Introdução ao uso do *Gromacs* como ferramenta para dinâmica molecular

Anderson Lima dos Santos

1 de maio de 2023

PREFÁCIO

Esse material tem como objetivo apresentar e auxiliar estudantes de física e biologia com o uso do ***Gromacs***, um pacote de dinâmica molecular projetado para simulações de proteínas, lipídios e ácidos nucleicos. Todo conteúdo mostrado nas pagínas seguintes foram coletadas no *site* de tutoriais do *Gromacs*, fórum de discussões do *Gromacs* e nos dois anos de Iniciação Cíentifica(IC) feitas na Universidade de São Paulo(USP), tendo como orientação a Prof. Helena Maria Petrilli (USP) e Prof. Marcos Brown Gonçalves (UTFPR).
É importante ressaltar que para o estudante que tenha essa como seu primeiro contato com o *Gromacs*, que é de extrema importância sempre visitar o fórum de discussão, pois grande parte de suas dúvidas já devem ter sido resolvidas por algum usuário ao redor do mundo, sendo assim, você pode vir a encontra a resposta dentro do fórum de discussão. Para aqueles já familiarizados com o uso do *Gromacs* e queiram acrescentar problemas e informações ao máterial, assim como *feedback*, serão de grande ajuda.
Com isso em mente, o máterial que vou apresentar é baseado nas minhas dificuldades com o programa, e as respostas que encontrei para resolver esses problemas. Como exemplo estarei apresentando os resultados obtidos após fazer os tutoriais *Tutorial 1:Lysozyme in Water* e *Tutorial 5: Protein-Ligand Complex*.

Sumário

1 Uso do Linux e Windows **3**
1.1 Instalação do *Ubuntu* no Windows 3

2 Programas de visualização **5**
2.1 PyMol 5
2.2 Discovery Studio 5

3 Conseguindo as estruturas. **7**

4 Tutoriais *Gromacs* **9**
4.1 Um pouco sobre o *Gromacs* 9
4.2 Tutorial 1:Lysozyme in Water 10
4.2.1 Criação da topologia 11
4.2.2 Definição da caixa de simulação e Solvatação 13
4.2.3 Adicionando Íons 14
4.2.4 Minimização de Energia 15
4.2.5 Equilibração do sistema 17

5 Rodando o *Gromacs* de maneira automática **19**
5.1 Funções que o programa deve conter e suas limitações 19

Lista de Figuras

3.1 Estrutura vista pelo DS 7
3.2 Estrutura vista pelo Pymol 7
3.3 Estrutura 3HTB vista pelo DS 8
3.4 Estrutura 3HTB vista pelo Pymol 8

4.1 Estrutura 1ak1 com água 12
4.2 Estrutura 1ak1 sem água 12
4.3 Caixa com solvente(H_2O) 13
4.4 Sistema ionizado 15

Capítulo 1

Uso do Linux e Windows

Como assunto inicial acredito que seja importante entender que o *Gromacs* não possui uma interface visual, ou seja, para sua execução é necessario usar o terminal de seu computador, de preferência o terminal do *Linux*, pois o *Gromacs* foi projetado para ser usado com a interface *Linux*. Entretanto, no inicio de minha IC pesquisei formas de usa-lo tendo como sistema operacional o *Windows* e uma das formas encontradas é usando o subsistema *Linux* para *Windows*. Esse subsistema consiste em ativar o *Windows Subsystem for Linux(WSL)* nos progamas de seu computador, é uma ferramenta muito útil que permite o uso de *softwares* nativos para o *GNU/Linux* sem a necessidade de instalar emuladores ou maqínas virtuais. Sendo assim com ele ativo é possivel instalar o *Ubuntu* em sua máquina e com isso usar o *Ubuntu* como terminal para executar o *Gromacs*. Essa extenção funciona da mesma forma que o *Linux*, mas devido ao fato de estar sendo executada em *Windows* possivelmente ocorra uma perda de eficiência no decorrer da simulação, entretanto para simulações curtas como as vistas nos tutoriais será útil.

1.1 Instalação do *Ubuntu* no Windows

Para aqueles que tem como sistema operacional o *Linux*, podem pular para a próxima sessão.

Capítulo 2

Programas de visualização

Uma parte importante para processo de Dinâmica Molecular é visualizar o sistema estudado após cada passo executado. Para que possamos visualizar o sistema usamos software separados, como ***PyMOl, Discovery Studio, Alvograd etc.*** Nesse texto estarei focando no primeiro software citados, o *Discovery Studio e o Alvograd* são úteis, mas acredito que para nossos objetivos o Pymol será suficientes, além de ser o sistema que eu tive maior contato. Não existe mistério para executar o download do pymol, podemos fazer isso rapidamente acessando o *site* do criador.
Ter um sistema para visualizar a dinâmica é importante, pois muitos dos passos que executamos no gromacs podem vir a quebrar as estruturas biológicas devido a erro de códigos ou processos mal feitos, podemos também usar esses software para limpar as estruturas e até mesmo limpar a simulações, limpar a proteína ou construir o ligante, verificando também se esse se encontra bem localizado na região de interesse.
É claro que é mais confiável verificar esses dados com auxílio de comandos do próprio *Gromacs*[1] que retornam um gráfico de informações importantes, mas para os primeiros passos onde fazemos o sistema, estabilizamos e minimizamos os átomos o aspecto visual acaba sendo muito útil. Portanto é de grande utilidade buscarmos entender o funcionamento de ferramentas visuais para que possamos através delas verificarmos a nossa simlação, de modo que estejamos seguros dos passos que fizemos.

2.1 PyMol

2.2 Discovery Studio

[1]Devido a grande variedade de comandos é necessário dar um tempo para leitura dos principais

Capítulo 3

Conseguindo as estruturas.

O primeiro passo antes de iniciarmos o primeiro tutorial é conseguir as devidas estruturas que iremos usar durante o processo de dinâmica molecular, essas estruturas podem ser adquiridas no *site RCSB PDB Protein Data Bank* no formato PDB. Esse *site* possui um banco de dados com as estruturas proteicas e ácidos nucléicos que foram obtidos através da difração de raio-X por diversos cientistas ao redor do mundo. É importante ressaltar que esses arquivos podem vir a ter aminoácidos faltantes em sua estrutura, essa falta é devida a incerteza em se calcular a posição do átomo na próteina, sendo assim calculada duas ou mais posições possiveis para o mesmo átomo, portanto, é necessário após fazer o download da estrutura PDB, abri-la em um editor de texto e verificar a região responsável em informar os resíduos faltantes, essa região do arquivo estará com o seguinte nome *missing residues*.
Para nosso primeiro tutorial estaremos usando a estrutura de código PDB *1AKI*, abaixo podemos ver como é a estrutura vista com o *Discovery Studio(DS)*, e com o *PyMol* respectivamente

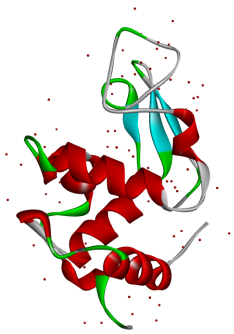

Figura 3.1: Estrutura vista pelo DS

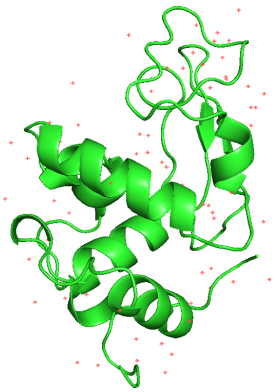

Figura 3.2: Estrutura vista pelo Pymol

Podemos verificar que as estruturas não apresentam diferenças vistas nos dois programas de visualização, tendo isso em mente podemos ter a livre escolha de qual programa iremos utilizar no decorrer de nossa simulação, podemos até utilizar de ambos se assim desejarmos. Entretanto, estarei seguindo com uso de apenas um deles, para que assim não ocorra confusão na apresentação dos dados.
Embora muitas vezes a proteína ou lipídio apresenta uma aparência sem quebras ou até mesmo imperfeições do tipo estrutural, poderemos ter problemas no decorrer da simulação, caso não tomemos os devidos cuidados em verificar através de um arquivo de texto a área de *missing residues*, essa verificaão pode ser feita com uso do *Notepad++*[1], entretanto a estrutura *1AKI* usada no primeiro tutorial não apresenta falta de resíduos.
O segundo tutorial que iremos fazer usaremos a proteína de código PDB *3HTB*, abaixo podemos verificar as estruturas com o uso do DS e com o uso do *PyMol* respectivamente,

[1] ou qualquer outro *software* de edição de texto a sua escolha

Figura 3.3: Estrutura 3HTB vista pelo DS

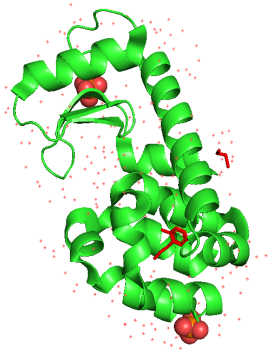

Figura 3.4: Estrutura 3HTB vista pelo Pymol

Assim como a estrutura *1AKI*, podemos verificar que existem mudanças apenas no aspecto visusl, podemos então verificar no arquivo de texto a região de *missing residues*para termos a segurança de estarmos lidando com uma proteína "completa", sem faltas em regiões significativas.

REMARK	465	MISSING RESIDUES
REMARK	465	THE FOLLOWING RESIDUES WERE NOT LOCATED IN THE
REMARK	465	EXPERIMENT. (M=MODEL NUMBER; RES=RESIDUE NAME; C=CHAIN
REMARK	465	IDENTIFIER; SSSEQ=SEQUENCE NUMBER; I=INSERTION CODE.)
REMARK	465	
REMARK	465	M RES C SSSEQI
REMARK	465	LEU A 164
REMARK	500	

Podemos verificar que na estrutura *3HTB* existe a falta do aminoácido LEU na posição 164 da estrutura, poderemos desconsiderar esse aminoácidos por dois motivos que são importantes

1. É um aminoácido que se encontra nas extremidades da proteína estudada.
2. É um aminoácido que não executa uma grande função para a proteína, ou seja, não se encontra na região ativa ou no sítio catalítico da estrutura.

Esses são dois grandes parâmetros para se levar em consideração quando fazemos uso dessas estruturas para simulação em Dinâmica Molecular, caso ocorra de termos resíduos faltantes que tem uma ou as duas caracteristicas acima, devemos criar esse resísuo através de programas externos, assim como usar ferramentas de Mecânica Quântica de forma que o resíduo fique bem ajustado para simulação.
Mais adiante faremos simulações com elementos que denominamos Ligantes, esses ligantes possuem estrutura própria, então será necessário o dobro de cuidado no decorrer da simulação devido ao fato que usaremos ferramentas externas para gerar aquilo que chamaremos de topologia do ligante, e para isso estaremos usando de ferramentas que fazem cálculos quânticos.

Capítulo 4

Tutoriais *Gromacs*

4.1 Um pouco sobre o *Gromacs*

O *software* ***Gromacs*** foi projetado com o objetivo de fazer simulações de dinâmica molecular para compostos químicos e biológicos, uma ferramenta que pudesse entregar resultados o mais rapído possível e de forma precisa para seus usuários. Para que as simulações sejam feitas com um número muito grande de átomos é necessário abrir mão de cálculos quânticos e nos auxiliarmos em bases de mecânica clássica, dessa forma, podemos tratar sistemas de multiplos corpos sem demandarmos muito tempo com cálculos quânticos, é claro que abrir mão de uma ferramenta pode acarretar na perda de informações, mas no nosso caso tais informações podem não ser úteis no primeiro momento.

Modelagem molecular de um sistema, tem como objetivo descrever um precesso mais geral em torno de um modelo realista, para que isso possa ser feito usamos de técnicas de mecânica estatística e mecânica clássica afim de descrever um sistema real da forma mais precisa possível. Em situações ideais, a equação de Schrödinger dependente do tempo(relatívistica)(4.1) é capaz de descrever de maneira precisa sistemas atômico e moleculares com alta precisão,

$$\hat{H}\left|\psi(\vec{r},t)\right\rangle = i\hbar\frac{\partial}{\partial t}\left|\psi(\vec{r},t)\right\rangle \tag{4.1}$$

entretanto sistemas que possuem muitos átomos tornam esse tratamento muito complexo e robusto, fazendo com que simplificações sejam necessárias, quanto mais complexo um sistema e quanto maior for o tempo de simulação mais aproximações se tornam necessárias. Abrimos assim mão de métodos *ab initio* e usamos de ferramentas semi empíricas para executarmos as simulações.

Para modelagem molecular usaremos de ferramentas semi empíricas, ou seja dados obtidos experimentalmente por cientistas ao redor do mundo, junto de técnicas computacionais de mecânica estatística e mecânica clássica, para que isso seja possível, existem dois métodos existentes

1. Simulações de Monte Carlo

2. Simulações de Dinâmica Molecular .

apesar que para sistema dinâmicos temos o segundo método mais usual. Grande parte do problema se deve a complexidade exibida pelo $\mathcal{H}$ dito como hamiltoniano do sistema

$$\mathcal{H} = \frac{p^2}{2m} + \mathcal{V}(r) \tag{4.2}$$

quando se tem mais de uma partícula no sistema, o potêncial $\mathcal{V}$ se apresenta muito complicado para ser tratado. Para tratarmos esse problema usamos mecãnica estatística e outras simplificações.

Dinâmica Molecular

As simulações de dinâmica molecular é regída pela segunda lei de newton para os N atomos que estão interagindo entre sí

$$m_i \frac{\partial^2 r_i}{\partial t^2} = F_i, \{i = 1, 2, 3, ...N\} \tag{4.3}$$

onde temos F_i como a derivada negativa de uma função potência $V(r_1, r_2, r_3, ..., r_N)$

$$F_i = -\frac{\partial V(r_1, ..., r_i, ..., r_N)}{\partial r_i} \tag{4.4}$$

as equações são resolvidas simultâneamete em pequenos passos de tempo, durante um intervalo de tempo pré-estabelecido, tendo sempre o controle da temperatura e pressão existentes no sistema estudado. As coordenadas em função do tempo são gravadas em um arquivo de saída, sendo esse denominado *arquivo de trajetória*. Temos que após um determinado passo de tempo o sistema atingirá o equilíbrio e a partir disso podemos obter propriedades macroscópicas importantes.

Apesar de obtermos muitos resultados importantes, devemos ter em mente que a ferramenta matemática usada são as leis de Newton, que implicam no uso de mecânica clássica, sendo assim, como podemos ver no próprio manual do *Gromacs* teremos,

> *Using Newton's equation of motion automatically implies the use of classical mechanics to describe the motion of atoms. This is all right for most atoms at normal temperatures, but there are exceptions. Hydrogen atoms are quite light and the motion of protons is sometimes of essential quantum mechanical character. For example, a proton may tunnel through a potential barrier in the course of a transfer over a hydrogen bond. Such processes cannot be properly treated by classical dynamics! Helium liquid at low temperature is another example where classical mechanics breaks down.*

ou seja, em determinadas condições poderemos ter limitações com a teoria usada.

Para não usarmos as simulações reais de dinâmica quântica, podemos fazer uma das duas coisas que se seguem, e foram retiradas do manual do *Gromacs*[1]:

1. Realizando a simulação usando osciladores harmônicos para ligação, devemos fazer correções na energia interna total dada por,
$$E = K + U \tag{4.5}$$
e calor específico dado por C_V, assim como a entropia S como a energia livre A ou G caso sejam calculadas.

2. Podemos tratar as ligações(e os ângulos de ligação) como restrições nas equações de movimento. A lógica por trás disso é que um oscilador quântico em seu estado fundamental se assemelha a uma ligação restrita mais de perto do que um oscilador clássico.

Existem muitas outras informações importantes para o uso do *Gromacs*, mas deixarei para o final de cada simulação explicar o significado de cada módulo usado e o que ele retorna como resultado, sendo assim podemos iniciar o primeiro tutorial.

4.2 Tutorial 1:Lysozyme in Water

Como dito anteriormente, grande parte do processo se da na interface do terminal *Ubuntu*, caso não o tenha instalado em sua máquina certifique-se de baixa-lo, para que assim possa prosseguir com o tutorial 1, se estiver usando windowns o Capítulo 1 mostra como fazer isso.

O gromacs possui uma linguagem própria de comandos, sendo assim, antes de iniciarmos o tutorial devemos entender a estrutura lógica usada. Escrevendo **gmx** no terminal *Ubunto*, você estará chamando o *gromacs*, caso não tenha ocorrido nenhum problema em seu download ele retornará uma mensagem de "Olá"para o usuário, como segue abaixo, seguido dos nomes daqueles responsáveis pela escrita do código[2],

[1] É de grande importância ler algumas partes desse manual, para se ter uma base daquilo que esta ocorrendo na dinâmica

[2] A versão usada no período de escrita desse texto é a versão 2020.1, logo, esse texto inicial pode ser diferente de versão para versão

:-) GROMACS - gmx, 2020.1-Ubuntu-2020.1-1 (-:

.
.
.

caso não apareça algo desse tipo, verifique se instalou o gromacs corretamente, sendo assim volte ao Capítulo 1, caso seja necessário, caso tenha aparecido essa mensagem em seu terminal pode-se iniciar iniciar os preparativos para a dinâmica. Abaixo temos a estrutura mais simples da linha de comando do gromacs, todos os códigos que inserir seguiram esse inicio de estrutura, ou seja, inicialmente você chamará o *gromacs*, e notificará para ele que tipo de método ele deve executar.

gmx *nome do módulo* ...

No lugar do *nome do módulo* escrevemos o método que queremos executar, veja o exemplo abaixo, caso o objetivo seja calcular o *RMSD* escrevemos o seguinte.

gmx rms ...

Caso queira saber a função que o módulo usado executa podemos usar da ferramenta *help* do gromacs.

gmx help *nome do método* ou
gmx *nome do método* ***-h***

tanto para as funções quanto para as linhas de códigos, podemos conseguir maiores informações utilizando a documentação do *Gromacs*, que irá nos fornecer uma quantidade enorme de auxílios para o uso dessa ferramenta. Voltaremos mais tarde com alguns dos métodos que usaremos na análise da simulação feita, mas no momento iremos focar na criação da topologia.

4.2.1 Criação da topologia

A topologia é um dos arquivos mais importantes pois nela ficará gravada as posições iniciais de cada um dos átomos assim como carga e suas massas, estaremos sempre fazendo referência a ela quando executarmos os módulos do *Gromacs*. O primeiro passo é eliminar qualquer resíduo da estrutura PDB que coletamos no *site Protein Data Bank*, qualquer estrutura que não seja de interesse podemos descartar, e nesse primeiro passo caso esse arquivo venha com as hélices da proteína duplicada podemos descartar sua cópia, pois a simulação apresentará defeitos caso tenhamos mais de uma estrutura, ligantes tambem devem ser descartados aqui, pois o gromacs não consegue criar uma topologia para ligantes, isso é feito por software externos, como veremos mais adiante.

Outro passo muito importante é verificar na estrutura PDB os resíduos que faltam como já enunciado anteriormente, resíduos nos extremos da proteína podem estar ausentes pois esses (caso não executem uma função muito importante para dinâmica) não afetaram no processo de criar a topologia executado pelo pdb2gmx, agora caso o resído se encontre no meio da estrutura é necessário modelar esses resíduos com software externos. Entenda que o pdb2gmx não gera estruturas, ele apenas gera topologias para uma quantidade finita de resíduos, particulamente aqueles definido pelo campo de força[3].

Podemos remover a água da estrutura pdb usando o comando a seguir, ou podemos usar ferramentas do *PyMol* ou *Discovery Studio*, para retirar essas águas,

grep -v HOH *nome_do_Arquivo*.pdb > *Arquivo*_clean.pdb

teremos então as estruturas após limpar as águas como as figuras abaixo.

[3]Estruturas metálicas também apresentam um impasse na dinâmica, sendo necessário um pouco mais de empenho e *softwares* externos, o *gromacs* é melhor quando usado com estruturas biológicas, durante o período de minha IC apenas tive contato com estruturas biológicas

Figura 4.1: Estrutura 1ak1 com água

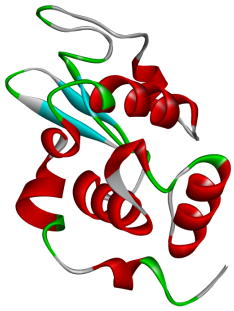

Figura 4.2: Estrutura 1ak1 sem água

Para executarmos a topologia de nossa proteina usaremos o módulo ***pdb2gmx***, para isso escrevemos o seguinte comando.

gmx pdb2gmx -f *nome_do_Arquivo*_clean.pdb **-o** *nome_do_Arquivo_processed.gro* **-water spce**

Quando executarmos o código acima o *gromacs* irá pedir o campo de força que iremos usar no processo. Aqui é uma parte muito importante na dinâmica, usaremos aqui a opção 15, o campo *OPLS-AA/L*, essa linha de código irá nos retornar três arquivos com extessões diferentes:

1. **.gro**: arquivo de estrutura pós-processada
2. **.top**: arquivo de topologia
3. **.itp**: arquivo de restrição de topologia

ao inspecionar o arquivo *topol.top* podemos identificar as seguintes linhas

; Include forcefield parameters
#include "oplsaa.ff/forcefield.itp"

que informa ao gromacs os paramêtros do campo de força usados para executar a simulação, seguidas por uma sessão que descreve os átomos presentes na proteína, temos as seguintes características

- **nr:** número do átomo
- **type:** tipo de átomo
- **residue:** nome do residuo ao qual o átomo pertence
- **atom:** nome do átomo
- **cgnr:** o número de carga numérica
- **charge:** carga do átomo
- **mass:** massa do átomo
- **typeB, chargeB, massB:** serão usados para energia livre de pertubação

o resto do arquivo é alto explicativo, consiste em especificar os pares de ligações, ângulos etc. Mais para o final do arquivo temos novamente uma indicação do campo de força usado e uma referência ao arquivo de restrição, assim como tem indicações ao nome das estruturas presentes e o número de estruturas presentes. Partimos então para o segundo passo, a solvatação.

4.2.2 Definição da caixa de simulação e Solvatação

No presente momento estamos simulando apenas uma estrutura em composto aquoso simples, mas é possivel fazer simulações em outros tipos de solventes, desde que tenhamos as características bem definidas para todas as estruturas presentes. Agora o que iremos fazer seguira dois passos, o primeiro será definir a caixa de simulação com uso do módulo *editconf*, em seguida iremos encher essa caixa de água com o módulo *solvate*.

Inicialmente podemos usar do seguinte comando para definição da caixa que iremos usar na simulação,

gmx editconf -f *nome do arquivo*.**gro -o** *nome da caixa*.**gro -c -d 1.0 -bt cubic**

o *nome do arquivo*, é o arquivo *.gro* que o *gromacs* retornou no passo anterior, o *1.0* define a distância da proteína para cada parede da caixa de simulação, essa distancia é de $1nm$. A distância é para assegurar as condições de contorno periódicas da simulação, isso evita que uma proteina encontre sua cópia da outra caixa de simulação. O código acima nos retorna um outro arquivo *.gro*, esse arquivo apenas contém as cordenadas da proteína centralizada na caixa, sendo assim ao abrir em um editor de texto simples veremos o seguinte

```
LYSOZYME
    1960
    1LYS      N    1   4.260   3.251   2.290
    1LYS     H1    2   4.336   3.305   2.252
    1LYS     H2    3   4.194   3.231   2.218
    1LYS     H3    4   4.216   3.303   2.363
    1LYS     CA    5   4.313   3.124   2.345
    1LYS     CB    7   4.411   3.161   2.457
    1LYS    HB1    8   4.487   3.212   2.418
      .
      .
      .
```

temos o aminoácido, o nome do átomo, o número, coordenada x, coordenada y, coordenada z, podemos então prosseguir com a simulação, o que faremos agora é preencher com solvente com uso do seguinte código,

gmx solvate -cp *nome da caixa*.**gro -cs spc216.gro -o** *nome solv*.gro **-p topol.top**

o *nome da caixa* é aquele definido anteriormente, enquanto o *nome solv* é o arquivo gro de saída que contem agora informações do solvente usado, usamos **spc216.gro** que informa um modelo genérico de solvente para água, e indicamos também o arquivo de topologia *topol.top* que irá sofrer alteração, será escrito nele informações das coordenadas das moléculas de água, quando abrimos o arquivo de saída no *PyMol*, podemos visualizar as águas inseridas para simulação, veja abaixo

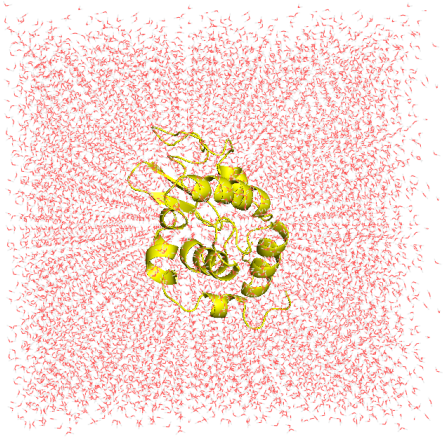

Figura 4.3: Caixa com solvente(H_2O)

o arquivo de topologia anterior teve seu nome alterado mudando para *#topol.top.1#*, esse contém a topologia anterior, estarei omitindo isso no decorrer desse texto, mas lembre-se sempre que quando você insere ou retira algo do arquivo de topologia, esse é reescrito de forma que o anterior tem seu título alterado. Abrindo o novo arquivo *topol.top* e visualizando as últimas linhas verificamos a seguinte informação,

[molecules]
; Compound #mols
Protein_chain_A 1
SOL 10644

vemos que tem uma nova linha escrita, essa denominada *SOl*, que faz referência ao solvente inserido no sistema e o número de moléculas inseridas.

4.2.3 Adicionando Íons

Agora que temos o sistema solvatado é necessário adicionar os íons, que serão os responsáveis por neutralizar o sistema, pois esse se encontra com carga líquida superior a 0. Para que possamos adicionar os íons iremos usar o módulo *genion* do *gromacs*. De acordo com o própio *gromacs* o genion lê o arquivo de topologia e insere em algumas posições onde se encontra água alguns íons informado pelo usuário. O *genion* usa como entrada um arquivo de extensão *.tpr*, esse arquivo contém todas as informações e parâmetros para todos os átomos do sistema, esse arquivo pode ser obtido com auxílio do módulo *grompp*.

Para usar o *grompp* precisamos de um arquivo de entrada de extensão *.mdp*, esse pode ser obtido no própio site de tutoriais do *gromacs*, como o intuito desse texto é auxiliar estarei colocando o arquivo de texto abaixo, caso o leitor queira copiar para um editor de texto simples, apenas lembre-se de salvar esse arquivo com o nome ***ions.mdp***.

```
; ions.mdp - used as input into grompp to generate ions.tpr
; Parameters describing what to do, when to stop and what to save
integrator = steep                      ; Algorithm (steep = steepest descent minimization)
emtol = 1000.0                  ; Stop minimization when the maximum force ¡ 1000.0 kJ/mol/nm
emstep = 0.01                                   ; Minimization step size
nsteps = 50000                  ; Maximum number of (minimization) steps to perform
; Parameters describing how to find the neighbors of each atom and how to calculate the interactions
nstlist = 1                     ; Frequency to update the neighbor list and long range forces
cutoff-scheme = Verlet                          ; Buffered neighbor searching
ns_type = grid                   ; Method to determine neighbor list (simple, grid)
coulombtype = cutoff             ; Treatment of long range electrostatic interactions
rcoulomb = 1.0                              ; Short-range electrostatic cut-off
rvdw = 1.0                                  ; Short-range Van der Waals cut-off
pbc = xyz                       ; Periodic Boundary Conditions in all 3 dimensions
```

É importante teros em mente que esse arquivo de texto é aplicavel apenas ao tutorial que estamos seguindo, para ser mais exato, esse arquivo de texto é aplicavel apenas para o campo de força *OPLS-AA*. Vamos ver mais adiante que com outros campo de força é necessário modificar tais arquivos de texto. Sendo assim para que possamos criar nosso arquivo *.tpr* precisamos seguir a seguinte estrutura de código,

gmx grompp -f ions.mdp -c *nome solv*.gro **-p topol.top -o ions.tpr**

Recebemos como retorno dois arquivos, o primeiro é o *ions.tpr*, que possui a descrição atômica do nosso sistema e que consequentemente iremos usar no sistema *genion* e o segundo é um arquivo *mdout.mdp*, que é o arquivo de saída *mdp*, agora informamos o genion sobre os íons, para que isso possa ser feito usaremos o seguinte código,

gmx genion -s ions.tpr -o *nome solv*_**ions.gro -p topol.top -pname NA -nname CL -neutral**

ao inserir esse código será pedido o grupo ao qual você ira retirar molécolas para inserir os íons nesse momento escolha o **grupo 13**, ou aquele que estiver escrito,

*(SOL) has 3**** elements*

com isso ele estará retirando molécolas do solvendo e inserindo íons, afim de neutralizar a carga líquida da proteína, como podemos ver na linha de comando a parte escrita *-neutral*.

- **-s:** fornece o arquivo de estrutura/estado
- **-o:** gera um arquivo *.gro* como saída
- **-p:** processamos a topologia
- **-pname, -nname:** nome dos íons que iremos usar, escrito apenas com os símbolos elementares, **importante não usar o nome dos átomos isso pode acarretar problemas nas etapas seguintes**

Agora podemos verificar o arquivo de topologia, que foi novamente reescrito, abrindo a nova topologia temos no final do arquivo a informação dos íons adcionados e por fim usando o *PyMol*, podemos abrir o arquivo ***nome solv_ions.gro*** e verificar a existência dos íons adcionados pelo programa, como mostra a Figura 4.4. Portanto podemos passar para a próxima etapa, *Minimização de Energia*

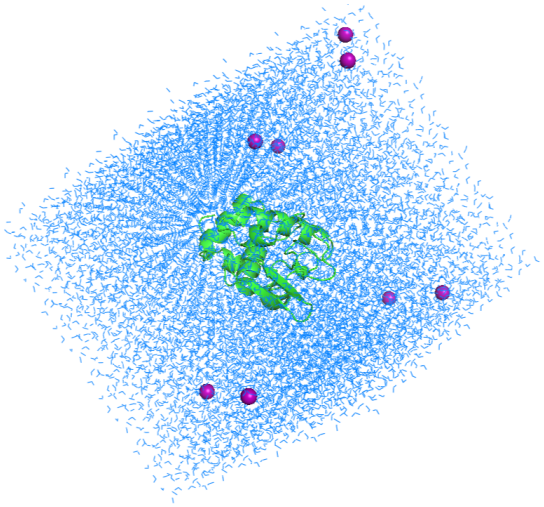

Figura 4.4: Sistema ionizado

4.2.4 Minimização de Energia

Temos agora o sistema montado, dentro da caixa de simulação com o solvente inserido e por fim neutralizado, é importante que tenha seguido corretamente cada passo que apresentei, caso tenha aparecido algum erro é necessário que volte e tente executar o passo novamente, para que não gere problemas mais para frente. Agora iniciaremos o processo de *minimização de energia*, esse processo consiste em garantir qu o sistema ão tenha choques ou geometrias inadequadas. para que isso seja possivel iremos relaxar a estrutura.

Esse processo se assemelha com a adição de íons feitas anteriormente, usaremos novamente um arquivo externo, ele irá conter as informações que precisamos para minimizar a estrutura, abaixo esta o arquivo que você deve usar como extenção ***.mdp***, usaremos o grompp novamente, e dessa vez usaremos o motor MD para que ele execute uma pequena simulação afim de deixar o sistema equílibrado e minimizado.

; minim.mdp - used as input into grompp to generate em.tpr
; Parameters describing what to do, when to stop and what to save
integrator = steep ; Algorithm (steep = steepest descent minimization)
emtol = 1000.0 ; Stop minimization when the maximum force $< 1000.0 kJ/mol/nm$
emstep = 0.01 ; Minimization step size
nsteps = 50000 ; Maximum number of (minimization) steps to perform

; Parameters describing how to find the neighbors of each atom and how to calculate the interactions
nstlist = 1 ; Frequency to update the neighbor list and long range forces
cutoff-scheme = Verlet ; Buffered neighbor searching
ns_type = grid ; Method to determine neighbor list (simple, grid)
coulombtype = PME ; Treatment of long range electrostatic interactions
rcoulomb = 1.0 ; Short-range electrostatic cut-off
rvdw = 1.0 ; Short-range Van der Waals cut-off
pbc = xyz ; Periodic Boundary Conditions in all 3 dimensions

Esse arquivo assim como o *ions.mdp* é necessário para que possamos criar o arquivo tpr, que como já dito anteriormente terá todas as informações dos átomos internos a caixa, para que possamos executar seguiremos o seguinte código,

gmx grompp -f minim.mdp -c *nome solv* **ions.gro -p topol.top -o em.tpr**

agora com o arquivo tpr criado podemos executar essa pequena simulação, usando o *mdrun.*

gmx mdrun -v -deffnm em

Temos o sinalizador **-v**, ele irá tornar o *mdrun* detalhado, mostrando cada passo na tela, imprimindo o progresso de cada passo no cmd, os sinalizadores *-deffnm* irá definir os nomes dos arquivos de entrada e saída, caso você não tenha usado o nome **'em'**, será necessário especificar com o sinalizados **-s** o nome que usou.

- **em.log:** arquivo de log de texto ASCII do processo EM
- **em.edr:** arquivo binário de energia
- **em.trr:** Trajetória binária de precisão total
- **em.gro:** Estrutura com energia minimizada

Podemos verificar se a minimização ocorreu de maneira correta avaliando a saída final. Assim como dito no *site* do tutorial 1, podemos verificar analisando a Energia potêncial e a força máxima. A energia pot encial deve apresentar-se negativa na ordem de 10^5 ou 10^6, variando de sistema para sistema, e a força máxima como definido no arquivo *.mdp* deve ser menor que 1000kj $^{-1}nm{-}1$, caso ocorra algo fora do previsto, podemos alterar o arquivo *minim.mdp* afim de obter resultads melhores. Abaixo estão os resultados obtidos com minha minimização, se os seus não estiverem iguais não se preocupe, o principal é estarem com as características citadas anteriormente.

Steepest Descents converged to Fmax ¡ 1000 in 855 steps
Potential Energy = -5.8745306e+05
Maximum force = 8.8212927e+02 on atom 736
Norm of force = 2.0725371e+01

Podemos fazer uma analise visual com uso do *gromacs* para que isso seja feito usamos o módulo *energy* do gromacs, com a entrada *em.edr*, que contém as informações dessa minimização, para que possamos obter um gráfico usamos do seguinte código,

gmx energy -f em.edr -o potential.xvg

ao inserir esse código, você irá obter a seguinte saída,

Select the terms you want from the following list by
selecting either (part of) the name or the number or a combination.
End your selection with an empty line or a zero.

1 Bond	2 Angle	3 Proper-Dih.	4 Ryckaert-Bell.
5 LJ-14	6 Coulomb-14	7 LJ-(SR)	8 Coulomb-(SR)
9 Coul.-recip.	10 Potential	11 Pressure	12 Vir-XX
13 Vir-XY	14 Vir-XZ	15 Vir-YX	16 Vir-YY
17 Vir-YZ	18 Vir-ZX	19 Vir-ZY	20 Vir-ZZ
21 Pres-XX	22 Pres-XY	23 Pres-XZ	24 Pres-YX
25 Pres-YY	26 Pres-YZ	27 Pres-ZX	28 Pres-ZY
29 Pres-ZZ	30 #SurfSurfTen	31 T-rest	

escolha a opção 10 que corresponde a energia potencial do sistema, em seguida selecione 0, para que tenha entrada. Podemos avaliar esse gráfico através de um programa em python simples, se abrirmos o arquivo *potential.xvg*, veremos que ele é apenas um arquivo que associa para cada tempo(eixo x) uma energia potencial(eixo y).

4.2.5 Equilibração do sistema

Capítulo 5

Rodando o *Gromacs* de maneira automática

Após executar ambos os tutoriais o leitor pode notar que existe uma difículdade em rodar o gromacs de maneira que não ocorra nenhum erro, sem que o mesmo esteja presente em todo processo. Esses erros podem ocorrer por diversos motivos, erro de escrita no terminal, estrutura biológica que não possuem topologia no arquivo correto, podendo até presentar erros mais graves quando tratamos do tutorial 5. Tendo em vista essa difículdade acredito que uma parte fundamental dos avanços tecnológicos é poder automatizar partes do processo, de forma que o usuário possa focar apenas naquelas partes que são realmente importantes, ou ficar atento apenas ao significado dos resultados.

E importante entretanto deixar claro que todo o processo executado nos capítulos anteriores são importantes para o aprendizado, é necessário entender o que o gramacs faz e o que cada passo faz, para que seja possivel entender os resultados físicos obtidos, tendo isso em mente esse capítulo apresenta minha tentativa de escrever um código que execute o gromacs de maneira autônoma, apresntarei os problemas que se tem na escrita desse código e os meios para sobrepor as díficuldades, sendo assim denominei esse código como *Programa de Auxílio e automoção do Gromacs (PAaG)*, então daqui em diante farei referência a ele como *PAaG*.

5.1 Funções que o programa deve conter e suas limitações

Para que possamos escrever um código que consiga lidar com o processo completo, é necessário ter em mente quais os problemas que se deve resolver e quais são as suas limitações, fazer uma lista desses problemas geralmente se mostra muito útil pois assim fica mais fácil de visualizar respostas, sendo assim abaixo eu estruturei uma visão mais geral daquilo que devemos executar.

Preparação manual das estruturas
$\Downarrow$
Escrita das linhas de comando de maneira organizada e verifiação das saídas de erros
$\Downarrow$
Finalização da simulação
$\Downarrow$
Análise

- **Preparação manual das estruturas**: Essa parte é destinada a preparar a estrutura para o Gromacs, partes que não é confiável executar de maneira autônoma serão feitas aqui, como:
  - Limpeza das "impurezas"da estrutura
  - Retirada das águas[1]

[1]Esse passo o código consegue executar, mas já que estaremos com a mão na massa podemos fazer isso manualmente

  - Eliminar caso exista estruturas duplas do código PDB
  - Retirada do ligante

- **Escrita das linhas de comando de maneira organizada e verifiação das saídas de erros**: Essa parte aqui é a automática, ou seja, aqui o código executará tudo de maneira autônoma sem que o usuário precise acompanhar, e caso exista um erro ele irá parar automáticamente. Dentre aquilo que ele pode fazer está

  - Escrever as chamadas gromacs automaticamente, escrever um arquivo de texto, onde poderemos verificar passo a passo do processo seguido, de maneira que podemos ver os arquivos que foram escritos e as saídas que foram dadas, assim como os erros encontrados.

- **Análise:** O ideal seria que essa parte fosse também automática, mas devido a necessidade de criação de grupos é preciso uma pouco mais de cuidado para escrever ela.