	CL-	0,4709	1,105	+1,0
MET	NA+	0,4167	0,170	0,0
MET	CL-	0,4709	1,105	0,0
NA+	CL-	0,3875	0,076	-1,0

Apenas uma modificação é feita em relação à interação entre água e o resíduo MET. Para tentar reproduzir forças hidrofóbicas o parâmetro ε_{ij} é dividido pela metade, diminuindo assim a intensidade da interação entre estas duas partículas na simulação.

As interações ligadas, ou seja, o potencial que representa a ligação química (equação 2.4.5) e o potencial que representa o potencial angular (equação 2.4.6), foram baseadas no trabalho de Angelescu et al. e Voicescu e Ionescu (Angelescu et al., 2007; Voicescu & Ionescu, 2012). Alguns ajustes foram feitos para evitar o colapso da estrutura ao longo da simulação, e as constantes utilizadas aqui são um pouco maiores que as encontradas nos trabalhos citados. Abaixo, na Tabela 2, estão referenciados os valores que são utilizados para os potenciais ligados:

Tabela 2 - Parâmetros para as interações que consideram o potencial de ligação química e o potencial ângular. Na tabela são apresentadas as constantes destes potenciais retiradas do trabalho de Angelescu et al. e Voicescu e Ionescu (Angelescu et al., 2007; Voicescu & Ionescu, 2012).

k_r (kJ.mol^{-1}.nm^{-2})	k_θ (kJ.mol^{-1}.rad^{-2})	r_0 (Å)	θ_0
100000,0	100,0	8,0	180°

Em um segundo momento a ideia foi desvincular, em algum grau, o campo de forças da parametrização em função da BSA. As interações ligadas foram mantidas as mesmas, no entanto, modificou-se a quantidade e a natureza das partículas da proteína. Ao invés de ajustarmos a biomolécula apenas com três tipos de partículas, cada aminoácido agora seria representado por uma partícula específica.

Sendo assim, precisaríamos reparametrizar as constantes que aparecem no potencial de interação de van der Waals e rever a distribuição de cargas para a interação eletrostática. No entanto, como desejamos formular um campo de forças que seja o mais independente possível, ao invés de agora utilizar uma proteína e seu raio de giro para deduzir as constantes, partiu-se no sentido de interpretar fisicamente as constantes de Lennard-Jones. Como já mencionamos, a constante σ atua como um parâmetro dimensional da partícula, podendo, por exemplo, ser interpretado como o diâmetro da partícula, considerando, claro, a extensão de sua nuvem eletrônica. Desse modo, a partir dos volumes moleculares de cada aminoácido (Richards, 1977) e,

considerando que cada um deles comporta-se como uma esfera, calculou-se o diâmetro para todos eles. O diâmetro calculado foi então assumido como o valor da constante σ para cada aminoácido e foi apresentado na Tabela 3.

Tabela 3 – Volume molecular e constante de Lennard-Jones dos resíduos de amino ácido utilizados nas funções que caracterizam o campo de forças. Os resíduos são assumidos como esferas e o parâmetro $\boldsymbol{\sigma}$ de Lennard-Jones é então o diâmetro da esfera.

Amino ácido i	Volume i (nm^3)	σ_i (nm)
Alanina	0,092	0,560
Arginina	0,225	0,755
Asparagina	0,135	0,636
Ácido Aspártico	0,125	0,620
Cisteína	0,106	0,587
Ácido Glutâmico	0,155	0,666
Glutamina	0,161	0,675
Glicina	0,066	0,501
Histidina	0,167	0,683
Isoleucina	0,169	0,686
Leucina	0,168	0,685
Lisina	0,171	0,689
Metionina	0,171	0,689
Fenilalanina	0,203	0,729
Prolina	0,129	0,627
Serina	0,099	0,574
Treonina	0,122	0,615

Triptofano	0,240	0,771
Tirosina	0,203	0,729
Valina	0,142	0,647

Já para a constante ε, é delicado encontrar uma interpretação física. É possível inferir que ela está intimamente ligada à intensidade da interação, já que é o valor do poço energético de Lennard-Jones. No entanto, por se tratar de um conceito abstrato, manteve-se o valor inicialmente considerado para todos os aminoácidos presentes na proteína, ou seja, o valor de 2,479 kJ/mol. Neste segundo caso, nada foi considerado em relação à hidrofobicidade dos resíduos.

Finalmente, para a consideração a respeito das cargas de cada resíduo, foi estudada a relação entre o pK_a de cada seção do amino ácido em relação ao pH da simulação. De acordo com os valores de pK_a, apresentados na Tabela 4, é possível concluir se existe ou não protonação dos terminais e/ou das cadeias laterais de cada amino ácido a partir da equação de Henderson-Hasselbach. Considerando a reação de desprotonação de um ácido, tem-se o seguinte equilíbrio:

$$\text{Ácido} + H_2O \rightleftharpoons \text{Base Conjugada} + H_3O^+$$

Pode-se então escrever a equação para a constante de equilíbrio:

$$K_A = \frac{[\text{Base Conjugada}][H_3O^+]}{[\text{Ácido}]} \qquad (3.1.4)$$

Considerando a reação de protonação de uma base, tem-se o seguinte equilíbrio:

$$\text{Base} + H_2O \rightleftharpoons \text{Ácido Conjugado} + OH^-$$

Pode-se então, novamente, escrever a equação para a constante de equilíbrio:

$$K_B = \frac{[\text{Ácido Conjugado}][OH^-]}{[\text{Base}]} \quad (3.1.5)$$

É possível concluir a partir das expressões 3.3.4 e 3.3.5 que:

$$K_A K_B = [H_3O^+][OH^-] \quad (3.1.6)$$

Tomando o cologaritmo da expressão 3.3.6 e aplicando a propriedade da função logarítmica onde o logaritmo do produto é a soma dos logaritmos, obtém-se:

$$pK_A + pK_B = pH + pOH \quad (3.1.7)$$

Retomando as expressões 3.3.4 e 3.3.5 pode-se calcular o cologaritmo das constantes de equilíbrio:

$$pK_A = -\log(H^+) - \log\left(\frac{[\text{Base Conjugada}]}{[\text{Ácido}]}\right) = pH - \log\left(\frac{[\text{Molécula Desprotonada}]}{[\text{Molécula Protonada}]}\right) \quad (3.1.8)$$

$$pK_B = -\log(OH^-) - \log\left(\frac{[\text{Ácido Conjugado}]}{[\text{Base}]}\right) = pOH - \log\left(\frac{[\text{Molécula Protonada}]}{[\text{Molécula Desprotonada}]}\right) \quad (3.1.9)$$

Assumindo a hipótese de que os equilíbrios estabelecem-se independentemente uns dos outros, ou seja, não se afetam entre si, define-se a razão dentro do logaritmo das expressões 3.3.8 e 3.3.9 a partir das concentrações percentuais. Além disso, para que seja utilizada apenas uma expressão, as expressões 3.3.7, 3.3.8 e 3.3.9 são combinadas e o parâmetro ν_j passa a compor a expressão assumindo o valor 1 para bases e -1 para ácidos. Dessa forma chega-se a expressão de Henderson-Hasselbach (Po & Senozan, 2001):

$$pK_{A,j} = pH + v_j \log\left(\frac{\alpha_j}{1-\alpha_j}\right) \quad (3.1.10)$$

em que α_j é a fração de grupos ionizados na molécula j.

Considerou-se, para este desenvolvimento, que a carga parcial de todos os aminoácidos do tipo j dentro da molécula de proteína é numericamente igual a concentração percentual de grupos ionizados deste aminoácido, portanto, o valor α_j.

Com essa análise era então atribuída carga a cada partícula de resíduo de aminoácido dentro da molécula de proteína. O valor da carga era calculado a partir da equação 3.3.10 e, caso fosse, em módulo, inferior a 0,1, era considerada nula. Caso fosse, em módulo, superior a 0,9, era considerada unitária. Estes valores foram atribuidos após alguns testes de sensibilidade para essa distribuição de cargas na molécula. Uma normalização é feita após a atribuição das cargas para que a molécula possua um valor de carga inteiro.

Seria possível calcular, iterativamente, a carga de cada resíduo com base nas alterações estruturais da molécula ao longo da simulação de dinâmica molecular. Existem ferramentas que levam em consideração a posição do resíduo dentro da molécula de proteína para calcular as variações de pK_A que ocorrem nesses aminoácidos devido a posição que assumem dentro da estrutura proteica (Dolinsky et al., 2004; Dolinsky et al., 2007). No entanto, para determinar essas alterações nos valores de pK_A são assumidas hipóteses, como a de que a carga se comporta como uma função contínua para a utilização da equação de Poisson-Boltzmann, que acarretam erros na ordem de 60% para a variação do pK_A (Davies et al., 2006). Por conta dos erros altos incorridos no uso dessas ferramentas, optou-se por utilizar a abordagem simples e prática exibida anteriormente.

Os valores de pH considerados nas simulações eram condizentes com os dados experimentais que seriam usados para a validação do campo e os valores de pK_a foram retirados do trabalho de Kenndler et al. (Kenndler et al., 2004).

Tabela 4 - Valores de pK_a para os terminais carboxila e amina dos amino ácidos e para as cadeias laterais, quando ionizáveis que foram utilizados com a equação de Hendersson-Hasselbach no cálculo da carga de cada resíduo na molécula de proteína.

Amino ácido	pK_a - Carboxila	pK_a - Amina	pK_a – Cadeia Lateral
Alanina	2,33	9,71	-
Arginina	2,03	9,00	12,10
Asparagina	2,16	8,76	-
Ácido Aspártico	1,95	9,66	3,71
Cisteína	1,91	10,28	8,14
Ácido Glutâmico	2,16	9,58	4,15
Glutamina	2,18	9,00	-
Glicina	2,34	9,58	-
Histidina	1,70	9,09	6,04
Isoleucina	2,26	9,60	-
Leucina	2,32	9,58	-
Lisina	2,15	9,16	10,67
Metionina	2,16	9,08	-
Fenilalanina	2,18	9,09	-
Prolina	1,95	10,47	-
Serina	2,13	9,05	-
Treonina	2,20	8,96	-
Triptofano	2,38	9,34	-
Tirosina	2,24	9,04	10,10
Valina	2,27	9,52	-

Ao testarmos essa nova configuração, obtivemos resultados inesperados que serão logo apresentados na próxima seção. Não obstante, estes resultados motivaram a formulação de algum parâmetro que, ao contrário do proposto anteriormente, considerava alguma alteração para levar em conta o fator hidrofóbico de cada resíduo. Na busca de algum modelo que mimetizasse as forças hidrofóbicas surgiu a ideia de propormos um modelo próprio que consideraria como parâmetro principal a hidropatia de cada amino ácido (Kyte & Doolittle, 1982). A hidropatia, apresentada na Tabela 5, funciona como um índice que traduz a hidrofobicidade de um amino ácido, a saber: quanto maior for a hidropatia de um amino ácido, mais hidrofóbico ele é.

Tabela 5 - Hidropatia dos amino ácidos baseadas na escala construída por Kyte e Doolittle (Kyte & Doolittle, 1982). As hidropatias são consideradas nos modelos a fim de mimetizar as forças hidrofóbicas. Um parâmetro foi adicionado na interação de van der Waals com base na hidropatia de cada aminoácido.

Amino ácido	Hidropatia
Alanina	1,80
Arginina	-4,50
Asparagina	-3,50
Ácido Aspártico	-3,50
Cisteína	2,50
Ácido Glutâmico	-3,50
Glutamina	-3,50
Glicina	-0,40
Histidina	-3,20
Isoleucina	4,50
Leucina	3,80
Lisina	-3,90

Metionina	1,90
Fenilalanina	2,80
Prolina	-1,60
Serina	-0,80
Treonina	-0,70
Triptofano	-0,90
Tirosina	-1,30
Valina	4,20

Com o objetivo então de alterar a intensidade da interação de um amino ácido com a água, formulou-se um parâmetro que interferiria no valor da constante ε_{ij} quando i ou j fossem a água. Estabeleceu-se que a constante ε_{ij} seria dividida pela metade nos casos em que a hidropatia fosse menor que 1 e não seria alterada nos outros casos. A escolha por dividir pela metade é arbitrária. Outros valores e modelos foram testados, no entanto, este foi o que apresentou melhores resultados. Com base nos valores de hidropatia então foi proposto o seguinte modelo:

$$H_{ij}(h_i)=\begin{cases}1, \text{ se } h_i\leq 1\\ 2, \text{ se } h_i>1\end{cases} \qquad (3.1.11)$$

em que H_{ij} é o fator resultante e h_i é a hidropatia do amino ácido em questão na interação considerada. Dessa forma, a expressão de Lennard-Jones que deveria ser considerada para o cálculo das forças de van der Waals precisa de alterações também. Além da incorporação do fator hidrofóbico H_{ij} proposto acima, aproveitou-se para que outro refinamento tomasse forma. Como os autores que propuseram o modelo de água que estamos utilizando consideraram que as interações com a água deveriam ser da forma 12-4, na última versão do campo de forças aqui proposto apenas as interações com a

água seguem a forma 12-4. Todas as outras interações seguem a forma de interação 12-6, que é a mais clássica em dinâmica molecular. É interessante notar que o potencial de interação intermolecular pode ser obtido a partir da mecânica quântica; no entanto, este é um problema extremamente complicado do ponto de vista numérico. Apesar disso, é possível mostrar, a partir da teoria de perturbações, que o potencial intermolecular tende assintoticamente para uma função inversamente proporcional a sexta potência da distância entre partículas, como considera o potencial de interação de Lennard – Jones. Portanto, temos então as seguintes equações modelando as interações de van der Waals:

$$U_{LJ}=4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{|\vec{r}_{I,j}|}\right)^{12}-\left(\frac{\sigma_{ij}}{|\vec{r}_{I,j}|}\right)^{6}\right] \tag{3.1.12}$$

$$U_{Mie}=\frac{3\sqrt{3}}{2}\left(\frac{\varepsilon_{ij}}{H_{ij}}\right)\left[\left(\frac{\sigma_{ij}}{|\vec{r}_{I,j}|}\right)^{12}-\left(\frac{\sigma_{ij}}{|\vec{r}_{I,j}|}\right)^{4}\right] \tag{3.1.13}$$

Todos os outros parâmetros e modelos considerados para o campo mantiveram-se como na proposta inicial.

Por fim, interações específicas para a formação das ligações de hidrogênio e pontes dissulfeto não foram consideradas. Apesar dessas ligações serem muito importantes para a conformação final de proteínas, um dos objetivos era manter o campo de forças simples, como as interações consideradas foram capazes de prever estruturas que produziram intensidades de espalhamento de raios X condizentes com as produzidas pela estrutura cristalográfica ou com dados experimentais, as interações para ligações de hidrogênio e pontes dissulfeto não foram consideradas.

Ainda neste tópico, é importante ressaltar que, como o objetivo é prever a estrutura das proteínas por meio de simulação molecular e utilizar curvas de intensidade de espalhamento de raios X para a verificação desta previsão, a baixa resolução de um campo de forças *coarse-grained* deve bastar para esta tarefa, uma vez que a curva de intensidade de espalhamento fornece informações de baixa resolução quando tratamos de soluções. Por se tratar de

uma solução, não existe ordem entre as moléculas, por isso, os resultados do experimento de espalhamento são de baixa resolução, uma vez que os dados são dependentes das orientações e configurações das moléculas em solução e também por conta das operações matemáticas necessárias para que tenhamos acesso a função p(r) perde-se ainda mais em resolução. Deste modo, comparar as estruturas *coarse-grained* por meio de curvas de espalhamento é adequado (Yang et al., 2011).

3.2. Método para as simulações

As simulações de dinâmica molecular foram executadas no *software* GROMACS (Berendsen et al., 1995) versão 5.0.7. Além deste *software* foram escritos quatro códigos de programação principais para a organização do campo de forças proposto. As proteínas escolhidas para o teste do campo de forças foram a lisozima e a albumina sérica bovina, uma vez que ambas já são muito conhecidas e possuem diversos estudos na literatura.

O primeiro código, GENINP, gerava, a partir do arquivo estrutural de uma proteína (PDB – *protein data bank*), um arquivo apenas com a sequência de aminoácidos para que a estrutura *coarse-grained* proposta pelo campo de forças desenvolvido pudesse começar a tomar forma. Este arquivo gerado foi considerado um arquivo *input* para que os outros fossem escritos. Um exemplo de arquivo *input* que foi gerado para a lisozima é apresentado na Figura 10 seguinte.

```
TITLE  Input file generated by GENINP.
   129  9.01  4.48
LYS VAL PHE GLY ARG CYS GLU LEU ALA ALA ALA MET LYS ARG HIS
GLY LEU ASP ASN TYR ARG GLY TYR SER LEU GLY ASN TRP VAL CYS
ALA ALA LYS PHE GLU SER ASN PHE ASN THR GLN ALA THR ASN ARG
ASN THR ASP GLY SER THR ASP TYR GLY ILE LEU GLN ILE ASN SER
ARG TRP TRP CYS ASN ASP GLY ARG THR PRO GLY SER ARG ASN LEU
CYS ASN ILE PRO CYS SER ALA LEU LEU SER SER ASP ILE THR ALA
SER VAL ASN CYS ALA LYS LYS ILE VAL SER ASP GLY ASN GLY MET
ASN ALA TRP VAL ALA TRP ARG ASN ARG CYS LYS GLY THR ASP VAL
GLN ALA TRP ILE ARG GLY CYS ARG LEU
```

Figura 10 - Exemplo de arquivo *input* gerado pelo código GENINP. Na figura observa-se o título gerado pelo código na primeira linha, a quantidade de resíduos de aminoácido, o tamanho da aresta da caixa que será utilizada na simulação e o pH que será considerado na simulação. Abaixo da segunda linha a sequência de aminoácidos de acordo com a abreviação em três letras de cada um.

A partir do exemplo conseguimos identificar que três informações são impressas na segunda linha, respectivamente: o número total de resíduos da biomolécula, o lado de uma caixa de simulação cúbica e o pH da simulação. O número de resíduos é calculado por contagem simples, e o tamanho da caixa é calculado por meio de uma função empírica construída a partir da experiência nas simulações feitas. É importante notar aqui que lidamos com um problema de balanceamento, a caixa não pode ser muito grande, pois ao encolher para que o sistema assuma a densidade correta, a simulação pode incorrer em desestabilidades numéricas e parar, mas também não pode ser pequena demais a ponto das partículas que queremos não caberem no volume que ela delimita. Dessa forma, testes foram realizados e chegou-se, de maneira empírica, à seguinte função para o cálculo do tamanho da caixa:

$$C=0.0088NR+4.8634 \qquad (3.2.1)$$

em que C é a aresta do volume cúbico e NR é o número de resíduos de aminoácido presentes na molécula proteica.

O pH é um valor de entrada para o código. Este valor é utilizado por outros códigos para que consigam calcular a carga de cada resíduo na cadeia polipeptídica.

Seguindo na cadeia computacional, o arquivo de *input* gerado serve de entrada para o código CF_LRP, que é o código núcleo do campo de forças. Este código é responsável por gerar todos os arquivos que serão usados pelo *software* de simulação para calcular as interações entre todas as partículas presentes na simulação. Como saída, o CF_LRP gera os arquivos de topologia e de tabelas para a diferenciação entre as interações 12-4 e 12-6 mencionadas anteriormente. Além disso, ele também escreve os arquivos que contém as instruções básicas para a simulação, tanto para a primeira etapa de minimização de energia quanto para a segunda etapa de produção no *ensemble* NPT, estas etapas serão exploradas, explicadas e justificadas no final desta seção.

Partindo para o terceiro código, o STR, vamos agora tratar de organizar o arquivo de configuração inicial para a simulação. Além dos arquivos de topologia, que contém as informações a respeito das interações entre as partículas, e os arquivos de instruções, que contêm a receita para a simulação, devemos também fornecer um arquivo com uma configuração inicial das partículas dentro da caixa de simulação. Por se tratar de simulação molecular clássica, ou seja, apoiada na teoria Newtoniana da mecânica, é necessário que entreguemos ao simulador uma estrutura inicial, ou seja, as posições iniciais de cada partícula, e a velocidade inicial de cada partícula. Este terceiro código trata, então, de gerar esse arquivo com as posições iniciais de cada partícula, já que as velocidades estão já colocadas nos arquivos de instruções gerados pelo CF_LRP. As velocidades iniciais de cada partícula serão geradas por meio de uma distribuição de Maxwell-Boltzmann calculada pelo *software* de simulação GROMACS. A distribuição é dependente da temperatura, que é colocada como 300 K, já que as simulações são realizadas a 300 K e da massa de cada partícula, que é dada nos arquivos gerados pelo CF_LRP. A distribuição de Maxwell-Boltzmann é caracterizada pela seguinte expressão (Zhigilei & Garrison, 1997):

$$\Delta(u)=\sqrt{\left(\frac{m}{2\pi kT}\right)^3}\,4\pi u^2 e^{-\frac{mu^2}{2kT}} \qquad (3.2.2)$$

em que m é a massa, k é a constante de Boltzmann, T é a temperatura e u a velocidade.

Por estarmos tratando de um modelo que pretende estudar o processo de *folding* de uma proteína, a estrutura inicial pode apresentar alguma influência. Para que essa influência possa ser mapeada, foram propostas três estruturas iniciais. Essas três foram desenvolvidas para que coubessem na caixa inicial e para que a caixa inicial não tomasse um tamanho muito grande. Por tamanho muito grande queremos dizer um volume que não permitisse uma densidade ao menos próxima da densidade da água. Isso se justifica pelo fato da instabilidade que a simulação molecular apresenta quanto ao encolhimento brusco da caixa. Para evitar que a simulação não consiga proceder por conta deste encolhimento brusco, foram então desenvolvidas três estruturas que estivessem de acordo com uma densidade inicial próxima de 1000 kg/m³.

As três estruturas propostas, exibidas na Figura 11, são as estruturas helicoidal, intestino e zig-zag. Os nomes foram escolhidos com base nos formatos propostos.

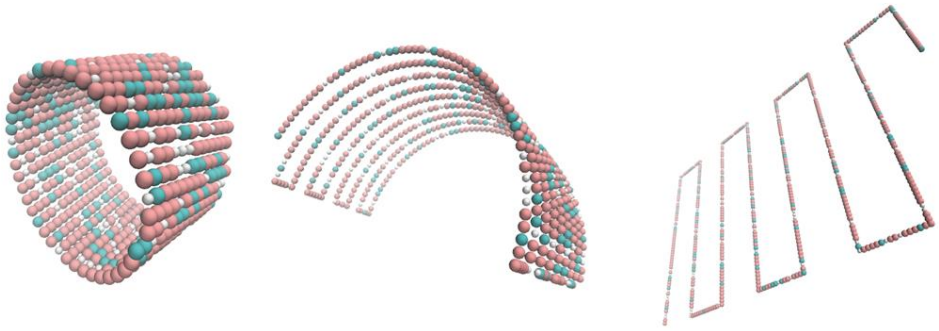

Figura 11 - Estruturas iniciais da molécula de proteína propostas para as simulações de dinâmica molecular. Da esquerda para a direita: helicoidal, intestino e zig-zag. As estruturas propostas não tem relação física com a estrutura primária da proteína, são estruturas propostas com o objetivo de caberem na caixa de simulação sem que a caixa seja grande demais a ponto de encolher bruscamente no início da simulação e causar uma parada no algoritmo de dinâmica molecular.

Ademais, depois de gerar as estruturas iniciais, o STR solvata e equilibra eletricamente as caixas de simulação adicionando água e contra íons respectivamente.

Com estes três códigos conclui-se, então, a preparação dos arquivos necessários para a simulação molecular. Um fluxograma do procedimento é exibido na Figura 12. Retomemos, portanto, as duas fases já citadas que são executadas na simulação propriamente dita que ocorre no *software* GROMACS.

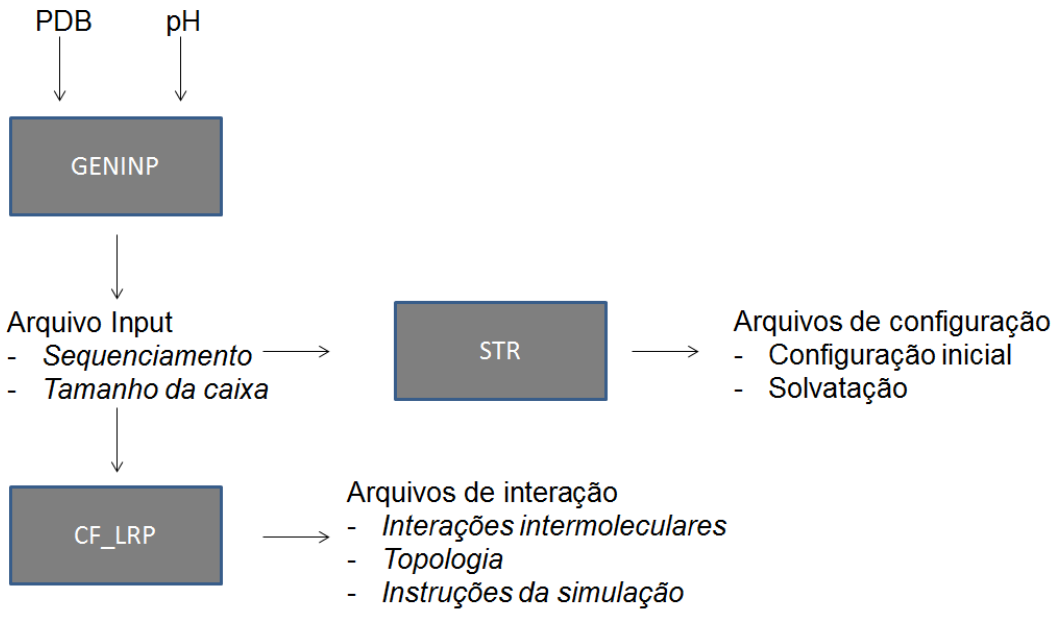

Figura 12 - Fluxograma representando a sequência de códigos de programação utilizados para a preparação dos arquivos necessários para a simulação de dinâmica molecular. No fluxograma são apresentadas as informações requeridas pelos códigos e as informações fornecidas pro eles.

A primeira etapa para a simulação é a minimização de energia. Por vezes a configuração inicial possui uma energia potencial grande, o que significa que algumas partículas podem estar submetidas a forças muito intensas. Se a simulação partir de uma configuração inicial com essa característica, é possível que a caixa da simulação “exploda”, ou seja, que o simulador perca de vista algumas partículas por conta do deslocamento muito brusco a que serão

submetidas na primeira iteração. Isto posto, realiza-se a minimização de energia para que este tipo de problema seja evitado.

A segunda etapa para a simulação é a chamada etapa de produção. Nesta é selecionado um *ensemble* e então o algoritmo de integração da equação de Newton, *leapfrog* (Allen & Tildesley, 2009), atualiza posições e velocidades de cada partícula dentro da caixa. Com o objetivo de mimetizar as condições de laboratório, o *ensemble* escolhido para a etapa de produção foi o NPT, ou seja, o número de partículas, a pressão e a temperatura são mantidos constantes ao longo da simulação. O número de partículas é mantido constante a partir das condições periódicas de contorno, já a pressão é mantida constante a partir de um algoritmo de barostato, no caso específico, o barostato de Berendsen (Berendsen et al., 1984). Para a temperatura ser mantida utiliza-se um algoritmo de termostato, no caso, o termostato de Berendsen modificado (Berendsen et al., 1984; Parrinello et al., 2007). Apesar desses algoritmos para a manutenção da temperatura e da pressão serem antigos e já terem sido substituídos por algoritmos mais eficientes, a utilização deles é devida a um motivo específico do simulador utilizado e diz respeito às interações intermoleculares que são utilizadas nas simulações. Para que o GROMACS faça uso de diferentes interações e para que consigamos utilizar o potencial de Mie 12-4, é necessário que se utilize a opção *group* do simulador, permitindo que as interações desejadas sejam inseridas manualmente, uma vez que o simulador não possui essas interações como padrão. Para isso apenas o barostato e termostato de Berendsen estão disponíveis. É importante ressaltar que o uso de ambos produziu bons resultados quando a intensidade de espalhamento de raio X foi calculada.

A etapa de produção é realizada considerando um tempo de 10,0 nanossegundos, um passo de 2,0 fentossegundos totalizando 5000000 de passos para o cálculo. A pressão é mantida em 1,0 bar e a temperatura em 300 K. As velocidades iniciais são calculadas a partir de uma distribuição de Maxwell-Boltzmann considerando a temperatura de 300 K.

Com a conclusão da simulação, é necessário que, inicialmente, verifique-se como evoluiu a densidade da caixa. Por estarmos no *ensemble* NPT, o volume é um grau de liberdade existente durante a simulação, portanto a verificação do volume específico, ou o inverso da densidade, é importante para verificarmos se atingimos a densidade da água. Como o sistema está bastante diluído, para garantirmos que o *ensemble* tenha conseguido reproduzir um sistema real, devemos ter conseguido atingir um valor de densidade próximo a 1000 kg/m³. Em seguida entra em ação o último código escrito, o TRAJ.

Este último código trata e avalia os dados que retiramos da simulação. A partir deste tratamento conseguimos calcular e verificar se o fator de forma da proteína é condizente com os dados experimentais. A partir desta comparação conseguimos fazer ajustes e/ou verificar inconsistências na metodologia.

O TRAJ ignora os 1,5 nanossegundos iniciais da produção, que são considerados como tempo de equilibração para a pressão, para a temperatura e para densidade. Esse valor de 1500 picossegundos foi avaliado depois da realização das simulações e era checado a partir da estabilidade da densidade, da estabilidade da temperatura e da estabilidade do raio de giro. Como as simulações são feitas no *ensemble* NPT, o grau de liberdade dentre as variáveis PVT é o volume. Com a variação do volume existe a variação da densidade e, portanto, com a estabilidade dela e da temperatura podemos considerar que o equilíbrio foi atingido. Para verificar se o processo de *folding* havia chegado ao equilíbrio dinâmico da estrutura verificou-se a estabilidade do raio de giro. Por se tratar de simulação de dinâmica molecular, diz-se equilíbrio dinâmico da estrutura porque a biomolécula não atingirá uma única forma final, mas oscilará entre diversas estruturas. Este valor, 1,5 nanossegundos, foi estipulado depois de observar algumas simulações e verificar que era o suficiente para a estabilização da densidade, da temperatura e do raio de giro. Com os últimos 8500 picossegundos é realizada a análise dos dados produzidos pela simulação. A trajetória é quebrada de um em um picossegundo e cada um deles é analisado. Ao final, é feita a média dessas 8500 análises e então essa média é comparada com os dados experimentais e com os dados calculados a partir das estruturas cristalográficas das proteínas.

As simulações feitas a partir do método CABS foram realizadas a partir de um servidor do grupo de pesquisa responsável por esse método. O servidor fica disponibilizado na *internet* e pode ser acessado a partir do seguinte endereço: <http://biocomp.chem.uw.edu.pl/CABSfold/>. Para utilizá-lo, deve-se selecionar a opção *de novo*, para que a previsão seja feita apenas com base na sequência de amino ácidos, e adicionar a sequência de amino ácidos segundo o código padrão de uma letra para amino ácidos. Esse é um código padrão, além do código de uma letra, existe também o código de três letras, ambos estipulados pela IUPAC (*International Union for Pure and Applied Chemistry*).

Com isso, a submissão da simulação é realizada e os resultados da simulação são disponibilizados em aproximadamente quatro dias. As simulações do método CABS são simulações Monte Carlo e, a partir da sequência de amino ácidos e de maneira estocástica, minimizam a energia da estrutura da biomolécula de proteína a fim de encontrar a estrutura nativa a partir dessa minimização. Os resultados obtidos a partir do método CABS foram utilizados para comparação com os resutlados obtidos pelo método desenvolvido neste trabalho.

Concluída a apresentação do desenvolvimento teórico, passamos então para a apresentação dos resultados.

4. RESULTADOS

Antes de apresentar propriamente os resultados obtidos, relatemos alguns pontos importantes ao longo do desenvolvimento deste trabalho para que fique mais claro o porquê da apresentação de cada resultado.

O campo de forças proposto começou com uma ideia simples e, de algum modo, rudimentar, envolvendo três tipos de partículas para a composição da proteína, divergindo apenas na carga elétrica de cada uma. Com a intenção de tentar reproduzir o efeito das interações hidrofóbicas/hidrofílicas, dividiu-se o parâmetro energético do potencial de interação de Mie pela metade no resíduo hidrofóbico, quando a interação se dava com a água.

Como o objetivo era tornar o cálculo da intensidade de espalhamento de raio X mais fenomenológico, com bases físicas e não heurísticas, tornamos o campo um pouco mais refinado ao criar uma partícula para cada aminoácido. Aqui, bases físicas significam o uso de um campo de forças que advém da física para a previsão da estrutura da proteína, ao invés da parametrização a partir de uma estrutura já conhecida ou do processo de tentativa e erro citado anteriormente. No entanto, no primeiro teste não adicionamos nenhuma modificação aos parâmetros energéticos do potencial de interação de Mie, mantivemos todos os valores idênticos para todos os resíduos. Essa escolha gerou um resultado esperado, mas ainda assim interessante para ser exibido aqui.

4.1. Hidrofobicidade

Ao visualizarmos a forma ao final da simulação molecular da lisozima, proteína utilizada nesta simulação, verificou-se que a estrutura atingida não era globular, como sabe-se ser a estrutura nativa desta proteína. Ao contrário, a biomolécula assumiu a forma de um laço, como exibido na Figura 13 abaixo.

Figura 13 - Estrutura ao final da simulação molecular da lisozima sem levar em consideração a modificação para as forças hidrofóbicas indicada na equação 3.1.13. O fator H considerado para as forças hidrofóbicas reduz o parâmetro energético do potencial de Mie, deste modo diminuindo a intensidade da interação entre os resíduos hidrofóbicos e a água.

A forma assumida não era esperada, uma vez que a lisozima é uma proteína globular, ou seja, possui a estrutura nativa semelhante a uma esfera ou um elipsoide de revolução.

Inicialmente tentou-se investigar se o problema estaria no modelo utilizado para a água. Foram alterados diversos parâmetros da simulação, como o raio de *cut-off*, para verificar se estes parâmetros exerceriam alguma influência neste resultado. Nenhuma dessas alterações produziu resultados significativos na estrutura, e a cadeia polipeptídica sempre finalizava a simulação em uma forma de laço similar a apresentada anteriormente.

Em um segundo momento, decidiu-se por modificar o campo de forças para levar em conta as interações devidas a hidrofobicidade de cada resíduo, que foi negligenciada nessa versão inicial do campo de forças. Com isso, foi proposto o modelo apresentado na seção anterior, no qual a hidropatia é utilizada para calcular o fator H que, por sua vez, altera o parâmetro energético do potencial de interação de Mie quando a interação for calculada entre a água e um resíduo de amino ácido.

Ao testarmos essa versão, a lisozima apresentou a seguinte estrutura ao final da simulação, apresentada na Figura 14.

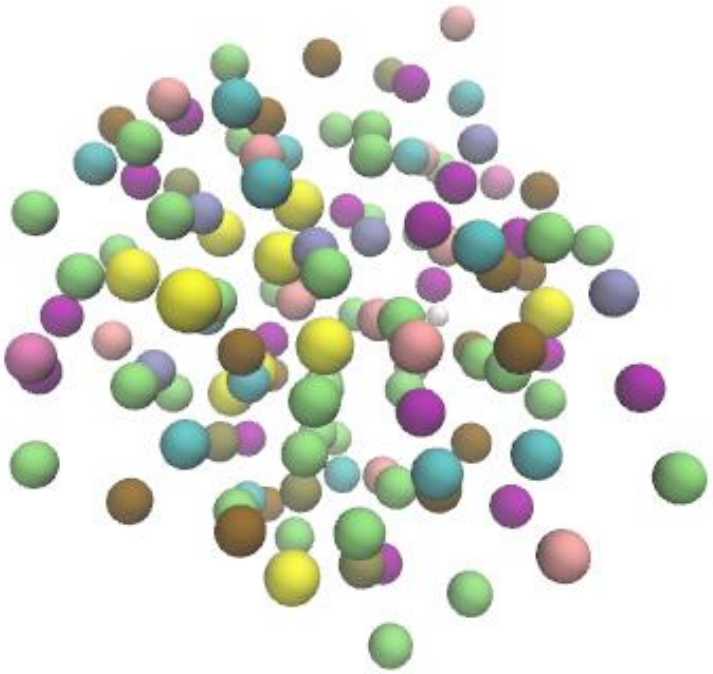

Figura 14 - Estrutura ao final da simulação molecular da lisozima levando em consideração a modificação para as forças hidrofóbicas indicada na equação 3.1.13. O fator H considerado para as forças hidrofóbicas reduz o parâmetro energético do potencial de Mie, deste modo diminuindo a intensidade da interação entre os resíduos hidrofóbicos e a água.

Verifica-se que a estrutura passou a apresentar a forma globular esperada para a estrutura nativa dessa biomolécula.

Conseguiu-se mostrar que, ao adicionarmos uma modificação na interação que os resíduos têm com a água, a forma globular da proteína foi assumida, ou seja, que a interação com a água é um fator importante no processo, como já é sabido. E, diferentemente das ligações de hidrogênio e pontes dissulfeto, não pode ser negligenciada, mesmo em um método de baixa resolução como o proposto neste trabalho.

Com o campo de forças permitindo atingir a estrutura globular, avançamos então para o cálculo da intensidade de espalhamento de raios X da lisozima e da albumina soro bovina. A escolha destas duas é justificada por conta da

grande quantidade dados disponíveis para ambas na literatura, permitindo comparações e avaliações do campo de forças proposto.

4.2. Influência da estrutura inicial

Em decorrência da complexidade do processo de *folding* de uma proteína, dados os controles cinético e termodinâmico concomitantes, é esperado que existam influências da estrutura inicial da biomolécula na simulação molecular no processo de dobramento da estrutura e, consequentemente, na estrutura atingida ao final da simulação de dinâmica molecular.

A primeira verificação feita foi a checagem da densidade atingida ao final da simulação em todas as simulações. Como o *ensemble* escolhido foi o NPT (número de partículas, pressão e temperatura constantes) a variável canônica que se torna dependente é justamente o volume, portanto é necessário avaliar se a densidade, inverso do volume específico, atingiu o valor esperado.

Foram realizados testes com as proteínas lisozima e albumina sérica bovina (BSA). Ambas foram simuladas partindo das três estruturas iniciais propostas. As simulações foram realizadas considerando as interações entre partículas descritas anteriormente pelo campo de forças *coarse-grained*. A duração total das simulações foi de 10,0 nanossegundos com passos de 2,0 picossegundos no *ensemble* NPT. A temperatura foi mantida em 300 K e a pressão foi mantida a 1,0 bar. Os resultados para a densidade das simulações moleculares com lisozima são apresentados na Figura 15.

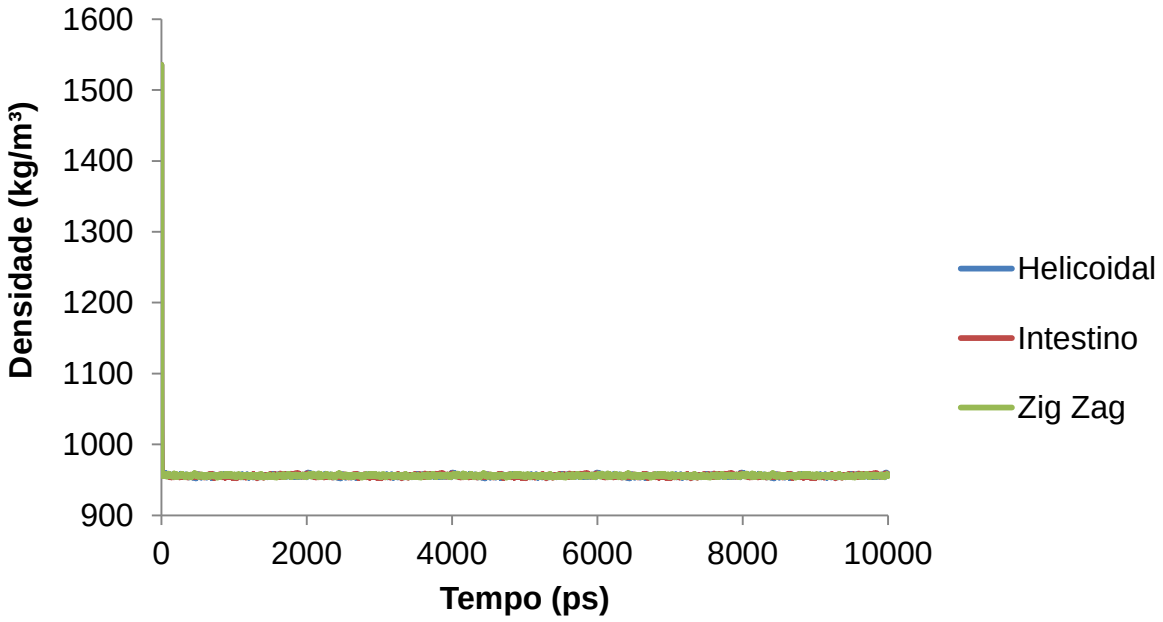

Figura 15 – Densidade da caixa de simula ção ao longo da simulação de dinâmica molecular para a lisozima. As simulações foram feitas durante 10,0 nanossegundos com passos de 2,0 fentossegundos no *ensemble* NPT a temperatura de 300 K e press ão de 1,0 bar.

Para avaliar o regime transiente da simulação, ou seja, os primeiros picossegundos de equilibração, é apresentado o gráfico para a densidade no início da simulação na Figura 16:

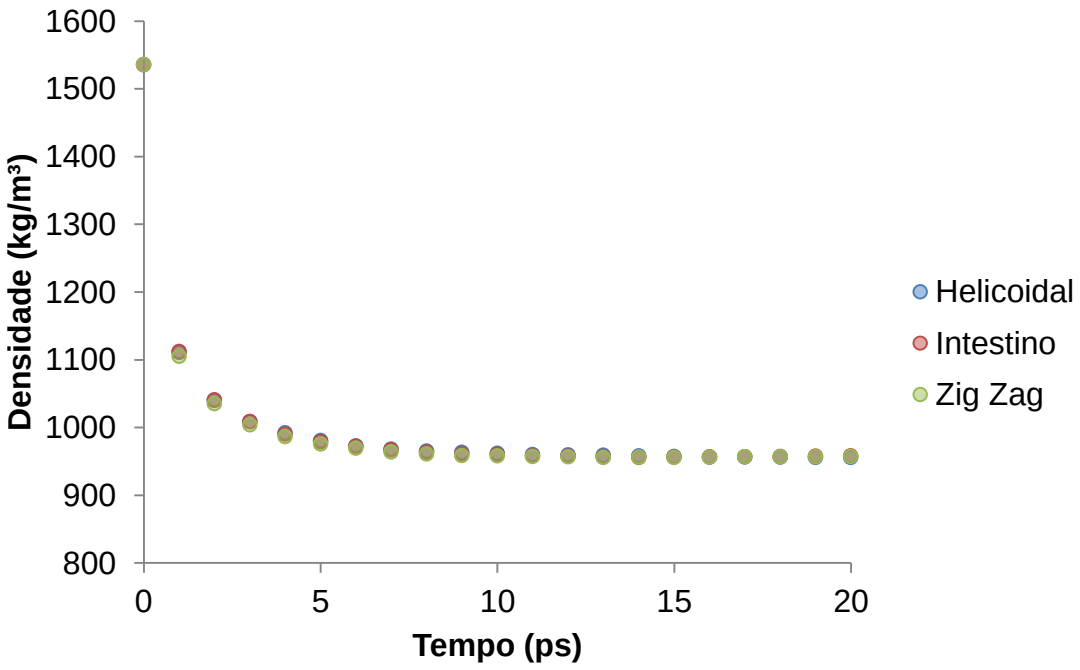

Figura 16 - Evolu ção das densidades para as simulações de dinâmica molecular com a lisozima. São apresentados os primeiros picossegundos da simulação para avaliação do período de equilibração da simula ção. As simulações partindo das três estruturas iniciais propostas são exibidas.

Os resultados para as densidades das simulações moleculares com albumina s érica bovina são apresentados na Figura 17:

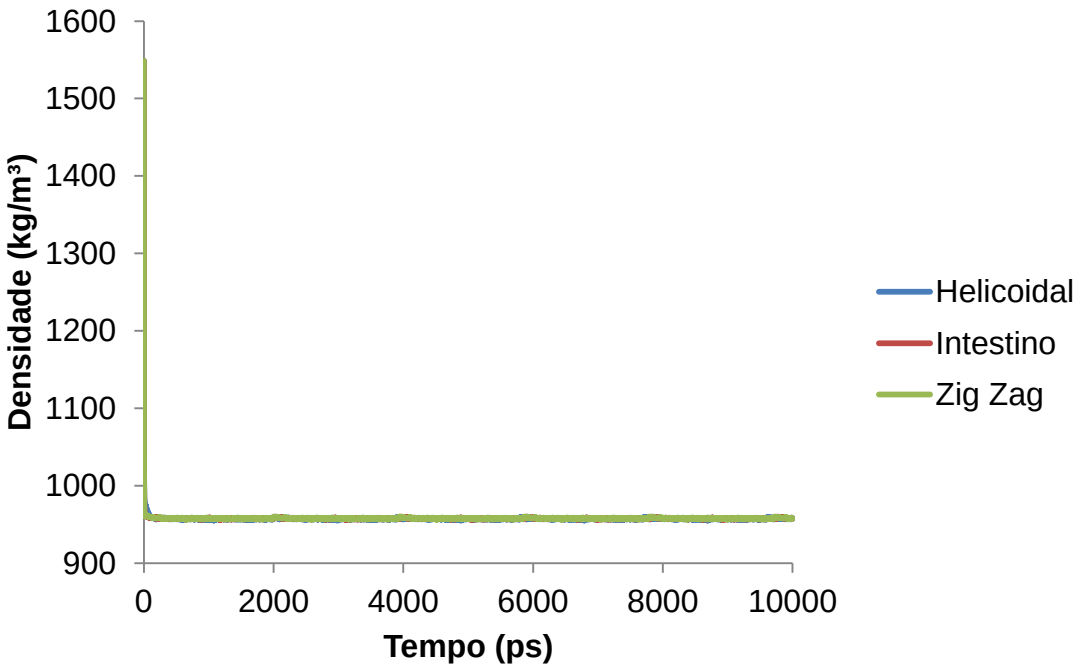

Figura 17 - Densidade da caixa de simula ção ao longo da simulação de dinâmica molecular para a albumina sérica bovina. As simulações foram feitas durante 10,0 nanossegundos com passos de 2,0 fentossegundos no *ensemble* NPT a temperatura de 300 K e press ão de 1,0 bar.

Assim como para o caso da lisozima, a evolução da densidade no início da simulação para avaliação do período transiente no caso da albumina sérica bovina também é mostrado na Figura 18:

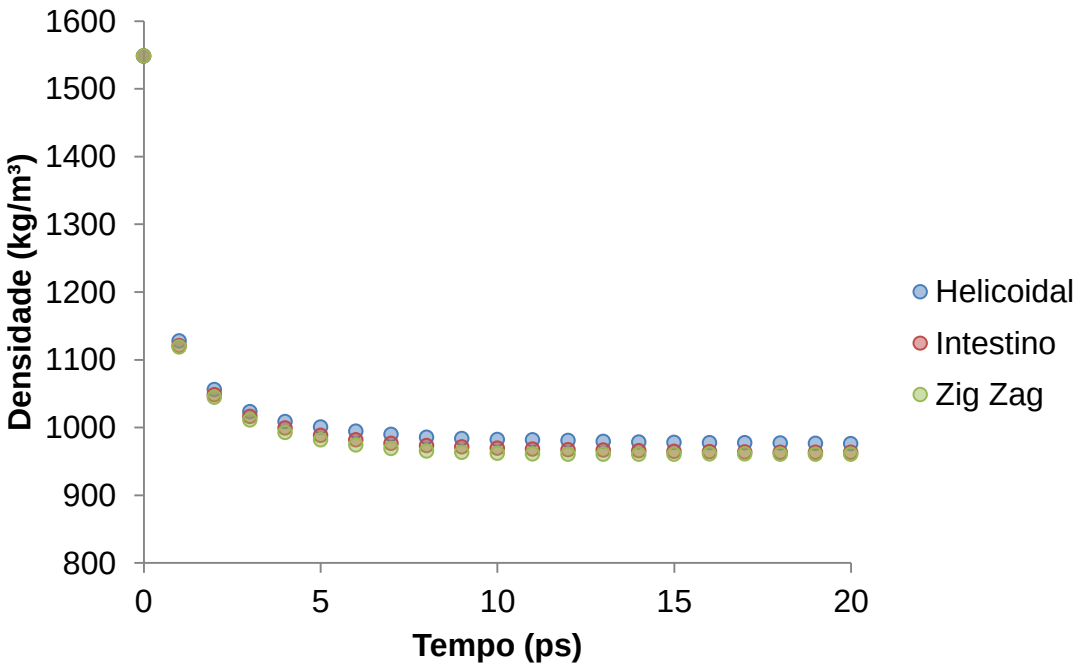

Figura 18 - Evolu ção das densidades para as simulações de dinâmica molecular com a lisozima. São apresentados os primeiros picossegundos da simulação para avaliação do período de equilibração da simula ção. As simulações partindo das três estruturas iniciais propostas são exibidas.

A partir dos resultados para as densidades conseguimos observar que a estrutura inicial não interfere na densidade final da simulação. Observa-se que tanto nas simulações com lisozima quanto albumina a densidade final assume aproximadamente o valor de 990 kg/m³ apresentando um desvio padrão de 1,2 kg/m³.

Apesar de estarmos utilizando diversos modelos *coarse-grained* observamos que os resultados s ão bastante satisfatórios. A densidade encontrada ao final das simulações é não só próxima a da água, que de acordo com a base de dados do NIST (*National Institute of Standards and Technology*) vale 996,56 kg/m³ nessas condições de pressão e temperatura, como muito semelhante em todas as simulações. Desvios da ordem de 0,66% em relação ao valor experimental da densidade da água pura são observados nos resultados obtidos.

A partir da observação dos transientes conseguimos notar que o equilíbrio, em relação a densidade da caixa de simulação de dinâmica molecular, para ambos

os casos é atingido de maneira rápida, com 10 picossegundos a densidade já está estabilizada para todos os casos nas duas proteínas testadas.

A segunda an álise realizada foi verificar a influência da estrutura inicial na estrutura final da biomolécula por meio do raio de giro. Como j á apresentado, foram propostas três estruturas iniciais para as simulações: helicoidal, intestino e zig zag. Em decorr ência da natureza do processo de *folding,* como j á discutido anteriormente, existe um controle cinético em conjunto com um controle termodinâmico. Imaginando que, *in vivo,* o processo de produ ção de uma proteína ocorre com a adição de um aminoácido por vez na cadeia em construção, espera-se que exista forte influência cinética, por conta da forma como é construída a cadeia, mas também forte influência termodinâmica, uma vez que se espera alguma estabilidade na conformaçã o nativa da biomolécula. É esperado, portanto, que exista influência, ao menos cinética, em relação a estrutura inicial da biomolécula na simulação. São propostas diferentes estruturas para verificar se essa influência cinética será significativa para a previsibilidade do método. Esper a-se que, mesmo com a mudança de estrutura inicial da biomolécula, os dados de espalhamento de raios X sejam próximos, uma vez que se trata de dados de baixa resolução, assim como a estrutura *coarse-grained*. Foram simuladas, então, as duas proteínas, lisozima e albumina sérica bovina, partindo, a cada simulação, de uma estrutura inicial diferente, mas mantendo todas as outras variáveis inalteradas. Ao final das simulações, foram comparados tanto a evolução do raio de giro da estrutura ao longo da simulação quanto o raio de giro médio dos últimos 500 picossegundos das simulações.

O raio de giro é uma propriedade geométrica que pode ser calculada a partir da equação 4.4.1 e também por meio da curva de espalhamento com a aproximação de Guinier:

$$R_G=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|\vec{r}-\vec{r}_{cm}\right|^2} \qquad (4.2.1)$$

em que N_B é o número total de resíduos de amino ácido e $\vec{r}_{cm}$ é a posição do centro de massa da molécula. Ele representa a distribuição dos resíduos ao redor do centro de massa da molécula. Fisicamente, o raio de giro pode ser interpretado, em uma proteína globular, como o grau de compactação dos resíduos, ou seja, quanto menor o raio de giro, mais compacta a proteína. No entanto, essa propriedade não nos diz muito a respeito da estrutura, afinal, ela reduz toda a estrutura a uma única medida. Muitas vezes o raio de giro é utilizado também como forma de avaliar as metaestruturas, estruturas assumidas pela proteína durante o processo de *folding,* ou estruturas de proteínas denaturadas.

Para o teste com lisozima, as estruturas, ao final das simulações de dinâmica molecular, atingidas para as três diferentes estruturas iniciais são apresentadas na Figura 19.

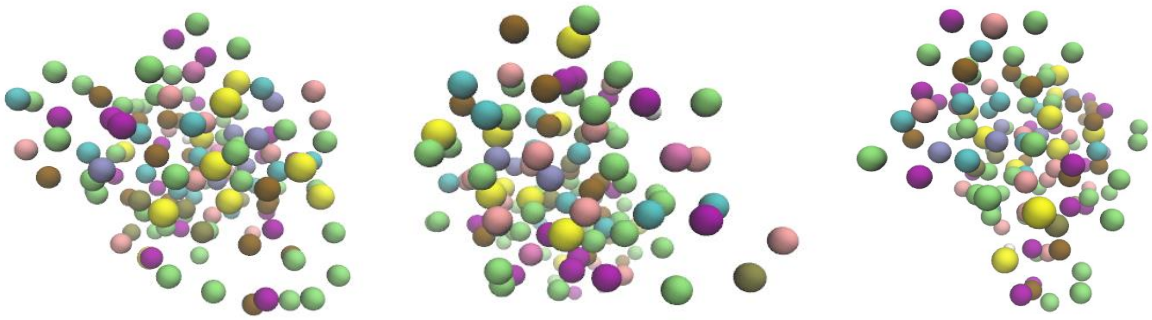

Figura 19 - Estruturas ao final das simulações moleculares para a lisozima. Nessas simulações variou-se a estrutura inicial da biomolécula para que fosse avaliada a influência dessa estrutura inicial, uma vez que é sabido existir influência cinética no processo de *folding* de uma biomolécula. Respectivamente, da esquerda para direita: helicoidal, intestino e zig zag.

Apesar de algumas diferenças visuais entre as estruturas, a análise do raio de giro nos fornece uma base de comparação melhor para a verificação da eficácia do campo de forças proposto. Abaixo, na Figura 20, é exibido como o raio de giro das três estruturas evoluiu ao longo dos 10,0 nanossegundos da simulação molecular.

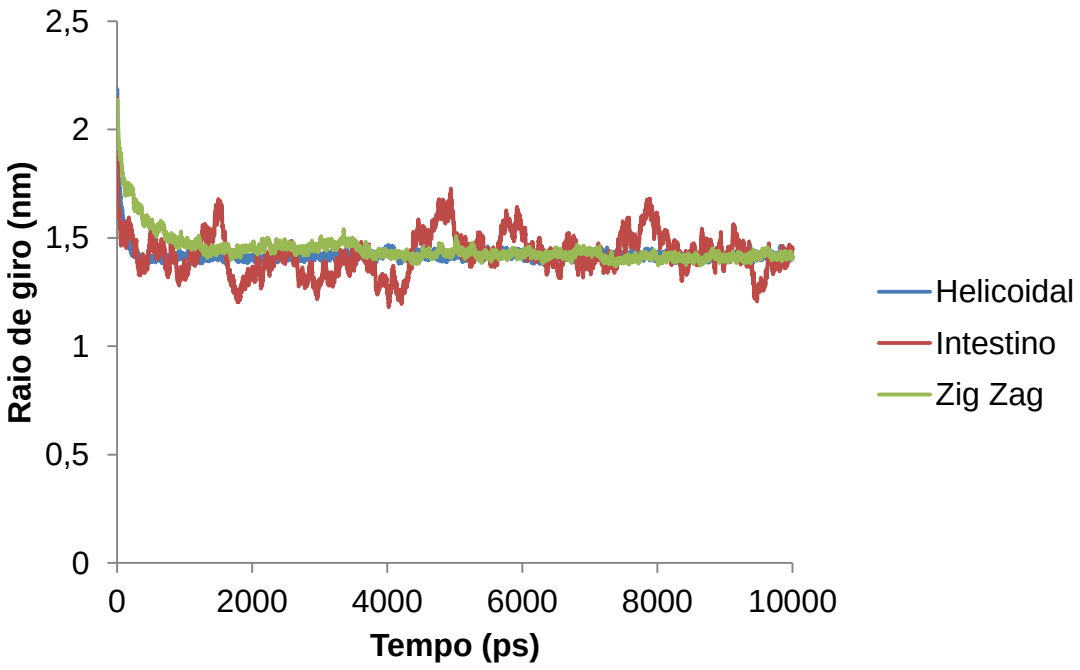

Figura 20 - Evolução do raio de giro para a lisozima nas três simulações de dinâmica molecular partindo cada vez de uma estrutura inicial diferente. O raio de giro é uma propriedade geométrica que traduz o quão compacta é a estrutura. O raio de giro da lisozima medido experimentalmente vale a 1,43 nm.

A partir da análise do gráfico consegue-se verificar que todas as estruturas iniciais propostas atingem um valor para a lisozima de aproximadamente 1,5 nm, como apresentado na Tabela 6. Os valores anotados na Tabela 6 são as médias calculadas para os últimos 8500 picossegundos de simulação de dinâmica molecular. O valor relatado na literatura para o raio de giro desta proteína é 1,43 nm (Krigbaum & Kügler, 1970). Nota-se, no entanto, que as estruturas helicoidal e zig zag atingem o valor estável mais rapidamente enquanto a simulação que parte da estrutura inicial intestino oscila em torno do valor médio.

Tabela 6 - Raio de giro da Lisozima calculado a partir da média dos últimos 8500 picossegundos das simulações de dinâmica molecular para as três estruturas iniciais diferentes. O valor relatado na literatura para o raio de giro da lisozima é de 1,43 nm. O desvio percentual é calculado sobre esse valor.

Estrutura	Raio de giro (nm)	Desvio (%)
Helicoidal	1,416	0,95
Intestino	1,421	0,60
Zig Zag	1,429	0,08

Para as simulações feitas com BSA, as estruturas, ao final das simulações de dinâmica molecular, atingidas para as três estruturas iniciais são apresentadas na Figura 21 abaixo:

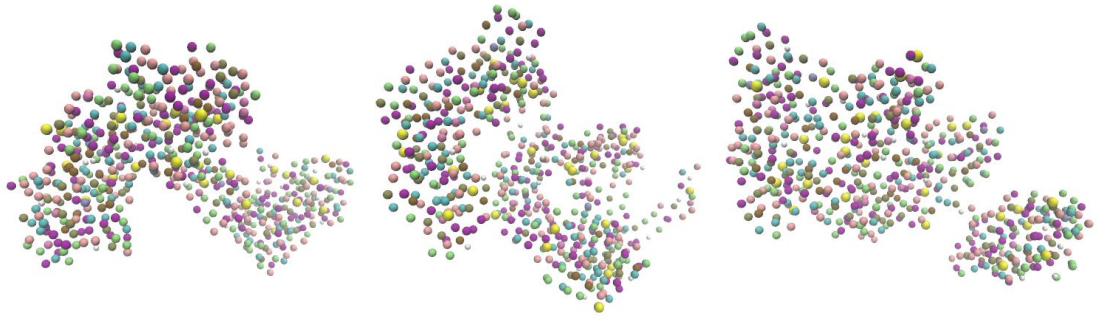

Figura 21 - Estruturas ao final das simulações moleculares para a albumina sérica bovina. Nessas simulações variou-se a estrutura inicial da biomolécula para que fosse avaliada a influência dessa estrutura inicial, uma vez que é sabido existir influência cinética no processo de *folding* de uma biomolécula. Respectivamente, da esquerda para direita: helicoidal, intestino e zig zag.

Novamente, apesar de algumas diferenças visuais a análise do raio de giro das estruturas nos fornece uma melhor base comparativa entre as três. Abaixo, na Figura 22, é exibido o gráfico que mostra a evolução do raio de giro dessas estruturas ao longo dos 10,0 nanossegundos da simulação de dinâmica molecular

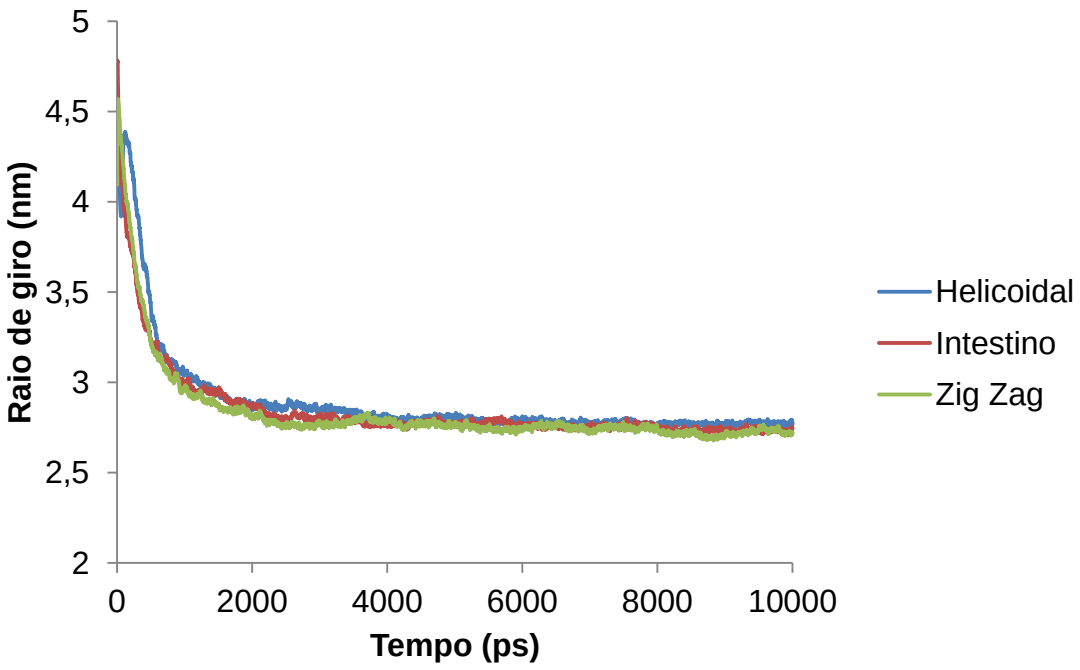

Figura 22 - Evolução do raio de giro para a albumina sérica bovina nas três simulações de dinâmica molecular partindo cada vez de uma estrutura inicial diferente. O raio de giro é uma propriedade geométrica que traduz o quão compacta é a estrutura. O raio de giro da albumina medido experimentalmente vale a 2,98 nm.

Observa-se que, apesar da maior concordância entre as três estruturas, no caso da albumina o raio de giro obtido a partir da simulação molecular desvia aproximadamente em 0,3 nm do experimental, que vale 2,98 nm (Anderegg et al., 1955). Os resultados finais são apresentados na Tabela 7:

Tabela 7 - Raio de giro da albumina sérica bovina calculado a partir da média dos últimos 8500 picossegundos das simulações de dinâmica molecular para as três estruturas iniciais diferentes. O valor relatado na literatura para o raio de giro da albumina é de 2,98 nm. O desvio percentual é calculado sobre esse valor.

Estrutura	Raio de Giro (nm)	Desvio (%)
Helicoidal	2,804	5,9
Intestino	2,775	6,9
Zig Zag	2,756	7,5

Como o raio de giro é uma importante grandeza que pode ser calculada a partir do experimento de espalhamento de raio X e os resultados são muito próximos aos experimentais, pode-se imaginar que, ao menos a parte inicial da curva de intensidade de espalhamento, o chamado regime de Guinier, que pode ser utilizado para a determinação do raio de giro, será calculada em boa correspondência com a estrutura cristalográfica. Não só isso, como também se pode inferir que a proteína simulada está assumindo, ao menos aproximadamente, uma forma tão compacta quanto a real, já que o raio de giro pode ser interpretado como o grau de compactação da proteína.

4.3. Cálculo da intensidade de espalhamento de raios-X

Para o cálculo da intensidade de espalhamento de raios-X foi utilizado o *software* CRYSOL (Svergun et al., 1995). Ao final da simulação de dinâmica molecular tem-se um arquivo de trajetória com 10 nanossegundos. Como foi observado anteriormente, pode-se dizer que o equilíbrio dinâmico, aqui colocado desta forma, pois a estrutura da biomolécula na simulação não se torna fixa em nenhum momento, é atingido antes dos 8500 picossegundos finais da simulação. Portanto, para o cálculo da intensidade de espalhamento são considerados esses últimos 8500 picossegundos.

Para que a média da intensidade de espalhamento destes últimos 8500 picossegundos seja calculada, a trajetória é particionada de um em um picossegundo por meio do comando editconf do simulador GROMACS. Essa partição gera 8500 arquivos com cada um contendo a estrutura que a biomolécula apresentava naquele picossegundo respectivo. A partir destes arquivos, o *software* CRYSOL (Svergun et al., 1995) foi utilizado para o cálculo da intensidade de espalhamento considerando que cada resíduo de aminoácido presente na molécula de proteína se comporta como um átomo de carbono, representando aqui uma estrutura de carbonos alfa.

Para o caso da lisozima não comparamos com dados, no entanto, estabelecemos essa comparação com o cálculo da curva de intensidade de espalhamento realizado a partir de uma simulação atomística com a estrutura cristalográfica da lisozima. Para a lisozima a curva de intensidade de espalhamento foi calculada até o valor q = 1 $Å^{-1}$. Isso equivale dizer que a resolução da curva é d = 6,28 Å aproximadamente. Este valor foi escolhido, pois é a máxima resolução disponível pelo *software* CRYSOL utilizado para o cálculo desta curva. Os resultados são exibidos na Figura 23 a seguir:

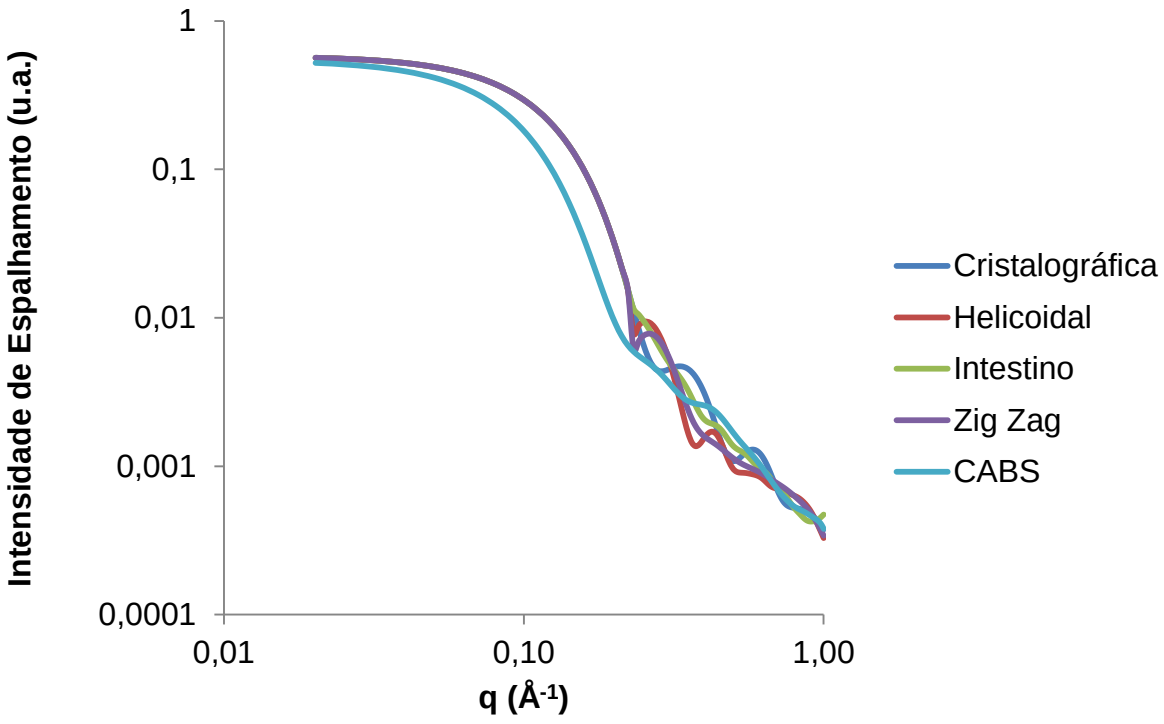

Figura 23 – Curva de intensidade de espalhamento de raios-X para a molécula de proteina lisozima. São apresentadas no gráfico as três curvas correspondentes às simulações de dinâmica molecular sendo iniciadas por cada estrutura inicial proposta, a curva da estrutura cristalográfica e a curva de intensidade de espalhamento calculada a partir da estrutura prevista pelo método CABS.

A partir da análise do gráfico acima, observa-se que a intensidade de espalhamento de raios X das três estruturas oriundas das simulações de dinâmica molecualr com o campo de forças proposto neste trabalho estão bastante concordantes com a intensidade de espalhamento da estrutura cristalográfica. Nota-se uma melhor concordância, no entanto, para o fator de

forma calculado a partir da estrutura que iniciou a simulação em intestino. Mesmo na parte final da curva é possível observar a tendência que é percebida na curva da simulação atomística com a estrutura cristalográfica. Com relação a curva da estrutura prevista pelo método CABS, observa-se que a intensidade de espalhamento desvia substanciamelnte da curva da estrutura cristalográfica. De modo a quantificar o quão concordante estão as curvas em relação à curva da estrutura cristalográfica, calculou-se o erro quadrático médio para as três simulações de dinâmica molecular realizadas, cada uma partindo de uma estrutura inicia distinta. O erro é calculado da seguinte maneira:

$$Er=\frac{\sum_{i=1}^{N_B}(\kappa_{CG}-\kappa_{CR})^2}{N_B} \qquad (4.3.1)$$

em que Er é o erro, N é o número de dados disponíveis, κ_{CG} é o valor da curva de alguma das estruturas *coarse-grained* utilizada nas simulaçoes de dinâmica molecular com o campo de forças desenvolvido neste trabalho e κ_{CR} é o valor da curva para a estrutura cristalográfica. Os erros calculados são apresentados na Tabela 8 abaixo:

Tabela 8 - Erro quadrático calculado para as três curvas de intensidade de espalhamento para a proteína lisozima previstas pelo método aqui proposto e o erro calculado para a estrutura prevista pelo método CABS. Cada curva de cada simulação de dinâmica molecular começando de uma estrutura distinta foi comparada com a curva calculada para a estrutura cristalográfica. O erro é calculado de acordo com a fórmula 4.3.1

Estrutura	Erro (Er)
Helicoidal	0,105
Intestino	0,032
Zig Zag	0,067
CABS	0,168

Verifica-se, portanto, que o apontamento a partir da análise visual das curvas realmente se traduz quando checamos os erros de cada uma das curvas. A curva para a estrutura intestino apresenta o menor erro dentre as curvas.

Para o caso da albumina sérica bovina comparamos com dados experimentais (Franco et al., 2015). Portanto a comparação foi não só feita em relação a uma estrutura cristalográfica, como também a estes dados. Os resultados obtidos são exibidos na Figura 24 abaixo:

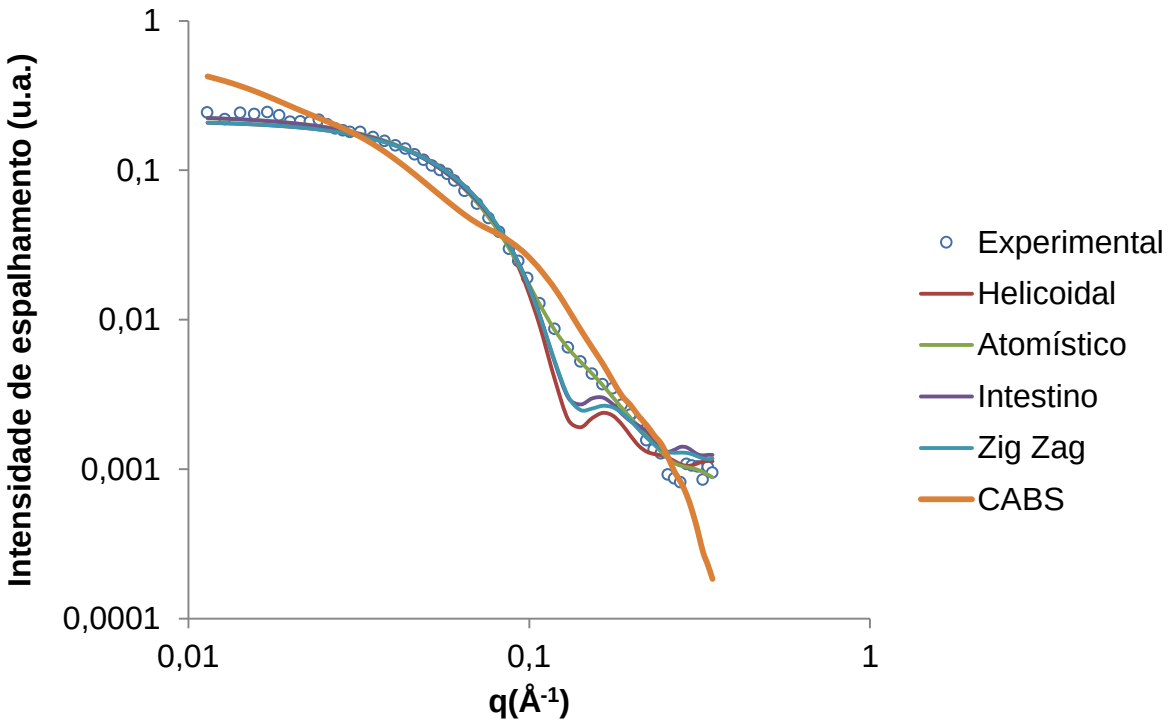

Figura 24 - Curva de intensidade de espalhamento de raios-X para a molécula de proteina albumina sérica bovina. São apresentadas no gráfico as três curvas correspondentes às simulações de dinâmica molecular sendo iniciadas por cada estrutura inicial proposta, a curva da estrutura cristalográfica, os dados experimentais de Franco (Franco et al., 2015) e a curva de intensidade de espalhamento calculada a partir da estrutura prevista pelo método CABS.

Novamente observa-se que a intensidade de espalhamento de raios X das estruturas simuladas estão bastante concordantes com os dados experimentais e com a intensidade de espalhamento de raios X calculada para a estrutura cristalográfica, especialmente para a intensidade de espalhamento calculada a partir da estrutura que inicia a simulação em intestino. A curva de intensidade

de espalhamento calculada para a estrutura prevista pelo método CABS apresenta desvios aparentemente maiores em relação à curva calculada a partir da estrutura cristalográfica.

De modo a quantificar o quão concordante estão as curvas em relação à curva da estrutura cristalográfica, calculou-se o erro quadrático médio para as três simulações de dinâmica molecular realizadas, cada uma partindo de uma estrutura inicia distinta e para a curva calculada a partir do método CABS. O erro novamente é calculado segundo a fórmula 4.3.1 e é apresentado na Tabela 9 abaixo:

Tabela 9 - Erro quadrático calculado para as três curvas de intensidade de espalhamento para a proteína albumina sérica bovina previstas pelo método aqui proposto e o erro calculado para a estrutura prevista pelo método CABS. Cada curva de cada simulação de dinâmica molecular começando de uma estrutura distinta foi comparada com a curva calculada para a estrutura cristalográfica. O erro é calculado de acordo com a fórmula 4.3.1

Estrutura	Erro (Er)
Helicoidal	0,077
Intestino	0,039
Zig Zag	0,041
CABS	0,232

Como observado visualmente, novamente confirmado pela quantificação do erro que a estrutura intestino apresentou o menor erro dentre as três estruturas inciais propostas para a realização das simulações de dinamica molecular.

Apesar de não ter existido motivação física para a escolha das estruturas iniciais, mas sim apenas ideias para que a proteína coubesse dentro da caixa de simulação, a estrutura em zig zag mostrou-se mais eficaz na previsão do fator de forma. Ademais, em ambos os casos apresentados o método aqui proposto foi mais eficaz que o método CABS na previsao da estrutura, comprovado pelo menor erro da curva de intensidade de espalhamento.

4.4. Sugestões para trabalhos futuros

Para futuros trabalhos, com o objetivo de tornar o cálculo da intensidade de espalhamento mais acurado, seria interessante construir um algoritmo para o cálculo da intensidade de espalhamento com base no fator de forma de cada aminoácido. No trabalho atual, como já mencionado, no cálculo da intensidade de espalhamento todos os resíduos são considerados carbonos alfa, e, portanto, foi considerado o fator de forma desses átomos.

Ademais, com o objetivo de tentar melhorar a resolução da previsão, tornar o campo de forças atomístico, ou diminuir o nível de compactação do campo de forças *coarse-grained*. Admitir uma partícula para a estrutura do *backbone* e uma partícula para a cadeia lateral, ou então alguma outra divisão para levar em conta mais propriedades para cada região da molécula de aminoácido.

Por fim, outra melhoria para o campo de forças seria a incorporação de termos no potencial de interação intermolecular para as ligações de hidrogênio e para as pontes dissulfeto, pois na versão apresentada neste trabalho ambas foram negligenciadas.

Bibliografia

AGOROGIANNIS, E. I.; AGOROGIANNIS, G. I.; PAPADIMITROU, A.; HADJIGEORGIOU, G. M. Protein misfolding in neurodegenerative diseases. *Neuropathology and Applied Neurobiology,* v. 30, p. 215-224, 2004.

ALAVI, F.; GHADARI, R.; ZAHEDI, M. Exporation of the binding properties of the human serum albumin sites with neurology drugs by docking and molecular dynamics simulation. *Journal of the Iranian Chemical Society*, v. 14, p. 19-35, 2017.

ALDER, B. J.; WAINWIRGHT, T. E. Studies in Molecylar Dynamics. I. General Method. *The Journal of Chemical Physics*, v. 31, p. 459-466, 1959.

ALLEN, M.; TILDESLEY, D. *Computer Simulation of Liquids.* New York: Oxford Science Publications, 2009.

ANDEREGG, J. W.; BEEMAN, W. W.; SHULMAN, S.; KAESBERG, P. An investigation of the size, shape and hydration of serum albumin by small-angle x-ray scattering. *Journal of the American Chemical Society,* v. 77, p. 2927-2937, 1955.

ANFINSEN, C. B. Principles that Govern Folding of Protein Chains *Science,* v. 181, p. 223-230, 1973.

ANGELESCU, D. G.; STENHAMMAR, J.; LINSE, P. Packaging of a flexible polyelectrolyte inside a viral capsid: effect of salt concentration and salt valence. *Journal of Physical Chemistry B,* v. 111, p. 8477-8485, 2007.

BAHAR, I.; JERNIGAN, R. L. Inter-residue potentials in globular proteins and the dominance of highly specific hydrophilic interactions at close separation. *Journal of Molecular Biology,* v. 266, p. 195-214, 1997.

BERENDSEN, H. J. C.; POSTMA, J. P. M.; VAN GUNSTEREN, W. F.; DINOLA, A.; HAAK, J. R. Molecular dynamics with coupling to an external bath. *Journal of Chemical Physics,* v. 81, p. 3684-3690, 1984.

BERENDSEN, H. J. C.; SPOEL, D.; DRUNEN, R. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communcation,* v.

91, p. 43-56, 1995.

BLANCO, M. A.; SAHIN, E.; ROBINSON, A. S.; ROBERTS, C. J. Coarse-Grained Model for Colloidal Protein Interactions, B22, and Protein Cluster Formation. *Journal of Physical Chemistry B*, v. 117, p. 16013-16028, 2013.

BORN, V. M. Quantenmechanick der Stoβvorgänge. *Zeitschrift für Physik*, v. 37, p. 863-867, 1926.

BRADLEY, P.; MISURA, K. M. S.; BAKER, D. Toward High-Resolution de Novo Structure Prediction for Small Proteins. *Science*, v. 309, p. 1868-1871, 2005.

BROOKS, B. R.; BRUCCOLERI, R. E.; OLAFSON, B. D.; STATES, D. J.; SWAMINATHAN, S.; KARPLUS, M. CHARMM: a program for macromolecular energy, minimization, and dynamics simulations. *Journal of Computational Chemistry*, v. 4, p. 187-217, 1983.

BROOKS, B. R.; BRUCCOLERI, R. E.; OLAFSON, B. D.; STATES, D. J.; SWAMINATHAN, S.; KARPLUS, M. CHARMM: a program for macromolecular energy, minimization, and dynamics calculation. *Journal of Computational Chemistry,* v. 4, p. 187-217, 1983.

BROOKS, C. L. Simulations of protein folding and unfolding. *Current Opinion in Structural Biology,* v. 8, p. 222-226, 1998.

BRYNGELSON, J. D.; ONUCHIC, J. N.; SOCCI, N. D.; WOLYNES, P. G. Funnels, Pathways and Energy Landscape of Protein Folding: A Synthesis. *Proteins: Structure, Function, and Bioinformatics*, v. 21, p. 167-195, 1995

BUCKINGHAM, R. A. The classical equation of state of gaseous helium, neon and argon. *Queen's University,* 1938.

BULACU, M.; GOGA, N.; ZHAO, W.; ROSSI, G.; MONTICELLI, L.; PERIOLE, X.; TIELEMAN, D.; MARINK, S. Improved angle potentials for coarse-grained molecular dynamics simulations. *Journal of Chemical Theory and Computation,* v. 9, p. 3282-3292, 2013.

BULACU, M.; VAN DER GIESSEN, E. Effect of bending and torsion rigidity on self-diffusion in polymer melts: A molecular-dynamics study. *The Journal of Chemical Physics*, v. 123, p. 3282-3292, 2013.

BURIONI, F.; CASSI, D.; CECCONI, F.; VULPIANI, A. Topological thermal instability and length of proteins. *Proteins*, v. 55, p. 529-535, 2004.

CHACÓN, P.; MORÁN, F.; DIAZ, J. F.; PANTOS, E.; ANDREU, J. M. Low-Resolution Structures of Proteins in Solution Retrieved from X-Ray Scattering with a Genetic Algorithm. *Biophysical Journal*, v. 74, p. 2760-2775, 1998.

CHACÓN, P.; MORÁN, F.; DÍAZ, J. F.; PANTOS, E.; ANDREU, J. M. Low-Resolution Structures of Proteins in Solution Retrieved from X-ray Scattering with a Genetic Algorithm. *Biophysical Journal*, v. 74, p. 2760-2775, 1998.

CLEMENTI, C. Coarse-grained models of protein folding: toy models or predictive tools? *Current Opinion in Structural Biology,* v. 18. p. 10-15, 2007.

COMPIANI, M.; CAPRIOTTI, E. Computational and Theoretical Methods for Protein Folding. *Biochemistry – American Chemical Society,* v. 52, p. 8601-8624, 2013.

COMPIANI, M.; CAPRIOTTI, E.; CASADIO, R. Dynamics of the minimally frustrated helices determine the hierarchical folding of small helical proteins. *Physical Review E,* v. 69, p. 1-8, 2004.

DARDEN, T.; DARRIN, Y.; PEDERSEN, L. Particle mesh Ewald: An N-log(N) for Ewald sums in large systems. *Journal of Chemical Physics,* v. 98, p. 89-92, 1993.

DAVIES, M. N.; TOSELAND, C. P.; MOSS, D. S.; FLOWER, D. R. Bechmarking pK(a) prediction. *BMC Biochemistry*, p. 7-18, 2006.

DEPAUL, A. J.; THOMPSON, E. J.; PATEL, S. S.; HALDEMAN, K.; SORIN, E. J. Equilibrium conformational dynamics in an RNA tetraloop from massively parallel molecular dynamics. *Nucleic Acids Research,* v. 38, p.4856-4867, 2010.

DILL, K. A. Theory of the Folding and Stability of Globular Proteins. *Biochemistry - American Chemical Society,* v. 24, p. 1501-1509, 1985.

DOLINSKY, T. J.; CZODROWSKI, P.; LI, H.; NIELSEN, J. E.; JENSEN, J. H.; KLEBE, G.; BAKER, N. A. PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. *Nucleic Acids Research*, v. 35, p. 522-525, 2007.

DOLINSKY, T. J.; NIELSEN, J. E.; MCCAMMON, J. A.; BAKER, N. A. PDB2PQR: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations. *Nucleic Acids Research*, v. 32, p. 665-667, 2004.

FEIGIN, L. A.; SVERGUN, D. I. *Structure Analysis by Small-angle X-ray and Neutron Scattering,* Plenum Press, 1987.

FERGUSON, D. M. Parametrization and evaluation of a flexible water model. *Journal of Computational Chemistry*, v. 16, p. 501-511, 1995.

FERGUSON, D. M. Parametrization and evaluation of a flexible water model. *Journal of Computational Chemistry*, v. 16, p. 501-511, 1995.

FERSHT, A. R. Transition-state structure as a unifying basis in protein-folding mechanisms: Contact order, chain topology, stability, and the extended nucleus mechanism. *Proceedings of the National Academy of Sciences of the United States of America,* v. 97, p. 1525-1529, 2000.

FRANCO, L. F.; OLIVEIRA, C. L. P. d.; FILHO, P. d. A. P. Thermodynamics of protein aqueous solutions: from the strucutre fator to the osmotic pressure. *AIChE J.,* v. 61, p. 2817-2880, 2015.

FRANCO, L. F. M.; CASTIER, M.; ECONOMOU, I. G. Anisotropic parallel self-diffusion coeficientes near the calcite surface: A molecular dynamics study. *The Journal of Chemical Physics*, v. 145, p. 084702-1-7, 2016.

FREEMAN, S.; HARRINGTON, M.; SHARP, J. *Biological Science*. Pearson Education Canada, 2ª ed., 2012.

GARCÍA, A. E.; SANBONMATSU, K. Y. α-Helical stabilization by side chain shielding of backbone hydrogen bonds. *Proceedings of the National Academy of Sciences of the United States of America*, v. 99, p. 2782-2787, 2002.

GIBBS, J. W. *Elementary Principles in Statistical Mechanics.* New York: Charles Scribner's Sons, 1902.

GIBBS, J. W. On the fundamental formula of statistical mechanics, with applications to astronomy and thermodynamics. *Proceedings of the American Association for the Advancement of Science*, v. 33, p. 57-58, 1884.

GRIFFITHS, D. J. *Mecânica Quântica.* Pearson, 2011.

GUINIER, A.; FOURNET, G. *Small-angle scattering of x-rays,* John Wiley & Sons , Inc., 1955.

HEILBRON, J. L. *Physics a short history.* Oxford University Press, 1ª ed., 2016.

HU, S.; WANG, M.; CAI, G.; HE, M. Genetic Code-guided Protein Synthesis and Folding in *Escherichia coli. Journal of Biological Chemistry,* v. 288, p. 30855-30861, 2013.

ITZHAKI, L. S.; OTZEN, D. E.; FERSHT, A. R. The Structure of the Transition State for Folding of Chymotrypsin Inhibitor 2 Analysed by Protein Engineering Methods: Evidence for a Nucleation-condensation Mechanism for Protein Folding. *Journal of Molecular Biology,* v. 254, p. 260-288, 1995.

JONES, J. E. On the determination of molecular fields. – II. From the equation state of a gas. *Exhibition Senior Research Student,* 1924.

JORGENSEN, W. L.; MAXWELL, D. S.; RIVES-TIRADO, J. Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. *Journal of American Chemical Society,* v. 118, p. 225-236, 1996.

KENNDLER, E.; VCELAKOVA, K.; ZUSKOVA, I.; GAS, B. Determination of cationic mobilities and pKa values of 22 amino acids by capillary zone electrophoresis. *Electrophoresis,* v. 25, p. 309-317, 2004.

KIKHNEY, A. G.; SVERGUN, D. I. A practical guide to small angle X-ray scattering (SAXS) of flexible and intrinsically disordered proteins. *FEBS Letters*, v. 589, p. 2570-2577, 2015.

KLIMECKA, M. M.; CHRUSZCZ, M.; FONT, J.; SKARINA, T.; SHUMILIN, I.; ONOPRYIENKO, O.; POREBSKI, P. J.; CYMBOROWSKI, M.; ZIMMERMAN, M. D.; HASSEMAN, J.; GLOMSKI, I. J.; LEBIODA, L.; SAVCHENKO, A.; EDWARDS, A.; MINOR, W. Structural analysis of a putative aminoglycoside N-acetyltransferase from *Bacillus anthracis*. *Journal of Molecular Biology,* v. 410, p. 411-423, 2011.

KOLINSKI, A. Protein modeling and structure prediction with a reduced representation. *Acta Biochimica Polonica*, v. 51, p. 349-371, 2004.

KOLINSKI, A.; KMIECIK, S.; JAMROZ, M.; BLASZCZYK, M. CABS-fold: Server for the de novo and consensus-based prediction of protein structure. *Nucleic Acids Research*, v. 41, p. 406-411, 2013.

KRIGBAUM, W. R.; KÜGLER, F. R. Molecular conformation of egg-white lysozyme and bovine alpha-lactalbumin in solution. *Biochemistry,* v. 9, p. 1216-1223, 1970.

KUCZERA, K. *Dynamics and thermodynamics of globins. Recent Developments in Theoretical Studies of Proteins*, pp. 1-64, Singapore: World Scientific, 1996.

KUNDU, S.; SORENSEN, D. C.; PHILLIPS, G. N. Automatic domain decomposition of proteins by Gaussian network model. *Proteins,* v. 57, p. 725-733, 2004.

KYTE, J.; DOOLITTLE, R. F. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology,* v. 157, p. 105-132, 1982.

LANDAU, L. D.; LIFSHITZ, E. M. *Classical Theory of Fields,* Pergamon Press, 1985.

LEMOS, N. A. *Mecânica Analítica.* Editora Livraria da Física, 2007.

LEVINTHAL, C. How to fold graciously. *Mössbaun Spectroscopy in Biological Systems Proceedings,* v. 67, p. 22-24, 1969.

LEVITT, M.; WARSHEL, A. Computer simulation of protein folding. *Nature*, v. 253, p. 694-698, 1975.

LIWO, A.; KHALILI, M.; SCHERAGA, H. A. Ab initio simulations of protein-folding

pathways by molecular dynamics with the united-residue model of polypeptide chains. *Proceedings of the National Academy of Sciences,* v. 102, p. 2362-2367, 2005.

LIWO, A.; MAISURADZE, G. G.; SENET, P.; CZAPLEWSKI, C.; SCHERAGA, H. A. Investigation of protein folding by coarse-grained molecular dynamics with the UNRES force field. *Journal of Physical Chemistry A*, v. 114, p. 4471-4485, 2010.

LUKIN, J. A.; KONTAXIS, G.; SIMPLACEANU, V.; YUAN, Y.; BAX, A.; HO, C. Quaternary structure of hemoglobin in solution. *Proceedings of the National Academy of Sciences of the United States of America,* v. 100, p. 517-520, 2003.

MAHON, B. *The man who changed everything – the life of James Clerk Maxwell*, Hoboden, New Jersey, Wiley, 2003.

MATUBAYASI, N. Free-energy analysis of protein solvation with all-atom molecular dynamics simulation combined with a theory of solutions. *Current Opinion in Structural Biology*, v. 43, p. 45-54, 2017.

MCQUARRIE, D. A. *Statistical Mechanics.* University Science Books, 2000.

MELLER, J. Molecular Dynamics. *Encyclopedia of Life Sciences,* 2001.

MORSE, P. M. Diatomic molecules according to the wave mechanics. II. vibrational levels. *Physical Review*, v. 34, p. 57-64, 1929.

MORSE, P. M. Diatomic molecules according to the wave mechanics. II. vibrational levels. *Physics Review*, v. 34, p. 57-64, 1929.

NANDY, A.; ROY, K.; SAHA, A. Structural exploration of PPARγ modulators through pharmacophore mapping, fragment-based design, docking, and molecular dynamics simulation analysis. *Medicinal Chemistry Research*, v. 26, p. 52-63, 2016.

NGUYEN, H. D.; MARCHUT, A. J.; HALL, C. K. Solvent effects on the conformational transition of a polyalanine peptide. *Protein Sciences,* v. 13, p. 2909-2924, 2004.

NIST: National Institute of Standards and Technology disponível em: <https://www.nist.gov/>. Acesso em 18 de abril de 2017.

ONUCHIC, J. N., SCHULTEN-LUTHEY, Z.; WOLYNES, P. G. Theory of protein folding: The energy landscape perspective. *Annual Revision of Physical Chemistry,* v. 48, p. 545-600, 1997.

ONUCHIC, J. N.; SCHULTEN-LUTHEY, Z.; WOLYNES, P. G. Theory of protein folding: the energy landscape perspective. *Annual Review of Physical Chemistry*, v. 48, p. 545-600, 1997.

ONUCHIC, J. N.; WOLYNES, P. G. Theory of protein folding. *Current Opinion in Structural Biology,* v. 14, p. 70-75, 2004.

OOSTENBRINK, C.; VILLA, A.; MARK, A. E.; VAN GUNSTEREN, W. F. A biomolecular force field based on the free enthalpy of hydration and solvation: The GROMOS force-field parameter sets 53A5 and 53A6. *Journal of Computational Chemistry,* v. 25, p. 1656-1676, 2004.

PACHENKO, A. R.; SCHULTEN-LUTHEY, Z.; WOLYNES, P. G. Foldons, protein structural modules, and exons. *Proceedings of the National Academy of Sciences of the United States of America*, v. 93, p. 2008-2013, 1996.

PARRINELLO, M.; BUSSI, G.; DONADIO, D. Canonical sampling trough velocity rescaling. *Journal of Chemical Physics,* v. 126, p. (01401) 1-7, 2007.

PATHRIA, R. K.; BEALE, P. D. *Statistical Mechanics.* Elsevier, 1996.

PAULING, L.; COREY, R. B.; BRANSON, H. R. The structure of proteins; two hydrogen-bonded helical configurations of the polypeptide chain. *Proceedings of the National Academy of Sciences of the United States of America*, v. 37, p. 205-211, 1951.

PO, H. N.; SENOZAN, N. M. The Henderson-Hasselbalch equation: Its history and limitations. *Journal of Chemical Education*, v. 78, p. 1499-1503, 2001.

REIS, R. A.; PAREDES, M. L. L.; CASTIER, M.; TAVARES, F. W. Evaluation of mixing and combining rules for asymmetric Lennard-Jones chain mixtures: Effect of

segment diameter, energy interaction, and chain lengh. *Fluid Phase Equilibria,* v. 259, p. 123-134, 2007.

RICHARDS, F. M. Areas, volumes, packing and protein structure. *Annual Review of Biophysics and Bioengineering,* v. 6, p. 151-176, 1977.

ROSENFELD, R.; VAJDA, S.; DELISSI, C. Flexible Docking and Design. *Annual Revision of Biophysical and Biomolecular Structure,* v. 24, p. 677-700, 1995.

RZEPIELA, A. J.; LOUHIVUORI, M.; PETER, C. Hybrid simulations: combining atomistic and coarse-grained force field using virtual sites. *Physical Chemistry Chemical Physics*, v. 13, p. 437-448, 2011.

SAUNDERS, M. G.; VOTH, G. A. Coarse-Graining Methods for Computational Biology. *Annual Review of Biophysics*, v. 42, p. 73-93, 2013.

SAVINO, C.; MIELE, A. E.; DRAGHI, F.; JOHNSON, K. A.; SCIARA, G.; BRUNORI, M.; VALLONE, B. Pattern of cavities in globins: The case of human hemoglobina. *Biopolymers*, v. 91, p. 1097-1107, 2009.

SHI, Q.; IZVEKOV, S.; VOTH, G. A. Mixed atomistic and coarse-grained molecular dynamics: simulation of a membrane-bound ion channel. *The Journal of Physical Chemistry B*, v. 110, p. 45-48, 2006.

SHINODA, W.; DEVANE, R.; KLEIN, M. L. Multi-property fitting and parametrization of a coarse grained model for aqueous surfactants. *Molecular Simulation,* v. 33, p. 27-36, 2007.

SMITH, A. V.; HALL, C. K. Assembly of a tetrameric alpha-helical bundle: computer simulations on an intermediate-resolution protein model. *Proteins,* v. 44, p. 376-391.

SORIN, E. J.; PANDE, V. S. Exploring the Helix-Coil transition via all-atom equilibrium ensemble simulations. *Biophysics Journal,* v. 88, p. 2472-2493, 2005.

STUHRMANN, H. B. Ein neues Verfahren zur Bestimmung der Oberflaechenform und der inneren Struktur von geloesten globularen Proteinen aus Roentgenkleinwinkelmessungen. *Zeitschrift für Physik. Chem. Neue Folge*, v. 72, p. 177-198, 1970.

STUHRMANN, H. B. Interpretation of small-angle scattering functions of dfilute solutions and gases. A representation of the structures related to a one-particle scattering function. *Acta Crystallographica*, Section A, v. A26, p. 297-306, 1970.

SUGANYA, S.; NANDAGOPAL, B.; ANBARASU, A. Natural inhibitors of HMG-CoA reductase – An in silico approach through molecular docking and simulation studies. *Journal of Cellular Biochemistry*, v. 118, p. 52-57, 2017.

SVERGUN, D. I. Restoring low resolution structure of biological macromolecules from solution scattering using simulated annealing. *Biophysical Journal,* v. 76, p. 2879-2886, 1999.

SVERGUN, D. I.; BARBERATO, C.; KOCH, M. H. J. CRYSOL – a Program to Evaluate X-ray Solution Scattering of Biological Macromolecules from Atomic Coordinates. *Journal of Applied Crystallography,* v. 28, p. 768-773, 1995.

SVERGUN, D. I.; KOCH. M. H. J. Small-angle scattering studies of biological macromolecules in solution. *Reports on Progress in Physics,* v. 66, p. 1735-1782, 2003.

SVERGUN, D. I.; PETOUKHOV, M. V.; KOCH, M. H. J. Determination of domain structure proteins from x-ray solution scattering. *Biophysical Journal*, v. 80, p. 2946-2953, 2001.

TAKADA, S. Coarse-grained molecular simulations of large biomolecules. *Current Opinion in Structural Biology,* v. 22, p. 130-137, 2012.

TOZZINI, V. Coarse-grained models for proteins. *Current Opinion in Structural Biology*, v. 15, p. 144-150, 2005.

URBAN, B.; CRUZ, L.; DING, F.; SAMMOND, D.; KHARE, S.; BULDYREV, S. V.; STANLEY, H. E.; DOKHOLYAN, N. V. Molecular dynamics simulation of amyloid betha dimer formation. *Biophysics Journal,* v. 87, p. 2310-2321, 2004.

VAN GUNSTEREN, W. F.; BILLETER, S. R.; EISING, A. A.; HÜNENBERGER, P. H./ KRÜGER. P.; MARK, A. E.; SCOTT, W. R. P.; TIRONI, I. G. *Biomolecular*

simulation: The GROMOS96 manual and user guide. Hochschulverlag AG an der ETH Zürich, 1996.

VAN GUTEREN, W. F.; BILLETER, S. R.; EISING, A. A.;HÜNENBERGER, P. H.; KRÜGER, P.; MARK, A. E.; SCOTT, W. R. P.;TIRONI, I. G. *Biomolecular Simulation: The GROMOS96 manual and user guide.* Zürich, Switzerland: Hochshulverlag AG an der ETH Zürich, 1996.

VANOMMESLAEGHE, K.; HATCHER, E.; ACHARYA, C.; KUNDU, S.; ZHONG, S.; SHIM, J.; DARIAN, E.; GUBENCH, O.; LOPES, P.; VOROBYOV, I.; MACKERELL JR, A. D. CHARMM General Force Field (CGenFF): A force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields. *Journal of Computational Chemistry,* v. 31, p. 671-690, 2010.

VOICESCU, M.; IONESCU, S.; ANGELESCU, D. G. Spectroscopic and coarse-grained simulation studies of the BSA and HSA protein adsorption on silver nanoparticles. *Journal of Nanoparticles Research,* v. 14, p. (1174) 1-13, 2012.

WALKER, A.; BONOMI, R.; POPOV, V.; GELOVANI, J. G. Investigating carbohydrate based ligands for galectin-3 with docking and molecular dynamics studies. *Journal of Molecular Graphics and Modellings*, v. 71, p. 211-217, 2017.

WARNER JR, H. R. Kinetic theory and rheology of dilute suspensions of finitely extendible dumbbells. *Ind. Eng. Chem. Fundam.*, v. 11, p. 379-387, 1972.

WARNER JR, H. R. Kinetic theory and rheology of dilute suspensions of finitely extendible dumbbells. *Industrial & Engineering Chemistry Fundamentals,* v. 11, p. 379-387, 1972.

WU, Y.; YUAN, X; GAO, X.; FANG, H.; ZI, J. Universal behavior of localization of residue fluctuations in globular proteins. *Physical Review E*, v. 67, p. 041909, 2003.

ZHANG, Y. Progress and challenges in protein structure prediction. *Current Opinion in Structural Biolology,* v. 18, p. 342-348, 2008.

ZHANG, Y.; APARICIO, R.; REIS, M. A. Improving protein template recognition by using small-angle x-ray scattering profiles. *Biophysical Journal*, v. 101, p. 2770-

2781, 2011.

ZHIGILEI, L. V.; GARRISON, B. J. Velocity distributions of molecules ejected in laser ablation. *Applied Physics Letters*, v. 71, p. 551-553, 1997.

APÊNDICE A – CÓDIGO GENINP

O código GENINP foi escrito para que um arquivo do banco de dados de proteínas (PDB) fosse lido e traduzido para um arquivo de *input* para os outros códigos. A saída deve ser a sequência do polipeptídio, o tamanho da aresta da caixa cúbica de simulação, a quantidade de resíduos de aminoácidos presentes na molécula de proteína e o pH da simulação.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define TRUE 1
#define MAX 50

void main(int argc, char **argv)
{
    char ent[] = "-f"; /* Comparação para ler o arquivo de entrada */
    char *arqent; /* String para receber o nome do arquivo de entrada */
    char sai[] = "-o"; /* Comparação para ler o arquivo de saída */
    char *arqsai; /* String para receber o nome do arquivo de saída */
    char res[MAX]; /* Vetor para armazenar o nome do resíduo no arquivo
PDB */
    char verif[4]; /* Vetor para verificação da extensão dos arquivos */
    char verinp[] = ".inp"; /* Verificação da extensão do arquivo de saída
informado pelo usuário */
    char verpdb[] = ".pdb"; /* Verificação da extensão do arquivo de
entrada informado pelo usuário */
    char erro[] = "ERRO: Arquivo de entrada não especificado.\n"; /* Erro
de arquivo de entrada não especificado */
    char erroextpdb[] = "ERRO: Arquivo de entrada com a extensão
incorreta.\n"; /* Erro de extensão de arquivo de entrada incorreta */
```

```
    char erroarg[] = "ERRO: Número de argumentos inconsistente.\n"; /*
Erro no número de argumentos */
    char exito[] = "Arquivo input gerado com êxito.\n"; /* Mensagem de
êxito de execução */
    char lixo[2*MAX]; /* String para jogar informações inúteis fora */
    char ato[MAX]; /* String para verificação se foi encontrado o termo
ATOM no arquivo pdb */
    char atom[] = "ATOM"; /* String para comparação com a string ato */
    char anisou[] = "ANISOU"; /* String para comparação com a string ato*/
    FILE *INP; /* Arquivo de saída */
    FILE *PDB; /* Arquivo de entrada */
    double descarte; /* Descarte de valores lidos */
    float ph; /* pH requerido do usuário */
    float tam; /* Tamanho da caixa sugerido */
    int compe; /* Comprimento da string do arquivo de entrada */
    int comps; /* Comprimento da string do arquivo de saída */
    int cont; /* Contador geral */
    int f; /* Índice do arquivo de entrada */
    int numres; /* Quantidade de resíduos */
    int o; /* Índice do arquivo de saída */
    int size; /* Tamanho da string */

    f = 0;
    o = 0;
    numres = 0;
    arqent = "";
    arqsai = "";

    if(argc%2 == 0)
    {
        printf("%s", erroarg);
        exit(EXIT_FAILURE);
    }

    for(cont = 0; cont < argc; cont++)
    {
        if(strcmp(argv[cont], ent) == 0)
        {
            f = cont + 1;
            arqent = argv[cont + 1];
        }

        if(strcmp(argv[cont], sai) == 0)
        {
            o = cont + 1;
            arqsai = argv[cont + 1];
        }
    }

    if(f%2 != 0 || arqent == "" || arqent == sai || f == 0)
    {
        printf("%s", erro);
        exit(EXIT_FAILURE);
    }

    compe = strlen(arqent);
    comps = strlen(arqsai);
    if(strcmp(strcpy(verif, &arqent[compe - 4]), verpdb) != 0)
    {
        printf("%s", erroextpdb);
        exit(EXIT_FAILURE);
```

```
    }
    else if(strcmp(arqsai,"") != 0)
    {
        if(strcmp(strcpy(verif, &arqsai[comps - 4]), verinp) != 0)
        {
            strcat(arqsai, verinp);
        }
    }

    printf("\nPor favor, digite o pH da simulacao: ");
    scanf("%f", &ph);


    PDB = fopen(arqent, "r");

    /* Loop para encontrar o primeiro registro ATOM */

    while(TRUE)
    {
        fscanf(PDB, "%s", ato);

        if(strcmp(atom, ato) == 0)
        {
            break;
        }
        else
        {
            fscanf(PDB, "%[^\n]%*c", lixo);
        }

    }

    fscanf(PDB, "%s%s", lixo, lixo);

    if(strcmp(lixo, "NH1AARG") == 0 || strcmp(lixo, "NH1BARG") == 0 ||
strcmp(lixo, "NH2AARG") == 0 || strcmp(lixo, "NH2BARG") == 0)
    {
        fscanf(PDB, "%s%d%s%s%s%s", lixo, &numres, lixo, lixo, lixo,
lixo);
    }
    else
    {
        fscanf(PDB, "%s%s%d%s%s%s%s", lixo, lixo, &numres, lixo, lixo,
lixo, lixo);
    }

    if(strcmp(lixo, "1.00") == 0 || strcmp(lixo, "0.50") == 0)
    {
        fscanf(PDB, "%s%s%s", lixo, lixo, ato);
    }
    else
    {
        fscanf(PDB, "%s%s", lixo, ato);
    }

    while(TRUE)
    {
        if(strcmp(atom, ato) == 0)
        {
            fscanf(PDB, "%s%s", lixo, lixo);
```

```
            if(strcmp(lixo, "NH1AARG") == 0 || strcmp(lixo, "NH1BARG") ==
0 || strcmp(lixo, "NH2AARG") == 0 || strcmp(lixo, "NH2BARG") == 0)
            {
                fscanf(PDB,  "%s%d%s%s%s%s",  lixo,  &numres,  lixo,  lixo,
lixo, lixo);
            }
            else
            {
                fscanf(PDB, "%s%s%d%s%s%s%s", lixo, lixo, &numres, lixo,
lixo, lixo, lixo);
            }

            if(strcmp(lixo, "1.00") == 0 || strcmp(lixo, "0.50") == 0)
            {
                fscanf(PDB, "%s%s%s", lixo, lixo, ato);
            }
            else
            {
                fscanf(PDB, "%s%s", lixo, ato);
            }

        }
        else if(strcmp(anisou, ato) == 0)
        {
            fscanf(PDB, "%s%s", lixo, lixo);

            if(strcmp(lixo, "NH1AARG") == 0 || strcmp(lixo, "NH1BARG") ==
0 || strcmp(lixo, "NH2AARG") == 0 || strcmp(lixo, "NH2BARG") == 0)
            {
                fscanf(PDB,  "%s%d%s%s%s%s%s%s%s%s",  lixo,  &numres,  lixo,
lixo, lixo, lixo, lixo, lixo, lixo, ato);
            }
            else
            {
                fscanf(PDB, "%s%s%d%s%s%s%s%s%s%s%s", lixo, lixo, &numres,
lixo, lixo, lixo, lixo, lixo, lixo, lixo, ato);
            }
        }
        else
        {
            break;
        }
    }

    fclose(PDB);
    PDB = fopen(arqent, "r");
    tam = 0.0088*numres + 4.8634;

    while(TRUE)
    {
        fscanf(PDB, "%s", ato);

        if(strcmp(atom, ato) == 0)
        {
            break;
        }
        else
        {
            fscanf(PDB, "%[^\n]%*c", lixo);
        }
    }
```

```
    if(o == 0)
    {
        INP = fopen("input.inp", "w");
    }
    else
    {
        INP = fopen(arqsai, "w");
    }

    fprintf(INP,   "TITLE     Input   file   generated   by   GENINP.\n");
fprintf(INP, "%6d%6.2lf%6.2lf\n", numres, tam, ph);
    fscanf(PDB, "%s%s", lixo, lixo);
    if(strcmp(lixo,  "NH1AARG")  ==  0  ||  strcmp(lixo,  "NH1BARG")  ==  0  ||
strcmp(lixo, "NH2AARG") == 0 || strcmp(lixo, "NH2BARG") == 0)
    {
        size = strlen(lixo);
        strcpy(res, &lixo[size - 4]);
        fscanf(PDB, "%s%d%s%s%s%s", lixo, &cont, lixo, lixo, lixo, lixo);
    }
    else
    {
        fscanf(PDB, "%s%s%d%s%s%s%s", res, lixo, &cont, lixo, lixo, lixo,
lixo);
    }

    if(strcmp(lixo, "1.00") == 0 || strcmp(lixo, "0.50") == 0)
    {
        fscanf(PDB, "%s%s%s", lixo, lixo, ato);
    }
    else
    {
        fscanf(PDB, "%s%s", lixo, ato);
    }

    fprintf(INP, "%3s ", res);
    numres = cont;

    while(TRUE)
    {
        if(strcmp(atom, ato) != 0 && strcmp(anisou, ato) != 0)
        {
            break;
        }
        else if(strcmp(atom, ato) == 0)
        {
            fscanf(PDB, "%s%s", lixo, lixo);

            if(strcmp(lixo, "NH1AARG") == 0 || strcmp(lixo, "NH1BARG") ==
0 || strcmp(lixo, "NH2AARG") == 0 || strcmp(lixo, "NH2BARG") == 0)
            {
                size = strlen(lixo);
                strcpy(res, &lixo[size - 3]);
                fscanf(PDB, "%s%d%s%s%s%s", lixo, &cont, lixo, lixo, lixo,
lixo);
            }
            else
            {
                fscanf(PDB,  "%s%s%d%s%s%s%s",  res,  lixo,  &cont,  lixo,
lixo, lixo, lixo);
```

```
                if(strcmp(res, "BARG") == 0 || strcmp(res, "AARG") == 0)
                {
                    size = strlen(res);
                    strcpy(res, &res[size - 3]);
                }
            }

            if(strcmp(lixo, "1.00") == 0 || strcmp(lixo, "0.50") == 0)
            {
                fscanf(PDB, "%s%s%s", lixo, lixo, ato);
            }
            else
            {
                fscanf(PDB, "%s%s", lixo, ato);
            }
        }
        else if(strcmp(anisou, ato) == 0)
        {
            fscanf(PDB, "%s%s", lixo, lixo);

            if(strcmp(lixo, "NH1AARG") == 0 || strcmp(lixo, "NH1BARG") ==
0 || strcmp(lixo, "NH2AARG") == 0 || strcmp(lixo, "NH2BARG") == 0)
            {
                fscanf(PDB,  "%s%s%s%s%s%s%s%s%s%s",  lixo,  lixo,  lixo,
lixo, lixo, lixo, lixo, lixo, lixo, ato);
            }
            else
            {
                fscanf(PDB,  "%s%s%s%s%s%s%s%s%s%s%s",  lixo,  lixo,  lixo,
lixo, lixo, lixo, lixo, lixo, lixo, lixo, ato);
            }
        }

        if(cont == numres + 1)
        {
            fprintf(INP, "%3s ", res);
            numres = cont;
            if(cont%15 == 0)
            {
                fprintf(INP, "\n");
            }
        }

    }

    fclose(PDB);
    fclose(INP);

    printf("%s", exito);
}
```

APÊNDICE B – CÓDIGO CF_LRP

O código CF_LRP recebe o arquivo gerado pelo GENINP e, a partir dos dados contidos no arquivo, gera os arquivos com os potenciais de interação intermolecular e intramolecular para a simulação, a distribuição de cargas da biomolécula de acordo com a equação de Hendersson-Hasselbach, os arquivos com as instruções gerais da etapa de minimização de energia e da etapa de produção e os arquivos necessários para a solvatação e a adição de íons à caixa cúbica de simulação de dinâmica molecular.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define TRUE 1
#define PI 3.1415926535
#define MAX 200

float hidro(float a);
void main(int argc, char **argv)
{
    char *res[30]; /* Armazenamento das siglas de cada resíduo */
    char *arqent = ""; /* Arquivo de entrada */
    char ent[] = "-f"; /* String para comparação e identificação do
arquivo de entrada */
    char erroarg[] = "ERRO: Número de argumentos inconsistente.\n";
    char erroinp[] = "ERRO: Arquivo de entrada com a extensão
incorreta.\n";
    char erroent[] = "ERRO: Arquivo de entrada não especificado.\n";
    char exito[] = "Arquivos gerados com exito.";
    char verinp[] = ".inp";
```

```
    char verif[4]; /* String para verificação da extensão do arquivo
de entrada */
    char lido[MAX]; /* String para leitura */
    char lixo[MAX]; /* String para jogar informações inúteis fora */
    FILE *ATP; /* Arquivo de saída com os tipos de partículas do campo
de força */
    FILE *INP; /* Arquivo de entrada com os dados da proteína */
    FILE  *OUT;  /*  Arquivo  de  input  modificado  com  o  número  de
partículas de água sugerido */
    FILE *PTO; /* Arquivo de topologia da proteína - Protein Topology
*/
    FILE *TOP; /* Arquivo global de topologia */
    FILE *WAT; /* Arquivo com uma água para solvatar */
    FILE *WTO; /* Arquivo de topologia da água - Water topology */
    FILE *TAB; /* Arquivo com a tabela para o potencial LJ12-4 */
    FILE *MDP; /* Arquivo com os parâmetros da simulação: minimização
e ensemble NPT */
    FILE *NDX; /* Arquivo index para diferenciar as interações */
    float par[MAX][MAX]; /* Matriz para armazenar os dados físico-
químicos dos aminoácidos */
    float r; /* Parâmetros para cálcular as funções do potencial LJ
12-4 */
    float ph, tam; /* pH da simulação e o tamanho da caixa */
    float mas; /* Massa da proteína para sugerir a quantidade de
partículas de água */
    float masb; /* Massa da BSA */
    float agu; /* Massa de água */
    float conc; /* Concentração de proteína */
    float conv = (3./4.)*(1./PI); /* Fator de conversão do volume para
o cálculo do raio no sigma */
    float hid; /* Parâmetro de hidrofobicidade */
    float pos, neg, posr, negr, posres, negres; /* Contadores de carga
*/
    int cont, ind, sof; /* Contadores */
    int f = 0; /* Índice do arquivo de entrada */
    int compe; /* Comprimento do string do arquivo de entrada */
    int npa; /* Número de partículas de água */
    int numres; /* Quantidade de resíduos */
    int k; /* Variável para identificar o resíduo da proteína lido */
    int carga; /* Carga da proteína */


    neg = 0;
    pos = 0;
    posr = 0;
    negr = 0;
    posres = 0;
    negres = 0;

    if(argc%2 == 0)
    {
        printf("%s", erroarg);
        exit(EXIT_FAILURE);
    }

    for(cont = 0; cont <= 2; cont++)
    {
        if(strcmp(argv[cont], ent) == 0)
        {
            f = cont + 1;
            arqent = argv[f];
```

```
        }
    }

    if(f%2 != 0 || arqent == "" || f == 0)
    {
        printf("%s", erroent);
        exit(EXIT_FAILURE);
    }

    compe = strlen(arqent);

    if(strcmp(strcpy(verif, &arqent[compe - 4]), verinp) != 0)
    {
        printf("%s", erroinp);
        exit(EXIT_FAILURE);
    }

    INP = fopen(arqent, "r");
    fscanf(INP, "%s", lido);

    if(strcmp(lido, "TITLE") == 0)
    {
        fscanf(INP, "%[^\n]%*c", lixo);
    }
    else
    {
        fclose(INP);
        INP = fopen(arqent, "r");
    }

    fscanf(INP, "%d %f %f", &numres, &tam, &ph);

    /* Inicializando o vetor de siglas de resíduos */
    res[1] = "ALA";
    res[2] = "ARG";
    res[3] = "ASN";
    res[4] = "ASP";
    res[5] = "CYS";
    res[6] = "GLU";
    res[7] = "GLN";
    res[8] = "GLY";
    res[9] = "HIS";
    res[10] = "ILE";
    res[11] = "LEU";
    res[12] = "LYS";
    res[13] = "MET";
    res[14] = "PHE";
    res[15] = "PRO";
    res[16] = "SER";
    res[17] = "THR";
    res[18] = "TRP";
    res[19] = "TYR";
    res[20] = "VAL";
    res[21] = "H2O"; /* Água */
    res[22] = "CL-"; /* Cloreto */
    res[23] = "NA+"; /* Sódio */

    /* Os parâmetros de cada aminoácido serão colocados seguindo a
ordem do vetor res */
    /* Parâmetros para o resíduo Alanina */
    par[1][1] = 1.80;  /* Hidropatia */
```

```
    par[1][2] = 2.33;  /* pKA da carboxila */
    par[1][3] = 9.71;  /* pKA da amina */
    par[1][4] = NAN;   /* pKA da cadeia lateral */
    par[1][5] = 6.00;  /* Ponto isoelétrico */
    par[1][6] = 71.08; /* Massa molar do resíduo */
    par[1][7] = 0.092;  /* Volume em nm³ */
    par[1][8] = 2*cbrt(conv*par[1][7]); /* Sigma */
    par[1][9] = 2.47882; /* Epsilon */
    par[1][10]  =  NAN;  /*  Parâmetro  ni  da  equação  de  Henderson-
Hasselbach */

    /* Parâmetros para o resíduo Arginina */
    par[2][1] = -4.50;
    par[2][2] = 2.03;
    par[2][3] = 9.00;
    par[2][4] = 12.10;
    par[2][5] = 10.76;
    par[2][6] = 156.18;
    par[2][7] = 0.225;
    par[2][8] = 2*cbrt(conv*par[2][7]);
    par[2][9] = 2.47882;
    par[2][10] = 1;

    /* Parâmetros para o resíduo Asparagina */
    par[3][1] = -3.50;
    par[3][2] = 2.16;
    par[3][3] = 8.76;
    par[3][4] = NAN;
    par[3][5] = 5.41;
    par[3][6] = 114.10;
    par[3][7] = 0.135;
    par[3][8] = 2*cbrt(conv*par[3][7]);
    par[3][9] = 2.47882;
    par[3][10] = NAN;

    /* Parâmetros para o resíduo Ácido Aspártico */
    par[4][1] = -3.50;
    par[4][2] = 1.95;
    par[4][3] = 9.66;
    par[4][4] = 3.71;
    par[4][5] = 2.77;
    par[4][6] = 115.09;
    par[4][7] = 0.125;
    par[4][8] = 2*cbrt(conv*par[4][7]);
    par[4][9] = 2.47882;
    par[4][10] = -1;

    /* Parâmetros para o resíduo Cisteína */
    par[5][1] = 2.50;
    par[5][2] = 1.91;
    par[5][3] = 10.28;
    par[5][4] = 8.14;
    par[5][5] = 5.07;
    par[5][6] = 103.14;
    par[5][7] = 0.106;
    par[5][8] = 2*cbrt(conv*par[5][7]);
    par[5][9] = 2.47882;
    par[5][10] = -1;

    /* Parâmetros para o resíduo Ácido Glutâmico */
    par[6][1] = -3.50;
```

```
par[6][2] = 2.16;
par[6][3] = 9.58;
par[6][4] = 4.15;
par[6][5] = 3.22;
par[6][6] = 129.11;
par[6][7] = 0.155;
par[6][8] = 2*cbrt(conv*par[6][7]);
par[6][9] = 2.47882;
par[6][10] = -1;

/* Parâmetross para o resíduo Glutamina */
par[7][1] = -3.50;
par[7][2] = 2.18;
par[7][3] = 9.00;
par[7][4] = NAN;
par[7][5] = 5.65;
par[7][6] = 128.13;
par[7][7] = 0.161;
par[7][8] = 2*cbrt(conv*par[7][7]);
par[7][9] = 2.47882;
par[7][10] = NAN;

/* Parâmetros para o resíduo Glicina */
par[8][1] = -0.40;
par[8][2] = 2.34;
par[8][3] = 9.58;
par[8][4] = NAN;
par[8][5] = 5.97;
par[8][6] = 57.05;
par[8][7] = 0.066;
par[8][8] = 2*cbrt(conv*par[8][7]);
par[8][9] = 2.47882;
par[8][10] = NAN;

/* Parâmetros para o resíduo Histidina */
par[9][1] = -3.20;
par[9][2] = 1.70;
par[9][3] = 9.09;
par[9][4] = 6.04;
par[9][5] = 7.59;
par[9][6] = 137.14;
par[9][7] = 0.167;
par[9][8] = 2*cbrt(conv*par[9][7]);
par[9][9] = 2.47882;
par[9][10] = 1;

/* Parâmetros para o resíduo Isoleucina */
par[10][1] = 4.50;
par[10][2] = 2.26;
par[10][3] = 9.60;
par[10][4] = NAN;
par[10][5] = 6.02;
par[10][6] = 113.16;
par[10][7] = 0.169;
par[10][8] = 2*cbrt(conv*par[10][7]);
par[10][9] = 2.47882;
par[10][10] = NAN;

/* Parâmetros para o resíduo Leucina */
par[11][1] = 3.80;
par[11][2] = 2.32;
```

```
par[11][3] = 9.58;
par[11][4] = NAN;
par[11][5] = 5.98;
par[11][6] = 113.16;
par[11][7] = 0.168;
par[11][8] = 2*cbrt(conv*par[11][7]);
par[11][9] = 2.47882;
par[11][10] = NAN;

/* Parâmetros para o resíduo Lisina */
par[12][1] = -3.90;
par[12][2] = 2.15;
par[12][3] = 9.16;
par[12][4] = 10.67;
par[12][5] = 9.74;
par[12][6] = 128.17;
par[12][7] = 0.171;
par[12][8] = 2*cbrt(conv*par[12][7]);
par[12][9] = 2.47882;
par[12][10] = 1;

/* Parâmetros para o resíduo Metionina */
par[13][1] = 1.90;
par[13][2] = 2.16;
par[13][3] = 9.08;
par[13][4] = NAN;
par[13][5] = 5.74;
par[13][6] = 131.19;
par[13][7] = 0.171;
par[13][8] = 2*cbrt(conv*par[13][7]);
par[13][9] = 2.47882;
par[13][10] = NAN;

/* Parâmetros para o resíduo Fenilalanina */
par[14][1] = 2.80;
par[14][2] = 2.18;
par[14][3] = 9.09;
par[14][4] = NAN;
par[14][5] = 5.48;
par[14][6] = 147.17;
par[14][7] = 0.203;
par[14][8] = 2*cbrt(conv*par[14][7]);
par[14][9] = 2.47882;
par[14][10] = NAN;

/* Parâmetros para o resíduo Prolina */
par[15][1] = -1.60;
par[15][2] = 1.95;
par[15][3] = 10.47;
par[15][4] = NAN;
par[15][5] = 6.30;
par[15][6] = 97.11;
par[15][7] = 0.129;
par[15][8] = 2*cbrt(conv*par[15][7]);
par[15][9] = 2.47882;
par[15][10] = NAN;

/* Parâmetros para o resíduo Serina */
par[16][1] = -0.80;
par[16][2] = 2.13;
par[16][3] = 9.05;
```

```
par[16][4] = NAN;
par[16][5] = 5.68;
par[16][6] = 87.07;
par[16][7] = 0.099;
par[16][8] = 2*cbrt(conv*par[16][7]);
par[16][9] = 2.47882;
par[16][10] = NAN;

/* Parâmetros para o resíduo Treonina */
par[17][1] = -0.70;
par[17][2] = 2.20;
par[17][3] = 8.96;
par[17][4] = NAN;
par[17][5] = 5.60;
par[17][6] = 101.10;
par[17][7] = 0.122;
par[17][8] = 2*cbrt(conv*par[17][7]);
par[17][9] = 2.47882;
par[17][10] = NAN;

/* Parâmetros para o resíduo Triptofano */
par[18][1] = -0.90;
par[18][2] = 2.38;
par[18][3] = 9.34;
par[18][4] = NAN;
par[18][5] = 5.89;
par[18][6] = 186.21;
par[18][7] = 0.240;
par[18][8] = 2*cbrt(conv*par[18][7]);
par[18][9] = 2.47882;
par[18][10] = NAN;

/* Parâmetros para o resíduo Tirosina */
par[19][1] = -1.30;
par[19][2] = 2.24;
par[19][3] = 9.04;
par[19][4] = 10.10;
par[19][5] = 5.66;
par[19][6] = 163.17;
par[19][7] = 0.203;
par[19][8] = 2*cbrt(conv*par[19][7]);
par[19][9] = 2.47882;
par[19][10] = -1;

/* Parâmetros para o resíduo Valina */
par[20][1] = 4.20;
par[20][2] = 2.27;
par[20][3] = 9.52;
par[20][4] = NAN;
par[20][5] = 5.96;
par[20][6] = 99.13;
par[20][7] = 0.142;
par[20][8] = 2*cbrt(conv*par[20][7]);
par[20][9] = 2.47882;
par[20][10] = NAN;

/* Parâmetros para a água */
par[21][1] = 0;
par[21][2] = 0;
par[21][3] = 0;
par[21][4] = 0;
```

```
    par[21][5] = 0;
    par[21][6] = 54.0;
    par[21][7] = 0.4371;
    par[21][8] = 0.4371;
    par[21][9] = 3.74468;

    /* Parâmetros para o cloreto */
    par[22][1] = 0;
    par[22][2] = 0;
    par[22][3] = 0;
    par[22][4] = 0;
    par[22][5] = 0;
    par[22][6] = 35.45300;
    par[22][7] = 0.441724;
    par[22][8] = 0.441724;
    par[22][9] = 0.492833;

    /* Parâmetros para o sódio */
    par[23][1] = 0;
    par[23][2] = 0;
    par[23][3] = 0;
    par[23][4] = 0;
    par[23][5] = 0;
    par[23][6] = 22.98977;
    par[23][7] = 0.333045;
    par[23][8] = 0.333045;
    par[23][9] = 0.0115980;

    for(cont = 1; cont <= 20; cont++)
    {
        if(isnan(par[cont][10]) == 0) /* Se existir pKa, ou seja, se a
cadeia for ionizável */
        {
            if(par[cont][10] == 1)
            {
                par[cont][11] = pow(10, (par[cont][4] - ph)/1)/(1 +
pow(10, (par[cont][4] - ph)/1));
            }
            else if(par[cont][10] == -1)
            {
                par[cont][11] = (pow(10, (par[cont][4] - ph)/-1)/(1 +
pow(10, (par[cont][4] - ph)/-1)))*(-1);
            }

            par[cont][11] = round(par[cont][11]*1000)/1000;
        }
    }

    for(cont = 1; cont <= 20; cont++)
    {
        par[cont][12] = 0;
    }

    mas = 0;

    for(cont = 1; cont <= numres; cont++)
    {
        fscanf(INP, "%s", lido);
        for(ind = 1; ind <= 20; ind++)
        {
            k = strcmp(lido, res[ind]);
```

```
        if(k == 0)
        {
            par[ind][12] = par[ind][12] + 1;
            mas = mas + par[ind][6];
            break;
        }
    }

}

for(cont = 1; cont <= 20; cont++)
{
    if(par[cont][11] > 0)
    {
        pos = pos + par[cont][11]*par[cont][12];
    }
    else if(par[cont][11] < 0)
    {
        neg = neg + par[cont][11]*par[cont][12];
    }
}

ind = 1;
printf("\npos = %f  neg = %f\n\n", pos, neg);

while(pos - (int)pos >= 0.0001)
{
    posr = round(pos);
    printf("pos = %f  e  posr = %f\n", pos, posr);

    if(ind == 30)
    {
        if(pos - (int)pos > 0.5)
        {
            posres = 1 - (pos - (int)pos);
            break;
        }
        else
        {
            negres = -(pos - (int)pos);
            break;
        }
    }

    for(cont = 1; cont <= 20; cont++)
    {
        if(par[cont][11] > 0)
        {
           par[cont][11] = (posr/pos)*par[cont][11];
           par[cont][11] = round(par[cont][11]*1000)/1000;
        }
    }

    pos = 0;

    for(cont = 1; cont <= 20; cont++)
    {
        if(par[cont][11] > 0)
        {
            pos = pos + par[cont][11]*par[cont][12];
```

```
        }
    }

    ind = ind + 1;
}

ind = 1;

while(-neg + (int)neg >= 0.0001)
{
    negr = round(neg);
    printf("neg = %f  e  negr = %f\n", neg, negr);

    if(ind == 30)
    {
        if(-neg + (int)neg > 0.5)
        {
            negres = negres + 1 + neg - (int)neg;
            break;
        }
        else
        {
            posres = posres + -(neg - (int)neg);
            break;
        }
    }

    for(cont = 1; cont <= 20; cont++)
    {
        if(par[cont][11] < 0)
        {
            par[cont][11] = (negr/neg)*par[cont][11];
            par[cont][11] = round(par[cont][11]*1000)/1000;
        }
    }

    neg = 0;

    for(cont = 1; cont <= 20; cont++)
    {
        if(par[cont][11] < 0)
        {
            neg = neg + par[cont][11]*par[cont][12];
        }
    }

    ind = ind + 1;
}

fclose(INP);
INP = fopen(arqent, "r");
fscanf(INP, "%s", lido);

if(strcmp(lido, "TITLE") == 0)
{
    fscanf(INP, "%[^\n]%*c", lixo);
}
else
{
    fclose(INP);
    INP = fopen(arqent, "r");
```

```
    }

    fscanf(INP, "%d %f %f", &numres, &tam, &ph);

    ATP = fopen("atomtypes.itp", "w");
    fprintf(ATP, "; Conjunto de partículas do campo de forças\n\n");
    fprintf(ATP,    "[    defaults    ]\n1\t1\tyes\t0.5\t0.5\n\n");    /*
Explicação dos parâmetros: 1) Potencial LJ; 2) Regra de mistura; 3)
Gerar pares de interação 1-4; 4) Fator multiplicativo para os pares 1-
4 (VdW); 5) Fator multiplicativo para os pares 1-4 (eletrostático); */
    fprintf(ATP, "[ atomtypes ]\n");

    for(cont = 1; cont <= 23; cont++)
    {
        if(res[cont] != "H2O" && res[cont] != "CL-" && res[cont] !=
"NA+")
        {
            fprintf(ATP, "%3s     %3s%14.6f%7.3f  A  %12.6e  %12.6e\n",
res[cont],        res[cont],        par[cont][6],        par[cont][11],
(4*par[cont][9]*pow(par[cont][8],                                  6)),
(4*par[cont][9]*pow(par[cont][8], 12)));
            fprintf(ATP, "%3sC   %3sC%13.6f%7.3f  A  %12.6e  %12.6e\n",
res[cont], res[cont], par[cont][6], pow((par[cont][11] - 1 + negres),
2),               (4*par[cont][9]*pow(par[cont][8],                6)),
(4*par[cont][9]*pow(par[cont][8], 12))); /* Terminal Carboxila */
            fprintf(ATP, "%3sA   %3sA%13.6f%7.3f  A  %12.6e  %12.6e\n",
res[cont], res[cont], par[cont][6], pow((par[cont][11] + 1 + posres),
2),               (4*par[cont][9]*pow(par[cont][8],                6)),
(4*par[cont][9]*pow(par[cont][8], 12))); /* Terminal Amina */
        }
        else if(res[cont] == "H2O" || res[cont] == "CL-" || res[cont]
== "NA+")
        {
            if(res[cont] == "H2O")
            {
                fprintf(ATP, "%3s   %3s%14.6f%7.3f A %12.6e %12.6e\n",
res[cont],            res[cont],            par[cont][6],            0.,
((3*sqrt(3))/2)*par[cont][9]*pow(par[cont][8],                      4),
((3*sqrt(3))/2)*par[cont][9]*pow(par[cont][8], 12)); /* Água */
            }
            else if(res[cont] == "NA+" || res[cont] == "CL-")
            {
                fprintf(ATP, "%3s   %3s%14.6f%7.3f A %12.6e %12.6e\n",
res[cont],            res[cont],            par[cont][6],            1.,
(4*par[cont][9]*pow(par[cont][8],                                  6)),
(4*par[cont][9]*pow(par[cont][8], 12)));
            }
        }
    }

    fprintf(ATP, "\n\n[ nonbond_params ]\n");

    for(cont = 1; cont <= 23; cont++)
    {
        for(ind = cont; ind <= 23; ind++)
        {
            if((res[cont] == "H2O" || res[cont] == "CL-" || res[cont]
== "NA+") && (res[ind] == "H2O" || res[ind] == "CL-" || res[ind] ==
"NA+"))
            {
                if(res[cont] == "H2O" || res[ind] == "H2O")
```

```
                {
                    fprintf(ATP,  "%3s      %3s  1  %12.6e  %12.6e\n",
res[cont],                                                  res[ind],
((3*sqrt(3))/2)*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                 4),
((3*sqrt(3))/2)*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2), 12));
                }
                else
                {
                    fprintf(ATP,  "%3s      %3s  1  %12.6e  %12.6e\n",
res[cont],                                                  res[ind],
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]               +
par[ind][8])/2),                                                 6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]               +
par[ind][8])/2), 12)));
                }
            }
            else if(res[ind] == "H2O" || res[ind] == "CL-" || res[ind]
== "NA+")
            {
                if(res[ind] == "H2O")
                {
                    hid = hidro(par[cont][1]);
                    fprintf(ATP,  "%3s      %3s  1  %12.6e  %12.6e\n",
res[cont],                                                  res[ind],
((3*sqrt(3))/2)*((sqrt(par[cont][9]*par[ind][9]))/hid)*pow(((par[cont]
[8]                +                par[ind][8])/2),                4),
((3*sqrt(3))/2)*((sqrt(par[cont][9]*par[ind][9]))/hid)*pow(((par[cont]
[8] + par[ind][8])/2), 12));
                    fprintf(ATP,  "%3sC     %3s  1  %12.6e  %12.6e\n",
res[cont],                                                  res[ind],
((3*sqrt(3))/2)*((sqrt(par[cont][9]*par[ind][9]))/hid)*pow(((par[cont]
[8]                +                par[ind][8])/2),                4),
((3*sqrt(3))/2)*((sqrt(par[cont][9]*par[ind][9]))/hid)*pow(((par[cont]
[8] + par[ind][8])/2), 12));
                    fprintf(ATP,  "%3sA     %3s  1  %12.6e  %12.6e\n",
res[cont],                                                  res[ind],
((3*sqrt(3))/2)*((sqrt(par[cont][9]*par[ind][9]))/hid)*pow(((par[cont]
[8]                +                par[ind][8])/2),                4),
((3*sqrt(3))/2)*((sqrt(par[cont][9]*par[ind][9]))/hid)*pow(((par[cont]
[8] + par[ind][8])/2), 12));
                }
                else
                {
                    fprintf(ATP,  "%3s      %3s  1  %12.6e  %12.6e\n",
res[cont],                                                  res[ind],
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]               +
par[ind][8])/2),                                                 6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]               +
par[ind][8])/2), 12)));
                    fprintf(ATP,  "%3sC     %3s  1  %12.6e  %12.6e\n",
res[cont],                                                  res[ind],
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]               +
par[ind][8])/2),                                                 6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]               +
par[ind][8])/2), 12)));
                    fprintf(ATP,  "%3sA     %3s  1  %12.6e  %12.6e\n",
res[cont],                                                  res[ind],
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]               +
par[ind][8])/2),                                                 6)),
```

```
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                }
            }

            else if(res[cont] == res[ind])
            {
                fprintf(ATP, "%3s  %3s  1 %12.6e %12.6e\n", res[cont],
res[ind],   (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                  6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3s  %3sC 1 %12.6e %12.6e\n", res[cont],
res[ind],   (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                  6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3s  %3sA 1 %12.6e %12.6e\n", res[cont],
res[ind],   (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                  6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3sC %3sC 1 %12.6e %12.6e\n", res[cont],
res[ind],   (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                  6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3sC %3sA 1 %12.6e %12.6e\n", res[cont],
res[ind],   (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                  6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3sA %3sA 1 %12.6e %12.6e\n", res[cont],
res[ind],   (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                  6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
            }
            else
            {
                fprintf(ATP, "%3s  %3s  1 %12.6e %12.6e\n", res[cont],
res[ind],   (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                  6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3s  %3sC 1 %12.6e %12.6e\n", res[cont],
res[ind],   (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                  6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3s  %3sA 1 %12.6e %12.6e\n", res[cont],
res[ind],   (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                  6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3sC %3s  1 %12.6e %12.6e\n", res[cont],
res[ind],   (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]   +
par[ind][8])/2),                                                  6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
```

```
                fprintf(ATP, "%3sC %3sC 1 %12.6e %12.6e\n", res[cont],
res[ind],    (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]    +
par[ind][8])/2),                                                    6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3sC %3sA 1 %12.6e %12.6e\n", res[cont],
res[ind],    (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]    +
par[ind][8])/2),                                                    6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3sA %3s  1 %12.6e %12.6e\n", res[cont],
res[ind],    (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]    +
par[ind][8])/2),                                                    6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3sA %3sC 1 %12.6e %12.6e\n", res[cont],
res[ind],    (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]    +
par[ind][8])/2),                                                    6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
                fprintf(ATP, "%3sA %3sA 1 %12.6e %12.6e\n", res[cont],
res[ind],    (4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]    +
par[ind][8])/2),                                                    6)),
(4*(sqrt(par[cont][9]*par[ind][9]))*pow(((par[cont][8]                +
par[ind][8])/2), 12)));
            }
        }
    }

    fclose(ATP);

    WAT = fopen("water.gro", "w");
    fprintf(WAT, "Water\n     1\n     1H2O    H2O     1    3.000    3.000
3.000\n    6.00000   6.00000   6.00000");
    fclose(WAT);

    WAT = fopen("cloreto.gro", "w");
    fprintf(WAT, "Cloreto\n     1\n     1CL-    CL-     1    3.000    3.000
3.000\n    6.00000   6.00000   6.00000");
    fclose(WAT);

    WAT = fopen("sodio.gro", "w");
    fprintf(WAT, "Sódio\n     1\n     1NA+    NA+     1    3.000    3.000
3.000\n    6.00000   6.00000   6.00000");
    fclose(WAT);

    WTO = fopen("water.itp", "w");
    fprintf(WTO,  "[  moleculetype  ]\nH2O               1\n\n[  atoms
]\n\t1\tH2O\t1\tH2O\tH2O\t1\t0.000000\t54.000");
    fclose(WTO);

    WTO = fopen("ions.itp", "w");
    fprintf(WTO, "[ moleculetype ]\nCL-        1\n\n[ atoms ]\n\t1\tCL-
\t1\tCL-\tCL-\t1\t-1.000000\t35.453\n\n[      moleculetype      ]\nNA+
1\n\n[ atoms ]\n\t1\tNA+\t1\tNA+\tNA+\t1\t1.000000\t22.989");
    fclose(WTO);

    TAB = fopen("table.xvg", "w");
    fprintf(TAB, "#\n# Tabela com potencial LJ 12-4\n#\n");

    for(r = 0.0; r <= 3.0; r = r + 0.002)
```

```
    {
        if(r == 0.0)
        {
            fprintf(TAB, "%12.10e    %12.10e %12.10e    %12.10e %12.10e
%12.10e %12.10e\n", r, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
        }
        else
        {
            fprintf(TAB, "%12.10e    %12.10e %12.10e    %12.10e %12.10e
%12.10e %12.10e\n", r, 1/r, 1/pow(r, 2), -1/pow(r, 4), -4/pow(r, 5),
1/pow(r, 12), 12/pow(r, 13));
        }
    }

    fclose(TAB);

    TAB = fopen("table_Protein_Protein.xvg", "w");

    fprintf(TAB, "#\n# Tabela com potencial LJ 12-6\n#\n");

    for(r = 0.0; r <= 3.0; r = r + 0.002)
    {
        if(r == 0.0)
        {
            fprintf(TAB, "%12.10e    %12.10e %12.10e    %12.10e %12.10e
%12.10e %12.10e\n", r, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
        }
        else
        {
            fprintf(TAB, "%12.10e    %12.10e %12.10e    %12.10e %12.10e
%12.10e  %12.10e\n",  r,  1/r,  1/pow(r,2),  -1/pow(r,  6),  -6/pow(r,  7),
1/pow(r, 12), 12/pow(r, 13));
        }
    }

    fclose(TAB);

    TAB = fopen("table_Protein_Ion.xvg", "w");

    fprintf(TAB, "#\n# Tabela com potencial LJ 12-6\n#\n");

    for(r = 0.0; r <= 3.0; r = r + 0.002)
    {
        if(r == 0.0)
        {
            fprintf(TAB, "%12.10e    %12.10e %12.10e    %12.10e %12.10e
%12.10e %12.10e\n", r, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
        }
        else
        {
            fprintf(TAB, "%12.10e    %12.10e %12.10e    %12.10e %12.10e
%12.10e  %12.10e\n",  r,  1/r,  1/pow(r,2),  -1/pow(r,  6),  -6/pow(r,  7),
1/pow(r, 12), 12/pow(r, 13));
        }
    }

    fclose(TAB);

    TAB = fopen("table_Ion_Ion.xvg", "w");

    fprintf(TAB, "#\n# Tabela com potencial LJ 12-6\n#\n");
```

```
    for(r = 0.0; r <= 3.0; r = r + 0.002)
    {
        if(r == 0.0)
        {
            fprintf(TAB, "%12.10e    %12.10e %12.10e    %12.10e %12.10e
%12.10e %12.10e\n", r, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
        }
        else
        {
            fprintf(TAB, "%12.10e    %12.10e %12.10e    %12.10e %12.10e
%12.10e  %12.10e\n",  r,  1/r,  1/pow(r,2),  -1/pow(r,  6),  -6/pow(r,  7),
1/pow(r, 12), 12/pow(r, 13));
        }
    }

    fclose(TAB);

    PTO = fopen("protein.itp", "w");
    fprintf(PTO, "[ moleculetype ]\nPROCG\t1\n\n[ atoms ]\n");

    for(cont = 1; cont <= numres; cont++)
    {
        fscanf(INP, "%s", lido);
        for(ind = 1; ind <= 20; ind++)
        {
            k = strcmp(lido, res[ind]);

            if(k == 0)
            {
                k = ind;
                break;
            }
        }

        if(cont == 1) /* Terminal Amina */
        {
            fprintf(PTO,  "%3d     %sA%5d  PROCG     %sA%6d%8.3f%8.3f\n",
cont, res[k], cont, res[k], cont, par[k][11] + 1 + posres, par[k][6]);
        }
        else if(cont == numres) /* Terminal Carboxila */
        {
            fprintf(PTO,  "%3d     %sC%5d  PROCG     %sC%6d%8.3f%8.3f\n",
cont, res[k], cont, res[k], cont, par[k][11] - 1 + negres, par[k][6]);
        }
        else
        {
            fprintf(PTO,  "%3d    %s  %5d  PROCG    %s  %6d%8.3f%8.3f\n",
cont, res[k], cont, res[k], cont, par[k][11], par[k][6]);
        }
    }

    /* Parâmetros de ligação */
    fprintf(PTO, "\n[ bonds ]\n");
    for(cont = 1; cont <= numres - 1; cont++)
    {
        fprintf(PTO,  "%3d%5d%5d%10.5f%10.1f\n",  cont,  cont  +  1,  1,
0.8, 100000./2);
    }

    /* Parâmetros de ângulo */
```

```
    fprintf(PTO, "\n[ angles ]\n");
    for(cont = 1; cont <= numres - 2; cont++)
    {
        fprintf(PTO,  "%3d%5d%5d%5d%10.3f%10.3f\n",  cont  +  1,  cont,
cont + 2, 1, 180., 100./2);
    }

    fclose(PTO);
    fclose(INP);

    masb = 66463; /* Massa de uma molécula de BSA */
    agu = 10000*54;
    conc = masb/(masb + agu);
    agu = (mas/conc) - mas;
    npa = agu/54;
    carga = (pos + posres) + (neg + negres);
    TOP = fopen("topologia.top", "w");

    if(carga < 0)
    {
        fprintf(TOP,        "#include       \"atomtypes.itp\"\n#include
\"protein.itp\"\n#include   \"water.itp\"\n#include   \"ions.itp\"\n\n[
system ]\nPROCG in water\n\n[ molecules ]\nPROCG%5d\nH2O%7d\n", 1, npa
+ carga);
        fprintf(TOP, "NA+%7d", -carga);
    }
    else if(carga > 0)
    {
        fprintf(TOP,        "#include       \"atomtypes.itp\"\n#include
\"protein.itp\"\n#include   \"water.itp\"\n#include   \"ions.itp\"\n\n[
system ]\nPROCG in water\n\n[ molecules ]\nPROCG%5d\nH2O%7d\n", 1, npa
- carga);
        fprintf(TOP, "CL-%7d", carga);
    }

    fclose(TOP);

    INP = fopen("protein.itp", "r");
    OUT = fopen("temp.inp", "w");

    fscanf(INP, "%[^\n]%*c", lido);
    fscanf(INP, "%[^\n]%*c", lido);
    fscanf(INP, "%s%s%s", lido, lido, lido);

    for(cont = 1; cont <= numres; cont++)
    {
        fscanf(INP, "%s%s%[^\n]%*c", lixo, lido, lixo);
        fprintf(OUT, "%s ", lido);

        if(cont%15 == 0)
        {
            fprintf(OUT, "\n");
        }
    }

    fclose(OUT);
    fclose(INP);

    remove(arqent);
    OUT = fopen("temp.inp", "r");
    INP = fopen(arqent, "w");
```

```
    fprintf(INP,     "TITLE          Input     file     generated     by
CF_LRP.\n%6d%6.2f%6.2f%6d%6d\n", numres, tam, ph, npa, carga);

    for(cont = 1; cont <= numres; cont++)
    {
        fscanf(OUT, "%s", lido);
        fprintf(INP, "%s ", lido);

        if(cont%15 == 0)
        {
            fprintf(INP, "\n");
        }
    }

    fclose(OUT);
    fclose(INP);
    remove("temp.inp");


    MDP = fopen("minim.mdp", "w");
    fprintf(MDP,  ";Molecular  dynamics  parameter  file  generated  by
CF_LRP\n;These  parameters  are  destined  to  the  energy  minimization
step\n\n");
    fprintf(MDP,     "title\t\t\t=     Energy     minimization\n\n;Run
control\nintegrator\t\t=      steep\ntinit\t\t\t=      0\nnsteps\t\t\t=
50000\ninit-step\t\t=      0\n\n;Energy      minimization\nemtol\t\t\t=
1000.0\nemstep\t\t\t=      0.1\n\n;Output      control\nnstxout\t\t\t=
500\nnstvout\t\t\t= 500\nnstfout\t\t\t= 500\n");
    fprintf(MDP,  "nstlog\t\t\t=  500\nenergygrps\t\t=  Protein  Water
Ion\n\n;Neighbor  searching\ncutoff-scheme\t\t=  group\nnstlist\t\t\t=
1\nns-type\t\t\t=          grid\npbc\t\t\t=          xyz\nrlist\t\t\t=
1.0\n\n;Electrostatics\ncoulombtype\t\t=            PME\nrcoulomb\t\t=
1.0\nepsilon-r\t\t=   78.41\n\n;VdW\nvdwtype\t\t\t=   User\nrvdw\t\t\t=
1.0\n\n;Tables\nenergygrp-table\t\t= Protein Protein Ion Ion Protein
Ion\n\n;Temperature coupling\n");
    fprintf(MDP,    "tcoupl\t\t\t=    no\ntc-grps\t\t\t=    system\ntau-
t\t\t\t=  0.1\nref-t\t\t\t=  300\n\n;Pressure  coupling\npcoupl\t\t\t=
no\npcoupltype\t\t=  isotropic\ntau-p\t\t\t=  2.0\ncompressibility\t\t=
4.5e-5\nref-p\t\t\t=    1.0\n\n;Velocity    generation\ngen-vel\t\t\t=
no\ngen-temp\t\t= 300\ngen-seed\t\t= -1\n\n;Bonds\n");
    fprintf(MDP,     "constraints\t\t=    none\nconstraint-algorithm\t=
LINCS\ncontinuation\t\t= no\nlincs-order\t\t= 4\n");
    fclose(MDP);

    MDP = fopen("npt.mdp", "w");
    fprintf(MDP,  ";Molecular  dynamics  parameter  file  generated  by
CF_LRP\n;These  parameters  are  destined  to  the  NPT  ensemble
simulation\n\n;Run      control\nintegrator\t\t=      md\ntinit\t\t\t=
0\ndt\t\t\t=        0.002\nnsteps\t\t\t=        5000000\ncomm-mode\t\t=
linear\nnstcomm\t\t\t=      10\n\n;Output      control\nnstxout\t\t\t=
500\nnstvout\t\t\t= 500\nnstfout\t\t\t= 500\n");
    fprintf(MDP,           "nstlog\t\t\t=           500\nnstenergy\t\t=
500\nenergygrps\t\t= Protein Water Ion\n\n;Neighbor searching\ncutoff-
scheme\t\t=  group\nnstlist\t\t\t=  1\nns-type\t\t\t=  grid\npbc\t\t\t=
xyz\nrlist\t\t\t=              1.5\n\n;Electrostatics\ncoulombtype\t\t=
user\nrcoulomb\t\t=  1.5\nepsilon-r\t\t=  78.41\n\n;VdW\nvdwtype\t\t\t=
user\nrvdw\t\t\t= 1.5\n\n;Tables\nenergygrp-table\t\t= Protein Protein
Ion        Ion        Protein        Ion\n\n;Ewald\nfourierspacing\t\t=
0.16\noptimize_fft\t\t= yes\n\n;Temperature coupling\ntcoupl\t\t\t= v-
rescale\n");
```

```
    fprintf(MDP,    "tc-grps\t\t\t=    system\ntau-t\t\t\t=    0.1\nref-
t\t\t\t=            300\n\n;Pressure            coupling\npcoupl\t\t\t=
berendsen\npcoupltype\t\t=                      isotropic\ntau-p\t\t\t=
2.0\ncompressibility\t\t=     4.5e-5\nref-p\t\t\t=     1.0\n\n;Velocity
generation\ngen-vel\t\t\t=   yes\ngen-temp\t\t=   300\ngen-seed\t\t=   -
1\n\n;Bonds\nconstraints\t\t= none\n");
    fprintf(MDP,    "constraint-algorithm\t=    LINCS\ncontinuation\t\t=
no\nlincs-order\t\t= 4\n");
    fclose(MDP);
    NDX = fopen("index.ndx", "w");
    fprintf(NDX, "[ Protein ]\n");
    for(cont = 1; cont <= numres; cont++)
    {
        fprintf(NDX, "%6d", cont);

        if(cont%15 == 0)
        {
            fprintf(NDX, "\n");
        }
    }

    fprintf(NDX, "\n\n[ Water ]\n");
    ind = 1;

    for(cont = numres + 1; cont <= numres + npa - abs(carga); cont++)
    {
        fprintf(NDX, "%6d", cont);

        if(ind%15 == 0)
        {
            fprintf(NDX, "\n");
        }

        ind = ind + 1;
    }
    fprintf(NDX, "\n\n[ Ion ]\n");
    ind = 1;

    for(cont = numres + npa - abs(carga) + 1; cont <= npa + numres;
cont++)
    {
        fprintf(NDX, "%6d", cont);

        if(ind%15 == 0)
        {
            fprintf(NDX, "\n");
        }

        ind = ind + 1;
    }

    fprintf(NDX, "\n\n[ System ]\n");
    ind = 1;

    for(cont = 1; cont <= numres + npa; cont++)
    {
        fprintf(NDX, "%6d", cont);

        if(ind%15 == 0)
        {
            fprintf(NDX, "\n");
```

```
        }

        ind = ind + 1;
    }

    fprintf(NDX, "\n");
    fclose(NDX);
    printf("\n%s\n", exito);
}

float hidro(float a)
{
    float b; /* Parâmetro calculado */

    if(a < -1)
    {
        b = 1;
    }
    else if(a > 1)
    {
        b = 2;
    }
    else
    {
        b = 1;
    }
    return b;
}
```

APÊNDICE C – CÓDIGO STR

O código STR recebe o arquivo *input* gerado pelo GENINP modificado pelo CF_LRP. A modificação consiste na inserção da quantidade de partículas de água necessárias para a concentração desejada e na inserção da quantidade de íons necessários para a caixa cúbica de simulação de dinâmica molecular possuir carga líquida nula. O STR coloca a proteína dentro da caixa cúbica de simulação de dinâmica molecular na estrutura inicial desejada e solvata a proteína. Ao final, ele adiciona os íons necessários dependendo da carga total da proteína.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define TRUE 1
#define MAX 100
#define PI 3.141592

void lista(int a, char *b); /* Função para gerar a lista de comandos
de solvatação */
void listaclor(int a, char *b); /* Função para gerar a lista de
comandos para colocação dos íons cloreto */
void listasodi(int a, char *b); /* Função para gerar a lista de
comandos para colocação dos íons sódio */


void main(int argc, char **argv)
```

```
{
    char ent[] = "-f"; /* Comparação para identificar onde está o
arquivo de entrada */
    char *arqent; /* Nome do arquivo de entrada */
    char sai[] = "-o"; /* Comparação para identificar onde está o
arquivo de saída */
    char *arqsai; /* Nome do arquivo de saída */
    char *res[30]; /* Vetor para guardar as abreviações de todos os
possíveis resíduos */
    char verif[4]; /* String para verificação da extensão do arquivo
*/
    char verinp[] = ".inp"; /* Verificador da extensão .inp */
    char vergro[] = ".gro"; /* Verificador da extensão .gro */
    char erroarq[] = "ERRO: Arquivo de entrada não especificado.\n";
    char erroinp[] = "ERRO: Extensão do arquivo de entrada
incorreta.\n";
    char erronum[] = "ERRO: A opção fornecida não é adequada.\n";
    char erroarg[] = "ERRO: Número de argumentos inconsistente.\n";
    char lido[MAX]; /* String para leitura */
    char lixo[MAX]; /* String para descarte de informações */
    char command[MAX]; /* String para o comando no gromacs */
    char ultag[MAX]; /* Última água */
    FILE *INP; /* Arquivo de input */
    FILE *OUT; /* Arquivo de saída */
    FILE *LIS; /* Lista de comandos */
    int cont; /* Contador geral */
    int compe, comps; /* Variáveis para medir o comprimento das
strings dos arquivos de entrada e da saída */
    int sto; /* Controle de parada nos loops while */
    int f, o; /* Variáveis para identificação da entrada e da saída */
    int opc; /* Guarda a opção escolhida pelo usuário para a estrutura
da proteína */
    int numres, npa; /* Número de resíduos e número de partículas de
água */
    int carga; /* Íons necessários para a neutralização da caixa */
    float tam, ph; /* Tamanho da caixa e pH */

    arqent = "";
    arqsai = "";
    f = 0;
    o = 0;

    /* Sistema de proteção contra entradas incorretas e loop para
leitura do arquivo de entrada e de saída */
    if(argc%2 == 0 || argc == 1)
    {
        printf("%s", erroarg);
        exit(EXIT_FAILURE);
    }

    for(cont = 0; cont < argc; cont++)
    {
        if(strcmp(argv[cont], ent) == 0)
        {
            f = cont + 1;
            arqent = argv[f];
        }
        if(strcmp(argv[cont], sai) == 0)
        {
            o = cont + 1;
            arqsai = argv[o];
```

```
        }
    }

    if(f%2 != 0 || arqent == "" || strcmp(arqent, sai) == 0 || f == 0)
    {
        printf("%s", erroarq);
        exit(EXIT_FAILURE);
    }

    compe = strlen(arqent);
    comps = strlen(arqsai);

    /* Teste da extensão dos arquivos de entrada e de saída */
    if(strcmp(strcpy(verif, &arqent[compe - 4]), verinp) != 0)
    {
        printf("%s", erroinp);
        exit(EXIT_FAILURE);
    }
    else if(strcmp(arqsai, "") != 0)
    {
        if(strcmp(strcpy(verif, &arqsai[comps - 4]), vergro) != 0)
        {
            strcat(arqsai, vergro);
        }
    }
    else if(strcmp(arqsai, "") == 0)
    {
        arqsai = "output.gro";
    }

    /* Preenchendo o vetor com os nomes dos resíduos */

    res[1] = "ALA";
    res[2] = "ARG";
    res[3] = "ASN";
    res[4] = "ASP";
    res[5] = "CYS";
    res[6] = "GLU";
    res[7] = "GLN";
    res[8] = "GLY";
    res[9] = "HIS";
    res[10] = "ILE";
    res[11] = "LEU";
    res[12] = "LYS";
    res[13] = "MET";
    res[14] = "PHE";
    res[15] = "PRO";
    res[16] = "SER";
    res[17] = "THR";
    res[18] = "TRP";
    res[19] = "TYR";
    res[20] = "VAL";
    res[21] = "H2O";
    res[22] = "CL-";
    res[23] = "NA+";

    printf("Estruturas disponiveis:\n\n1 - Helicoide\n2 - Intestino\n3
- Zig zag\n\nEstrutura escolhida: ");
    scanf("%d", &opc);

    /* Sistema de proteção a entradas incorretas */
```

```
    if(isnan(opc) != 0 || opc > 3 || opc < 1)
    {
        printf("%s", erronum);
        exit(EXIT_FAILURE);
    }
    INP = fopen(arqent, "r");
    OUT = fopen("temp.gro", "w");

    fscanf(INP, "%s", lido);
    if(strcmp(lido, "TITLE") == 0)
    {
        fclose(INP);
        INP = fopen(arqent, "r");
        fscanf(INP, "%[^\n]%*c", lixo);
        fscanf(INP,  "%d  %f  %f  %d  %d",  &numres,  &tam,  &ph,  &npa,
&carga);
    }
    else
    {
        fclose(INP);
        INP = fopen(arqent, "r");
        fscanf(INP, "%d %f %f %d", &numres, &tam, &ph, &npa);
    }

    fprintf(OUT, "Protein coarse-grained\n%5d\n", numres);

    if(opc == 1)
    {
        float X, Y, Z; /* Coordenadas dos resíduos */
        float phi, dphi; /* Ângulo e incremento do ângulo */
        float delta; /* Incremento de altura */

        Z = tam/2.0 - tam/3.0;
        phi = 0.0;
        dphi = 0.15;
        delta = 0.012;


        for(cont = 1; cont <= numres; cont++)
        {
            X = (tam/2.0) + ((tam/2.0) - 1.0)*cos(phi);
            Y = (tam/2.0) + ((tam/2.0) - 1.0)*sin(phi);

            fscanf(INP, "%s", lido);
            fprintf(OUT, "%5d%-5s%5s%5d%8.3f%8.3f%8.3f\n", 1, "PROCG",
lido, cont, X, Y, Z);

            Z = Z + delta;
            phi = phi + dphi;
        }
    }
    else if(opc == 2)
    {
        float X, Y, Z; /* Coordenadas dos resíduos */
        int phi, delta; /* Espaçamento angular e incremento */
        int traj, k, vol; /* Controle da trajetória */

        phi = 0.;
        delta = 10;
        traj = 1;
        vol = 1;
```

```
        k = 0;
        X = ((tam/2.0) - 1.0)*cos(phi/100.0) + (tam/2.0);
        Y = ((tam/2.0) - 1.0)*sin(phi/100.0) + (tam/2.0);
        Z = 0.5;

        for(cont = 1; cont <= numres; cont++)
        {
            fscanf(INP, "%s", lido);
            fprintf(OUT, "%5d%-5s%5s%5d%8.3f%8.3f%8.3f\n", 1, "PROCG",
lido, cont, X, Y, Z);

            if(traj == 1 && phi == 0) /* Mudança para trajetória reta
*/
            {
                traj = 2;
                k = cont;
                vol = 1;
            }
            else if(traj == 1 && phi == 500) /* Mudança para
trajetória reta */
            {
                traj = 2;
                k = cont;
                vol = -1;
            }
            else if(traj == 2 && cont == k + 4) /* Mudança para
trajetória curva */
            {
                traj = 1;
            }

            if(traj == 1)
            {
                phi = phi + vol*delta;
                X = ((tam/2.0) - 1.0)*cos(phi/100.0) + (tam/2.0);
                Y = ((tam/2.0) - 1.0)*sin(phi/100.0) + (tam/2.0);
            }

            if(traj == 2)
            {
                Z = Z + 0.2;
            }
        }

    }
    else if(opc == 3)
    {
        float X, Y, Z; /* Coordenadas dos resíduos */
        int traj, k, vol, ext1; /* Controle da trajetória */

        X = 1.0;
        Y = 1.0;
        Z = 1.0;
        traj = 1;
        vol = 1;

        for(cont = 1; cont <= numres; cont++)
        {
            fscanf(INP, "%s", lido);
            fprintf(OUT, "%5d%-5s%5s%5d%8.3f%8.3f%8.3f\n", 1, "PROCG",
lido, cont, X, Y, Z);
```

```
        if(traj == 1 && (tam - Y < 1.0 || Y < 1.0))
        {
            traj = 2;
            k = cont;

            if(vol == 1)
            {
                vol = -1;
            }
            else if(vol == -1)
            {
                vol = 1;
            }
        }
        else if(traj == 2 && cont == k + 4)
        {
            traj = 1;
        }

        if(traj == 1)
        {
            Y = Y + vol*0.2;
        }
        else if(traj == 2)
        {
            X = X + (0.2/sqrt(2.0));
            Z = Z + (0.2/sqrt(2.0));
        }
    }
}

fprintf(OUT, "%10.5lf%10.5lf%10.5lf\n", tam, tam ,tam);
fclose(OUT);
fclose(INP);

strcpy(command, "editconf -f temp.gro -o ");
strcat(command, arqsai);
system(command);

remove("temp.gro");

/* Sessão para solvatar a molécula de proteína coarse grained */
if(carga > 0)
{
    lista(npa - carga, arqsai);
}
else if (carga < 0)
{
    lista(npa + carga, arqsai);
}
LIS = fopen("lista.txt", "r");

for(cont = 1; cont <= 2*npa; cont++)
{
    fscanf(LIS, "%[^\n]%*c", command);
    system(command);
}

fclose(LIS);
remove("lista.txt");
```

```
    if(carga > 0)
    {
        listaclor(carga, arqsai);
    }
    else if(carga < 0)
    {
        listasodi(carga, arqsai);
        carga = -carga;
    }

    LIS = fopen("lista.txt", "r");

    for(cont = 1; cont <= 2*carga; cont++)
    {
        fscanf(LIS, "%[^\n]%*c", command);
        system(command);
    }

    fclose(LIS);
    remove("lista.txt");
}

void lista(int a, char *b)
{
    int cont; /* Contador para o comando */
    FILE *SAI; /* Arquivo de saída para lista */

    SAI = fopen("lista.txt", "w");
    fprintf(SAI,  "gmx  insert-molecules  -f  %s  -ci  water.gro  -o
solv%d.gro -nmol 1 -allpair -try 1000000\nrm %s\n", b, 1, b);

    for(cont = 1; cont < a - 1; cont ++)
    {
        fprintf(SAI, "gmx insert-molecules -f solv%d.gro -ci water.gro
-o solv%d.gro -nmol 1 -allpair -try 1000000\nrm solv%d.gro\n", cont,
cont + 1, cont);
    }

    fprintf(SAI, "gmx insert-molecules -f solv%d.gro -ci water.gro -o
%s -nmol 1 -allpair -try 1000000\nrm solv%d.gro\n", cont, b, cont);

    fclose(SAI);
}

void listaclor(int a, char *b)
{
    int cont; /* Contador para o comando */
    FILE *SAI; /* Arquivo de saída para lista */

    SAI = fopen("lista.txt", "w");
    fprintf(SAI,  "gmx  insert-molecules  -f  %s  -ci  cloreto.gro  -o
solv%d.gro -nmol 1 -allpair -try 1000000\nrm %s\n", b, 1, b);

    for(cont = 1; cont < a - 1; cont ++)
    {
        fprintf(SAI,   "gmx   insert-molecules   -f   solv%d.gro   -ci
cloreto.gro   -o   solv%d.gro   -nmol   1   -allpair   -try   1000000\nrm
solv%d.gro\n", cont, cont + 1, cont);
    }
```

```
    fprintf(SAI, "gmx insert-molecules -f solv%d.gro -ci cloreto.gro -
o %s -nmol 1 -allpair -try 1000000\nrm solv%d.gro\n", cont, b, cont);

    fclose(SAI);
}

void listasodi(int a, char *b)
{
    int cont; /* Contador para o comando */
    FILE *SAI; /* Arquivo de saída para lista */

    SAI = fopen("lista.txt", "w");
    a = -a;
    fprintf(SAI,  "gmx  insert-molecules  -f  %s  -ci  sodio.gro  -o
solv%d.gro -nmol 1 -allpair -try 1000000\nrm %s\n", b, 1, b);

    for(cont = 1; cont < a - 1; cont ++)
    {
        fprintf(SAI, "gmx insert-molecules -f solv%d.gro -ci sodio.gro
-o solv%d.gro -nmol 1 -allpair -try 1000000\nrm solv%d.gro\n", cont,
cont + 1, cont);
    }

    fprintf(SAI, "gmx insert-molecules -f solv%d.gro -ci sodio.gro -o
%s -nmol 1 -allpair -try 1000000\nrm solv%d.gro\n", cont, b, cont);

    fclose(SAI);
}
```

APÊNDICE D – CÓDIGO TRJ

O código TRJ é o código de processamento dos dados depois da simulação de dinâmica molecular concluída. O código é responsável pela partição da trajetória em picossegundos e pela ativação do *software* CRYSOL para o cálculo da curva de intensidade de espalhamento de raios-X.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define MAX 200
#define TRUE 1

void trajtxt(char *nometpr);
void trajpcg(char *nometpr);

void main(int argc, char **argv)
{
    char ent[] = "-f"; /* Sinalizador do arquivo de entrada -
trajetória atomística*/
    char *arqent; /* Nome do arquivo de entrada: Será a trajetória do
arquivo da simulação atomistica */
    char tpr[] = "-t"; /* Sinalizador do arquivo tpr atomístico */
    char *arqtpr; /* Nome do arquivo tpr */
    char trr[] = "-s"; /* Sinlizador do arquivo de entrada -
trajetória CG */
    char *arqtrr; /* Nome do arquivo de trajetória CG */
    char tpc[] = "-e"; /* Sinalizador do arquivo tpr CG */
    char *arqtpc; /* Nome do arquivo tpr CG */
    char arqpdb[MAX]; /* Nome do arquivo pdb */
    char arqgro[MAX]; /* String usada para o nome do arquivo gro */
```

```
    char vertrr[] = ".trr"; /* Verificador para a extensão do arquivo
de entrada */
    char vertpr[] = ".tpr"; /* Verificador para a extensão do arquivo
gro */
    char verpdb[] = ".pdb"; /* Verificador para a extensão do arquivo
pdb */
    char verinp[] = ".inp"; /* Verificador para a extensão do arquivo
inp */
    char verif[4]; /* String para verificação da extensão do arquivo
*/
    char erroarg[] = "ERRO: Número de argumentos inconsistente.\n"; /*
Erro de argumentos inconsistentes */
    char erroinp[] = "ERRO: Extensão do arquivo de entrada
incorreta.\n";
    char command[MAX]; /* String para guardar comandos que serão dados
ao sistema pela função system */
    char lixo[MAX]; /* String para jogar informações fora */
    char res[3]; /* Resíduo */
    double x, y, z; /* Coordenadas lidas no .gro */
    double des; /* Informações descartadas */
    double arm, jot; /* Variável para armazenar o dado lido no .fit */
    double med[MAX][MAX]; /* Matriz para as médias do fator de forma
*/
    int f, e, t, s; /* Indices de entrada e saída de arquivos */
    int cont, ador, i; /* Contador para loops em geral */
    int compe, compg, comps, compt, compi; /* Variáveis para
armazenamento do tamanho dos nomes dos arquivos */
    int numres; /* Número de resíduos */
    FILE *LIS; /* Lista de comandos */
    FILE *GRO; /* Arquivo gro */
    FILE *PDB; /* Arquivo pdb */
    FILE *FIT; /* Arquivo fit */

    arqent = "";
    arqtpr = "";
    arqtrr = "";
    arqtpc = "";
    f = 0;
    t = 0;
    s = 0;
    e = 0;

    for(cont = 0; cont < MAX; cont++)
    {
        for(ador = 0; ador < MAX; ador++)
        {
            med[cont][ador] = 0;
        }
    }

    /* Sistema de proteção contra entradas incorretas e loop para
leitura do arquivo de entrada e de saída */
    if(argc%2 == 0 || argc == 1)
    {
        printf("%s", erroarg);
        exit(EXIT_FAILURE);
    }

    for(cont = 0; cont < argc; cont++)
    {
        if(strcmp(argv[cont], ent) == 0)
```

```
        {
            f = cont + 1;
            arqent = argv[f];
        }
        if(strcmp(argv[cont], tpr) == 0)
        {
            t = cont + 1;
            arqtpr = argv[t];
        }
        if(strcmp(argv[cont], trr) == 0)
        {
            s = cont + 1;
            arqtrr = argv[s];
        }
        if(strcmp(argv[cont], tpc) == 0)
        {
            e = cont + 1;
            arqtpc = argv[e];
        }
    }

    compe = strlen(arqent);
    compg = strlen(arqtpr);
    comps = strlen(arqtrr);
    compt = strlen(arqtpc);

    if(strcmp(strcpy(verif, &arqent[compe - 4]), vertrr) != 0)
    {
        printf("%s", erroinp);
        exit(EXIT_FAILURE);
    }
    if(strcmp(strcpy(verif, &arqtpr[compg - 4]), vertpr) != 0)
    {
        printf("%s", erroinp);
        exit(EXIT_FAILURE);
    }
    if(strcmp(strcpy(verif, &arqtrr[comps - 4]), vertrr) != 0)
    {
        printf("%s", erroinp);
        exit(EXIT_FAILURE);
    }
    if(strcmp(strcpy(verif, &arqtpc[compt - 4]), vertpr) != 0)
    {
        printf("%s", erroinp);
        exit(EXIT_FAILURE);
    }

    /* Fim da seção de leitura dos arquivos de entrada e saída */

    strcpy(command, "echo 1 | gmx trjconv -s ");
    strcat(command, arqtpr);
    strcat(command, " -f ");
    strcat(command, arqent);
    strcat(command, " -pbc atom -o traj_prot.trr");

    system(command); /* Corrige a trajetória e isola a proteína */

    trajtxt(arqtpr);

    LIS = fopen("traj.txt", "r");
```

```
for(cont = 1501; cont <= 2000; cont ++)
{
    fscanf(LIS, "%[^\n]%*c", command);
    system(command);
}

fclose(LIS);
remove("traj.txt");

system("crysol *.pdb exp.dat -cst -sm 2.0 -ns 100");

LIS = fopen("del.txt", "r");

for(cont = 0; cont < 500; cont ++)
{
    fscanf(LIS, "%[^\n]%*c", command);
    system(command);
}

fclose(LIS);
remove("del.txt");

LIS = fopen("arqint.txt", "r");

for(cont = 0; cont < 500; cont ++)
{
    fscanf(LIS, "%[^\n]%*c", command);
    FIT = fopen(command, "r");
    fscanf(FIT, "%[^\n]%*c", lixo);
    fscanf(FIT, "%[^\n]%*c", lixo);
    fscanf(FIT, "%[^\n]%*c", lixo);
    fscanf(FIT, "%[^\n]%*c", lixo);
    fscanf(FIT, "%[^\n]%*c", lixo);
    fscanf(FIT, "%[^\n]%*c", lixo);
    fscanf(FIT, "%[^\n]%*c", lixo);

    for(ador = 0; ador < 54; ador ++)
    {
        for(i = 0; i < 4; i ++)
        {
            fscanf(FIT, "%lf", &arm);
            med[ador][i] = med[ador][i] + arm;
        }
    }

    fclose(FIT);
    remove(command);
}

fclose(LIS);
remove("arqint.txt");

for(cont = 0; cont < 104; cont ++)
{
    for(ador = 0; ador < 4; ador ++)
    {
        med[cont][ador] = med[cont][ador]/500;
    }
}

LIS = fopen("media.dat", "w");
```

```
    fprintf(LIS, "Dados da simulação atomística\n");

    for(cont = 0; cont < 54; cont++)
    {
        for(ador = 0; ador < 4; ador++)
        {
            fprintf(LIS, "  %8.6lE", med[cont][ador]);

            if(ador == 3)
            {
                fprintf(LIS, "\n");
            }
        }
    }

    fclose(LIS);
    remove("crysol_summary.txt");
    remove("delfit.txt");
    remove("traj_prot.trr");

    /* Seção para calcular os fatores de forma da simulação Coarse-
Grained */

    strcpy(command, "echo 2 | gmx trjconv -s ");
    strcat(command, arqtpc);
    strcat(command, " -f ");
    strcat(command, arqtrr);
    strcat(command, " -pbc atom -o traj_protcg.trr");

    system(command); /* Corrije a trajetória e isola a proteína em um
arquivo de trajetória único */

    trajpcg(arqtpc);
    LIS = fopen("traj.txt", "r");

    for(cont = 1501; cont <= 10000; cont ++)
    {
        fscanf(LIS, "%[^\n]%*c", command);
        system(command);
        sprintf(arqgro, "traj%d.gro", cont);
        GRO = fopen(arqgro, "r");
        sprintf(arqpdb, "traj%d.pdb", cont);
        PDB = fopen(arqpdb, "w");
        fscanf(GRO, "%[^\n]%*c", command);
        fprintf(PDB, "COMPND    %s\n", command);
        fprintf(PDB, "AUTHOR    TRJ_RLP\n");
        fscanf(GRO, "%d", &numres);

        for(ador = 1; ador <= numres; ador++)
        {
            fscanf(GRO, "%s %s %s %lf %lf %lf %lf %lf %lf", lixo, res,
lixo, &x, &y, &z, &des, &des, &des); /* Lendo uma linha do .gro */
            fprintf(PDB,   "ATOM   %5d   C                 PRO   A%4d
%8.3f%8.3f%8.3f%6.2f%6.2f\n", ador, ador, 10*x, 10*y, 10*z, 1., 0.);
/* Escrevendo uma linha do .pdb */
        }
        fprintf(PDB, "END");
        fclose(GRO);
        fclose(PDB);
    }
```

```
    fclose(LIS);
    remove("traj.txt");
    system("crysol *.pdb exp.dat -cst -sm 2.0 -ns 100");
    LIS = fopen("del.txt", "r");

    for(cont = 0; cont < 8500; cont ++)
    {
        fscanf(LIS, "%[^\n]%*c", command);
        system(command);
    }

    fclose(LIS);
    remove("del.txt");

    for(cont = 0; cont < MAX; cont ++)
    {
        for(ador = 0; ador < MAX; ador ++)
        {
            med[cont][ador] = 0;
        }
    }

    LIS = fopen("arqint.txt", "r");

    for(cont = 0; cont < 500; cont ++)
    {
        fscanf(LIS, "%[^\n]%*c", command);
        FIT = fopen(command, "r");
        fscanf(FIT, "%[^\n]%*c", lixo);
        fscanf(FIT, "%[^\n]%*c", lixo);
        fscanf(FIT, "%[^\n]%*c", lixo);
        fscanf(FIT, "%[^\n]%*c", lixo);
        fscanf(FIT, "%[^\n]%*c", lixo);
        fscanf(FIT, "%[^\n]%*c", lixo);
        fscanf(FIT, "%[^\n]%*c", lixo);

        for(ador = 0; ador < 105; ador ++)
        {
            for(i = 0; i < 4; i ++)
            {
                fscanf(FIT, "%lf", &arm);
                med[ador][i] = med[ador][i] + arm;
            }
        }

        fclose(FIT);
        remove(command);
    }
    fclose(LIS);
    remove("arqint.txt");

    for(cont = 0; cont < 105; cont ++)
    {
        for(ador = 0; ador < 4; ador ++)
        {
            med[cont][ador] = med[cont][ador]/500;
        }
    }
    LIS = fopen("media_cg.dat", "w");
    fprintf(LIS,  "Arquivo  de  dados  do  fator  de  forma  coarse-
grained\n");
```

```
    for(cont = 0; cont < 105; cont++)
    {
        for(ador = 0; ador < 4; ador++)
        {
            fprintf(LIS, "  %8.6lE", med[cont][ador]);

            if(ador == 3)
            {
                fprintf(LIS, "\n");
            }
        }
    }

    fclose(LIS);
    remove("crysol_summary.txt");
    remove("traj_protcg.trr");
    LIS = fopen("media.dat", "r");
    FIT = fopen("media_cg.dat", "r");
    fscanf(LIS, "%[^\n]%*c", lixo);
    fscanf(FIT, "%[^\n]%*c", lixo);
    PDB = fopen("formfactor.dat", "w");

    for(cont = 1; cont < 106; cont ++)
    {
        fscanf(LIS, "%lf", &arm);
        fprintf(PDB, "  %8.6lE", arm);
        fscanf(LIS, "%lf %lf %lf", &arm, &jot, &des);
        fprintf(PDB, "  %8.6lE  %8.6lE  %8.6lE", arm, jot, des);
        fscanf(FIT, "%lf %lf %lf %lf", &des, &des, &des, &arm);
        fprintf(PDB, "  %8.6lE\n", arm);
    }

    fclose(LIS);
    fclose(FIT);
    /*remove("media.dat");
    remove("media_cg.dat");*/

}

void trajtxt(char *nometpr)
{
    int cont; /* Contador */
    char aux[MAX]; /* Auxiliar para formar os comandos do trjconv */
    FILE *SAI; /* Arquivo de saída do trjconv */
    FILE *DEL; /* Arquivo para apagar os gro, pdb e log*/
    FILE *ARQ; /* Lista de arquivos fit para a média */
    SAI = fopen("traj.txt", "w");
    DEL = fopen("del.txt", "w");
    ARQ = fopen("arqint.txt", "w");
    strcpy(aux, "echo 1 | gmx trjconv -s ");
    strcat(aux, nometpr);
    strcat(aux, " -f traj_prot.trr -o ");

    for(cont = 1501; cont <= 2000; cont++)
    {
        fprintf(SAI, "%straj%d.pdb -dump %d -pbc atom\n", aux, cont,
cont);
        fprintf(DEL, "rm traj%d.pdb traj%d00.log traj%d00.alm\n",
cont, cont, cont);
        fprintf(ARQ, "traj%d00.fit\n", cont);
```

```
    }

    fclose(SAI);
    fclose(DEL);
    fclose(ARQ);
}

void trajpcg(char *nometpr)
{
    int cont; /* Contador */
    char aux[MAX]; /* Auxiliar para formar os comandos do trjconv */
    FILE *SAI; /* Arquivo de saída do trjconv */
    FILE *DEL; /* Arquivo para apagar os gro, pdb e log*/
    FILE *ARQ; /* Lista de arquivos fit para a média */

    SAI = fopen("traj.txt", "w");
    DEL = fopen("del.txt", "w");
    ARQ = fopen("arqint.txt", "w");

    strcpy(aux, "echo 2 | gmx trjconv -s ");
    strcat(aux, nometpr);
    strcat(aux, " -f traj_protcg.trr -o ");

    for(cont = 1501; cont <= 10000; cont++)
    {
        fprintf(SAI, "%straj%d.gro -dump %d -pbc atom\n", aux, cont,
cont);
        fprintf(DEL, "rm traj%d.gro traj%d.pdb traj%d00.log\n", cont,
cont, cont);
        fprintf(ARQ, "traj%d00.fit\n", cont);
    }

    fclose(SAI);
    fclose(DEL);
    fclose(ARQ);
}
```