# Tutorial de dinâmica molecular

Neste tutorial, você irá aprender a executar uma dinâmica molecular de um peptídeo usando a ferramenta GROMACS. Primeiro você irá aprender a instalar o GROMACS, em seguida iremos preparar a estrutura e o sistema. Por fim, você irá aprender a executar a dinâmica (faremos apenas 200 ps de dinâmica) e a analisar os resultados (usando o PyMOL e o RStudio).

Este tutorial foi testado no macOS 14.0. Para outros sistemas operacionais, consulte os links indicados em cada parte do tutorial. Para este tutorial recomendamos que você use ChimeraX ou PyMOL para visualizar os resultados (presumimos que você já tem conhecimento em algum desses programas).

### Passo 1 - instalando o GROMACS

Faça o download do GROMACS em https://manual.gromacs.org/2023.3/download.html

> Dica: você pode fazer isso, digitando no terminal:
>
> `wget https://ftp.gromacs.org/gromacs/gromacs-2023.3.tar.gz`
>
> Note que uma versão mais nova pode estar disponível quando você for fazer a instalação. Então, acesse o endereço principal para obter o link mais recente.

Descompacte o arquivo com o comando:

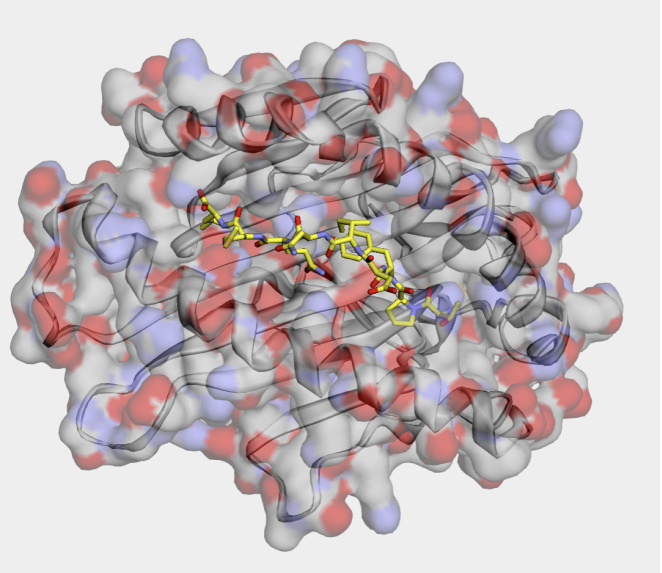
```
tar xfz gromacs-2023.3.tar.gz
```

Agora, vamos acessar o diretório e compilar o GROMACS. Note que você precisará ter um compilador C++ em sua máquina (por exemplo, no macOS você pode instalá-lo com o comando “brew install cmake”; para Windows acesse https://cmake.org/download/).

```
cd gromacs-2023.3
mkdir build
cd build
cmake .. -DGMX_BUILD_OWN_FFTW=ON -DREGRESSIONTEST_DOWNLOAD=ON
make
make check
sudo make install
source /usr/local/gromacs/bin/GMXRC
```

**Passo 2 - preparando a estrutura**

Neste tutorial, vamos analisar um peptídeo obtido na base de dados Propedia. Vamos usar como estudo de caso a molécula MHC complexada com o peptídeo TPYDINQML da proteína GAG do HIV2 (PDB ID: 1a1m).



*PDB 1a1m: cadeia A ligada ao peptídeo TPYDINQML.*

Faça o download do PDB no endereço:

http://bioinfo.dcc.ufmg.br/propedia/complex/view/1a1m-C-A

Você pode fazer isso pelo terminal:

```
cd ~

mkdir teste

cd teste

wget
http://bioinfo.dcc.ufmg.br/propedia/public/pdb/structures/complex/
1a1m_C_A.pdb
```

Agora, vamos remover as águas do PDB. Há centenas de maneiras de fazer isso, mas vamos usar GREP para isso, pois é mais simples e pode ser feito com uma linha de comando:

```
grep -v HOH 1a1m_C_A.pdb > 1a1m_C_A_sem_agua.pdb
```

> Note que Propedia já remove as águas por padrão, então o comando acima na verdade não faz nada. Entretanto, caso você trabalhe com outro PDB, esse comando pode ser útil.

Esse PDB contém uma estrutura de uma proteína de 278 aminoácidos complexada a um peptídeo de apenas 9 aminoácidos. Como queremos apenas executar um pequeno teste rápido, vamos remover a proteína do arquivo e deixar apenas o peptídeo.

Neste exemplo, a proteína está armazenada na cadeia A, enquanto o peptídeo está na cadeia C. Você pode conferir isso abrindo o arquivo. Veja um pedaço dele:

```
ATOM   2269  N   HIS A 278      41.887  29.549  63.289  1.00 84.72           N
ATOM   2270  CA  HIS A 278      42.576  29.630  64.582  1.00 88.14           C
ATOM   2271  C   HIS A 278      41.864  30.592  65.544  1.00 88.02           C
ATOM   2272  O   HIS A 278      42.321  31.740  65.742  1.00 87.44           O
ATOM   2273  CB  HIS A 278      44.070  29.998  64.437  1.00 92.70           C
ATOM   2274  CG  HIS A 278      44.840  29.088  63.520  1.00 96.81           C
ATOM   2275  ND1 HIS A 278      44.229  28.199  62.660  1.00 99.92           N
ATOM   2276  CD2 HIS A 278      46.171  28.975  63.290  1.00 97.14           C
ATOM   2277  CE1 HIS A 278      45.146  27.580  61.937  1.00 98.91           C
ATOM   2278  NE2 HIS A 278      46.332  28.034  62.297  1.00 98.07           N
ATOM   2279  OXT HIS A 278      40.816  30.173  66.077  1.00 85.54           O
ATOM   3111  N   THR C   1      12.995  27.226  17.890  1.00 12.34           N
ATOM   3112  CA  THR C   1      11.913  27.984  17.223  1.00 22.00           C
ATOM   3113  C   THR C   1      10.566  27.516  17.826  1.00 23.62           C
ATOM   3114  O   THR C   1      10.437  26.355  18.237  1.00 21.13           O
ATOM   3115  CB  THR C   1      12.030  27.835  15.702  1.00 22.60           C
ATOM   3116  OG1 THR C   1      11.064  28.649  15.021  1.00 30.74           O
```

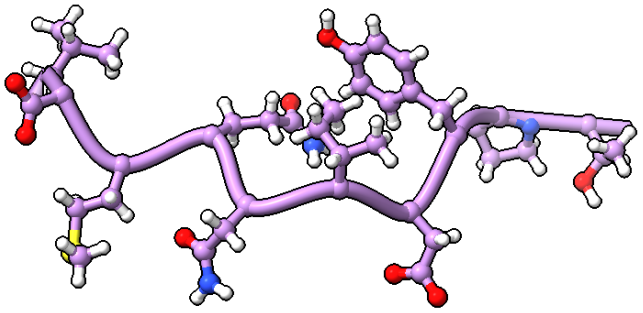
Precisamos remover todas as linhas que sejam correspondentes a cadeia A. Você pode fazer isso manualmente, abrindo o arquivo e apagando as linhas iniciadas em “ATOM” e que tenha a letra A na posição 22.

Mais uma vez, podemos fazer isso com grep:

```
grep -v "ATOM.* A " 1a1m_C_A_sem_agua.pdb > 1a1m_peptideo.pdb
```

Observe que o “-v” é usado para ignorar linhas que tenham o padrão.

**OPCIONAL**: abra o arquivo 1a1m_peptideo.pdb no ChimeraX para ver isso:



**Passo 3 - Convertendo PDB para GRO**

Agora, precisamos converter o arquivo pdb para o formato do GROMACS. Vamos fazer isso usando a ferramenta pdb2gmx, que faz parte do GROMACS.

```
gmx pdb2gmx -f 1a1m_peptideo.pdb -o 1a1m.gro -ff amber99sb-ildn
-water tip3p
```

Note que esse comando cria três arquivos:

- topol.top
- posre.itp
- 1a1m.gro

**Passo 4 - Preparando a caixa d’água**

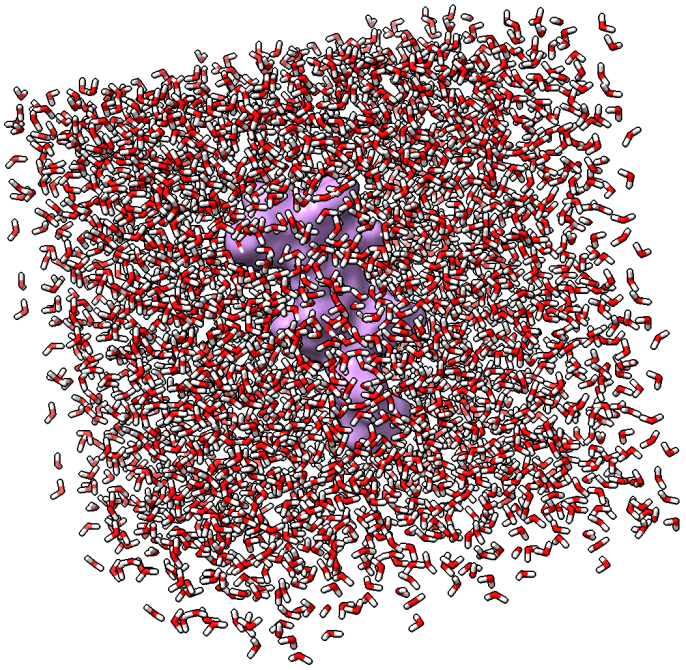
Agora, vamos criar uma caixa d’água. Execute o comando:

```
gmx editconf -f 1a1m.gro -o 1a1m_caixa.gro -c -d 1 -bt cubic
```

Criamos a caixa, mas ainda não enchemos ela de água. Faremos isso agora na etapa de criação do solvente.

gmx solvate -cp 1a1m_caixa.gro -cs spc216.gro -o 1a1m_solvatado.gro -p topol.top

**OPCIONAL**: Vamos abri o arquivo 1a1m_solvatado.gro no ChimeraX para ver como está:



**Passo 5 - Adicionando íons**

Veja que nosso peptídeo está no meio da caixa d'água.

Agora, precisaremos adicionar ions ao sistema (isso irá neutralizar o sistema). Crie um arquivo chamado "ions.mdp" e coloque o seguinte conteúdo:

```
; ions.mdp - used as input to 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 to minimize edge effects in a finite
system
```

Agora execute o comando:

```
gmx grompp -f ions.mdp -c 1a1m_solvatado.gro -p topol.top -o ions.tpr -maxwarn 1
```

Por fim, vamos usar o comando “genion” para concluir a ampliação da concentração salina no sistema:

```
echo SOL | gmx genion -s ions.tpr -p topol.top -o 1a1m_salgado.gro -pname NA -nname CL -neutral -conc 0.150
```

**Passo 6 - Minimizando a energia**

Agora, vamos realizar a etapa de minimização de energia do sistema. Isso é importante para remover comprometedores contatos de van der Waals inseridos pela adição das moléculas de água.

Crie um arquivo chamado “minim.mdp” e adicione o seguinte conteúdo:

```
; minim.mdp - used as input into grompp to generate em.tpr
integrator          = steep             ; Algorithm (steep = steepest descent minimization)
emtol               = 1000.0            ; Stop minimization when the maximum force < 1000.0 kJ/mol/nm
emstep      = 0.01      ; Energy 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
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 (yes/no)
```

Agora execute o comando:

```
gmx grompp -f minim.mdp -c 1a1m_salgado.gro -p topol.top -o em.tpr
```

Por fim, rode:

```
gmx mdrun -v -s em.tpr -deffnm em
```

**Passo 7 - Equilibrando o solvente e os íons**

Agora, vamos equilibrar o solvente e os íons ao redor do peptídeo.

Crie um arquivo chamado nvt.mdp e adicione o seguinte texto:

```
define              = -DPOSRES      ; position restrain the protein
; Run parameters
integrator          = md            ; leap-frog integrator
nsteps              = 50000         ; 2 * 50000 = 100 ps
dt                  = 0.002                 ; 2 fs
; Output control
nstxout             = 500           ; save coordinates every 1.0 ps
nstvout             = 500           ; save velocities every 1.0 ps
nstenergy           = 500           ; save energies every 1.0 ps
nstlog              = 500           ; update log file every 1.0 ps
; Bond parameters
continuation            = no                ; first dynamics run
constraint_algorithm    = lincs         ; holonomic constraints
constraints             = all-bonds         ; all bonds (even heavy atom-H bonds) constrained
lincs_iter              = 1                 ; accuracy of LINCS
lincs_order             = 4                 ; also related to accuracy
; Neighborsearching
cutoff-scheme   = Verlet
ns_type             = grid          ; search neighboring grid cells
nstlist             = 10            ; 20 fs, largely irrelevant with Verlet
rcoulomb            = 1.0           ; short-range electrostatic cutoff (in nm)
rvdw                = 1.0           ; short-range van der Waals cutoff (in nm)
; Electrostatics
coulombtype         = PME           ; Particle Mesh Ewald for long-range electrostatics
pme_order           = 4             ; cubic interpolation
fourierspacing      = 0.16    ; grid spacing for FFT
; Temperature coupling is on
tcoupl              = V-rescale             ; modified Berendsen thermostat
tc-grps             = Protein Non-Protein       ; two coupling groups - more accurate
tau_t               = 0.1       0.1         ; time constant, in ps
ref_t               = 300       300         ; reference temperature, one for each group, in K
; Pressure coupling is off
pcoupl              = no            ; no pressure coupling in NVT
; Periodic boundary conditions
pbc                 = xyz               ; 3-D PBC
; Dispersion correction
DispCorr            = EnerPres          ; account for cut-off vdW scheme
; Velocity generation
gen_vel             = yes           ; assign velocities from Maxwell distribution
gen_temp            = 300           ; temperature for Maxwell distribution
gen_seed            = -1            ; generate a random seed
```

Agora, execute o comando:

```
gmx grompp -f nvt.mdp -c em.gro -r em.gro -p topol.top -o nvt.tpr
```

e depois:

```
gmx mdrun -v -deffnm nvt
```

Isso irá levar aproximadamente 5 min para executar. Esse passo serve para estabilizar a temperatura do sistema. Agora, vamos estabilizar a pressão.

Crie um arquivo chamado “npt.mdp” e coloque o seguinte texto:

```
define              = -DPOSRES      ; position restrain the protein
; Run parameters
integrator          = md            ; leap-frog integrator
nsteps              = 50000         ; 2 * 50000 = 100 ps
dt                  = 0.002                 ; 2 fs
; Output control
nstxout             = 500           ; save coordinates every 1.0 ps
nstvout             = 500           ; save velocities every 1.0 ps
nstenergy           = 500           ; save energies every 1.0 ps
nstlog              = 500           ; update log file every 1.0 ps
; Bond parameters
continuation            = yes               ; Restarting after NVT
constraint_algorithm    = lincs         ; holonomic constraints
constraints             = all-bonds         ; all bonds (even heavy atom-H bonds) constrained
lincs_iter              = 1                 ; accuracy of LINCS
lincs_order             = 4                 ; also related to accuracy
; Neighborsearching
cutoff-scheme   = Verlet
ns_type             = grid          ; search neighboring grid cells
nstlist             = 10    ; 20 fs, largely irrelevant with Verlet scheme
rcoulomb            = 1.0           ; short-range electrostatic cutoff (in nm)
rvdw                = 1.0           ; short-range van der Waals cutoff (in nm)
; Electrostatics
coulombtype         = PME               ; Particle Mesh Ewald for long-range electrostatics
pme_order           = 4             ; cubic interpolation
fourierspacing      = 0.16          ; grid spacing for FFT
; Temperature coupling is on
tcoupl              = V-rescale             ; modified Berendsen thermostat
tc-grps             = Protein Non-Protein       ; two coupling groups - more accurate
tau_t               = 0.1       0.1         ; time constant, in ps
ref_t               = 300       300         ; reference temperature, one for each group, in K
; Pressure coupling is on
pcoupl                  = Parrinello-Rahman     ; Pressure coupling on in NPT
pcoupltype              = isotropic             ; uniform scaling of box vectors
tau_p                   = 2.0                       ; time constant, in ps
ref_p                   = 1.0                       ; reference pressure, in bar
compressibility     = 4.5e-5        ; isothermal compressibility of water, bar^-1
refcoord_scaling    = com
; Periodic boundary conditions
pbc                 = xyz           ; 3-D PBC
; Dispersion correction
DispCorr            = EnerPres      ; account for cut-off vdW scheme
; Velocity generation
gen_vel             = no            ; Velocity generation is off
```

Execute:

```
gmx grompp -f npt.mdp -c nvt.gro -r nvt.gro -t nvt.cpt -p topol.top -o npt.tpr
```

E por fim:

```
gmx mdrun -v -deffnm npt
```

Novamente, esse código levará aproximadamente 5 min para ser executado.

### Passo 8 - Executando a dinâmica

Agora podemos finalmente executar a dinâmica.

Crie um arquivo chamado “md.mdp” e adicione o seguinte texto:

```
; Run parameters
integrator          = md                ; leap-frog integrator
nsteps = 100000
dt = 0.002
; Output control
nstxout                     = 5000                  ; save coordinates every 10.0 ps
nstvout                     = 5000                  ; save velocities every 10.0 ps
nstenergy                   = 5000                  ; save energies every 10.0 ps
nstlog                      = 5000                  ; update log file every 10.0 ps
nstxout-compressed  = 5000      ; save compressed coordinates every 10.0 ps
                        ; nstxout-compressed replaces nstxtcout
compressed-x-grps   = System    ; replaces xtc-grps
; Bond parameters
continuation                = yes                   ; Restarting after NPT
constraint_algorithm    = lincs             ; holonomic constraints
constraints                 = all-bonds             ; all bonds (even heavy atom-H bonds) constrained
lincs_iter                  = 1                     ; accuracy of LINCS
lincs_order                 = 4                     ; also related to accuracy
; Neighborsearching
cutoff-scheme   = Verlet
ns_type             = grid              ; search neighboring grid cells
nstlist             = 10        ; 20 fs, largely irrelevant with Verlet scheme
rcoulomb            = 1.0               ; short-range electrostatic cutoff (in nm)
rvdw                = 1.0               ; short-range van der Waals cutoff (in nm)
; Electrostatics
coulombtype         = PME                   ; Particle Mesh Ewald for long-range electrostatics
pme_order           = 4                 ; cubic interpolation
fourierspacing      = 0.16              ; grid spacing for FFT
; Temperature coupling is on
tcoupl              = V-rescale                 ; modified Berendsen thermostat
tc-grps             = Protein Non-Protein       ; two coupling groups - more accurate
tau_t               = 0.1       0.1             ; time constant, in ps
ref_t               = 300       300             ; reference temperature, one for each group, in K
; Pressure coupling is on
pcoupl                      = Parrinello-Rahman     ; Pressure coupling on in NPT
pcoupltype                  = isotropic             ; uniform scaling of box vectors
tau_p                       = 2.0                           ; time constant, in ps
ref_p                       = 1.0                           ; reference pressure, in bar
compressibility     = 4.5e-5            ; isothermal compressibility of water, bar^-1
; Periodic boundary conditions
pbc                 = xyz               ; 3-D PBC
; Dispersion correction
DispCorr            = EnerPres          ; account for cut-off vdW scheme
; Velocity generation
gen_vel             = no                ; Velocity generation is off
```

Altere a linha “nsteps = 100000” para adicionar mais passos. Isso é equivalente a 200 ps.

Agora, execute o comando:

```
gmx grompp -f md.mdp -c npt.gro -r npt.gro -t npt.cpt -p topol.top
-o md.tpr
```

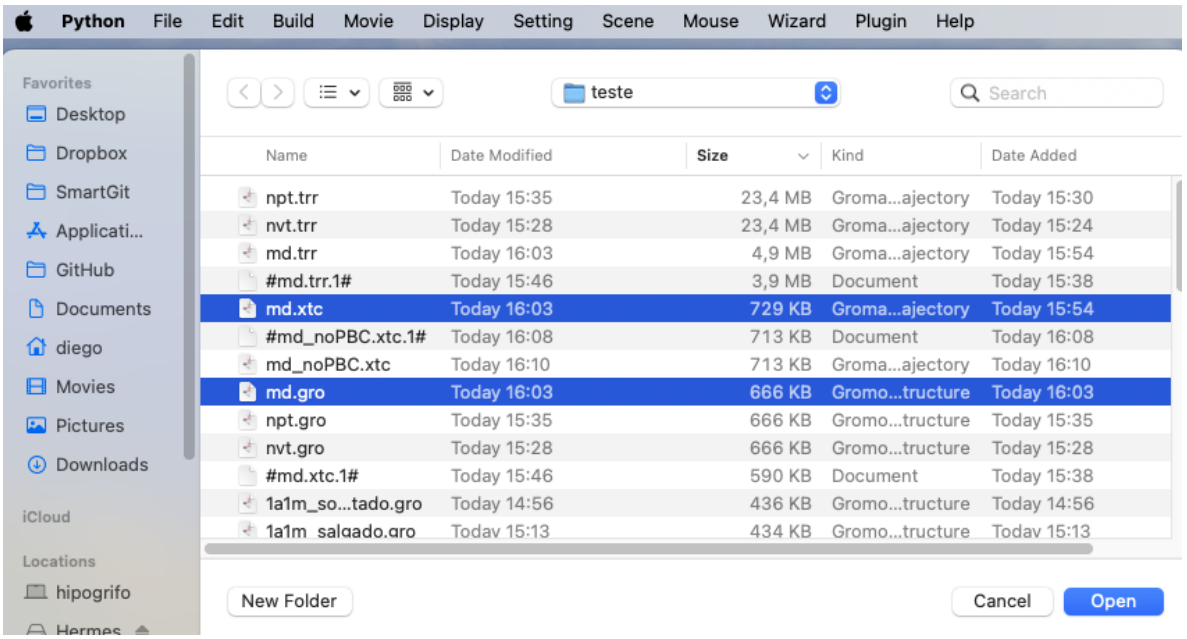
e por fim, ative a dinâmica rodando:

```
gmx mdrun -v -deffnm md
```

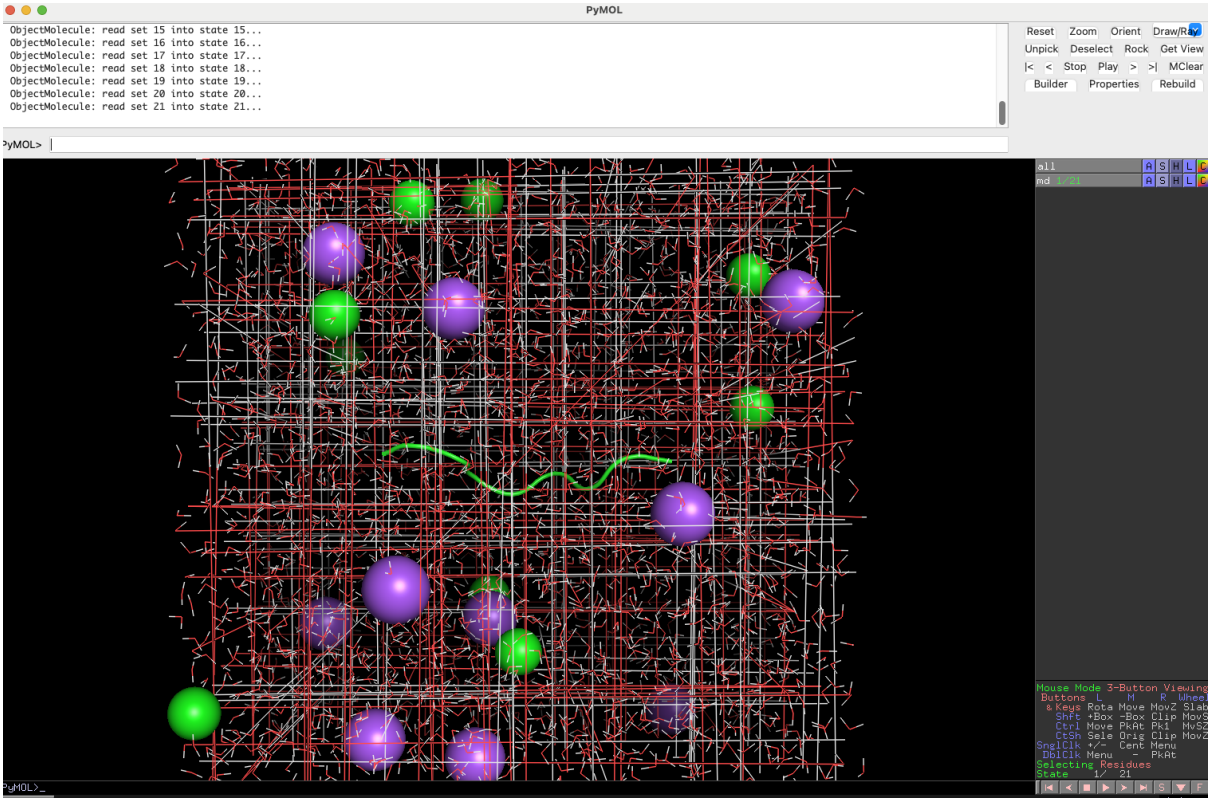
A execução total leva em torno de 10 minutos para 200ps. Pronto!

**Passo 9 - Visualizando o resultado**

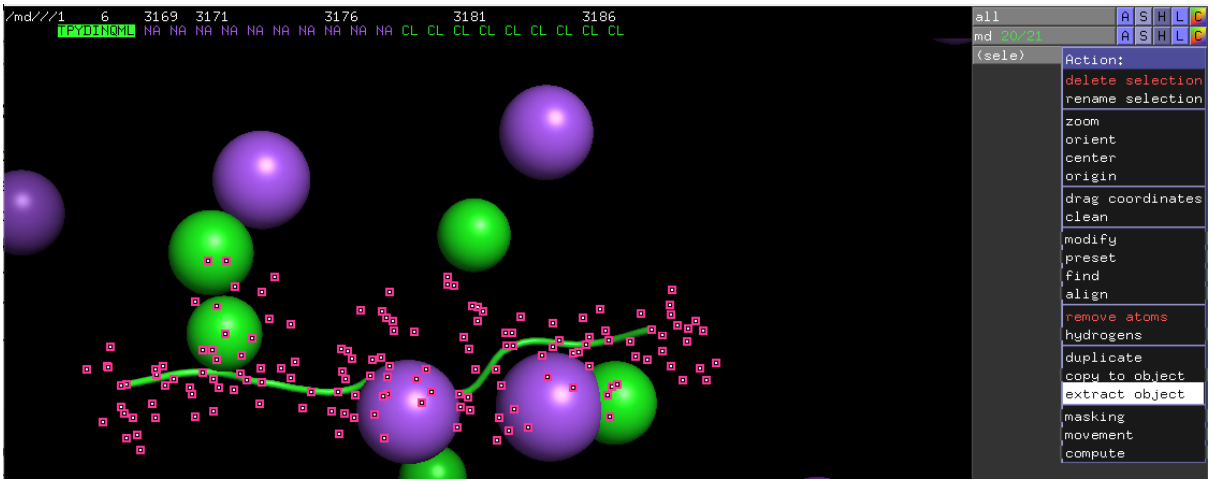
Ao terminar, abra o PyMOL e carregue os seguintes arquivos:



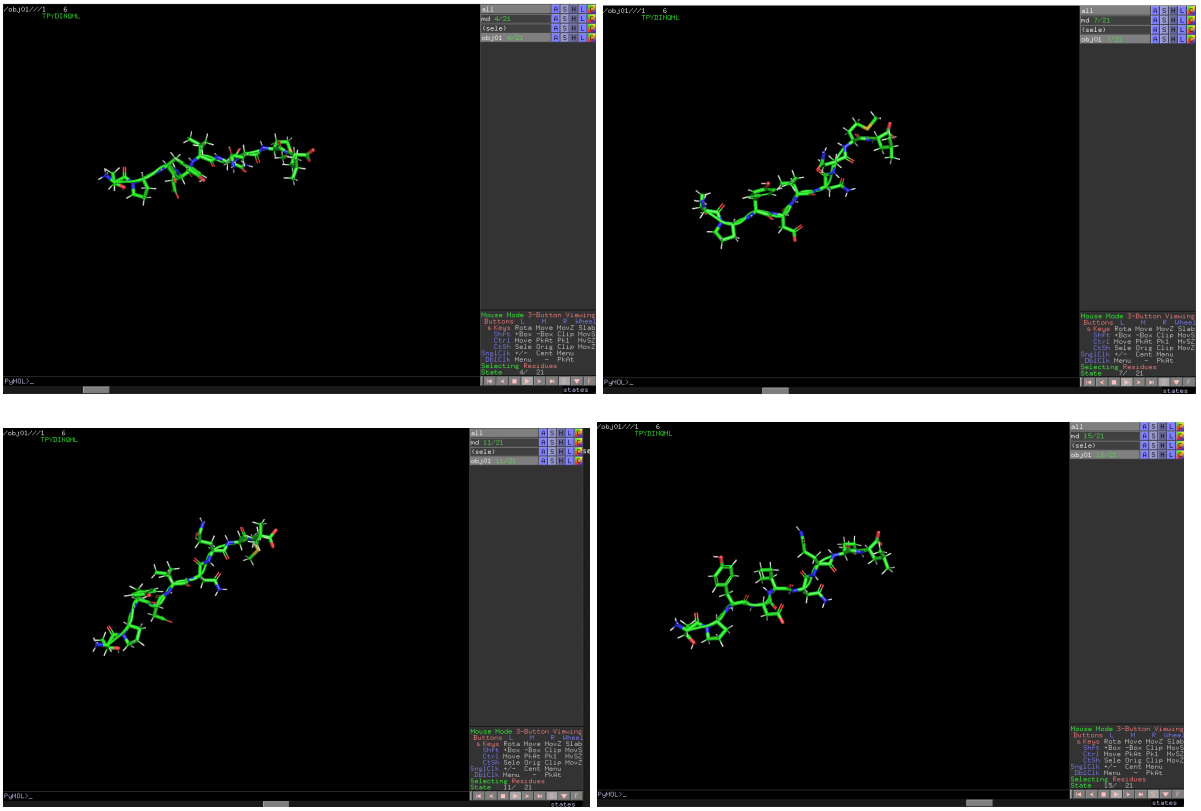

Você verá a seguinte estrutura:




Agora, clique clique no “S” no canto inferior direito. A sequência do peptídeo irá aparecer. Selecione o peptídeo e vá em (sele) > A > extract object:




Visualize como sticks e clique no botão play > no canto inferior direito. Você verá o peptídeo “dançando”:



Um peptídeo armazenado em um arquivo PDB com tamanho de 6 KB irá gerar um total de 60.000 KB (60MB) de dados de dinâmica (10.000x mais). Por isso tenha espaço de armazenamento disponível antes de executar uma dinâmica.

**Passo 10 - Visualizando os gráficos de RMSD e RMSF**

Opcionalmente, crie um arquivo chamado “gera_grafico.sh” e adicione os comandos:

```
gmx trjconv -s md.tpr -f md.xtc -b 0 -e 10000 -o md_noPBC.xtc -pbc mol -center -ur compact << eof
1
0
eof
gmx rms -s md.tpr -f md_noPBC.xtc -b 0 -e 10000 -o rmsd.xvg -tu ns << eof
4
4
eof
echo 4 | gmx rmsf -s md.tpr -f md_noPBC.xtc -b 0 -e 10000 -o rmsf_residue.xvg -res
```

Execute com:

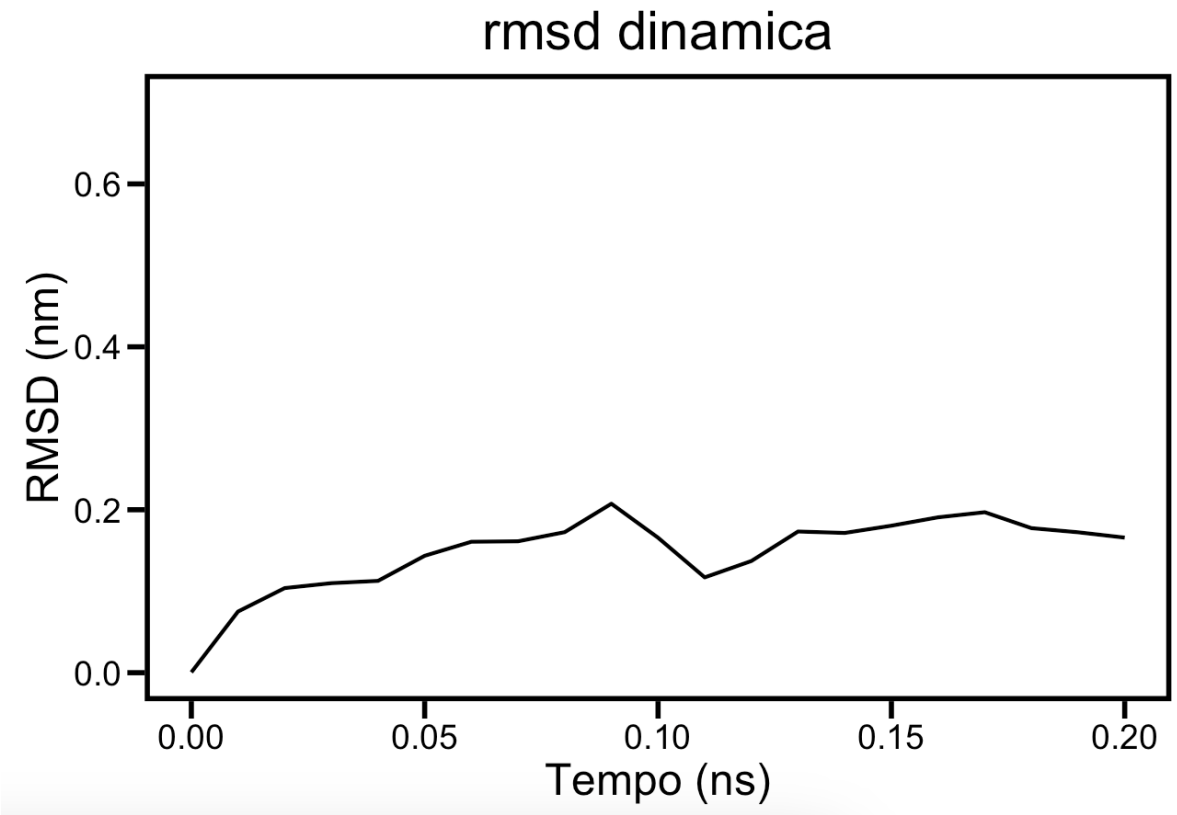
```
sh gera_grafico.sh
```

Dois arquivos serão criados:

- rmsf_residue.xvg
- rmsd.xvg

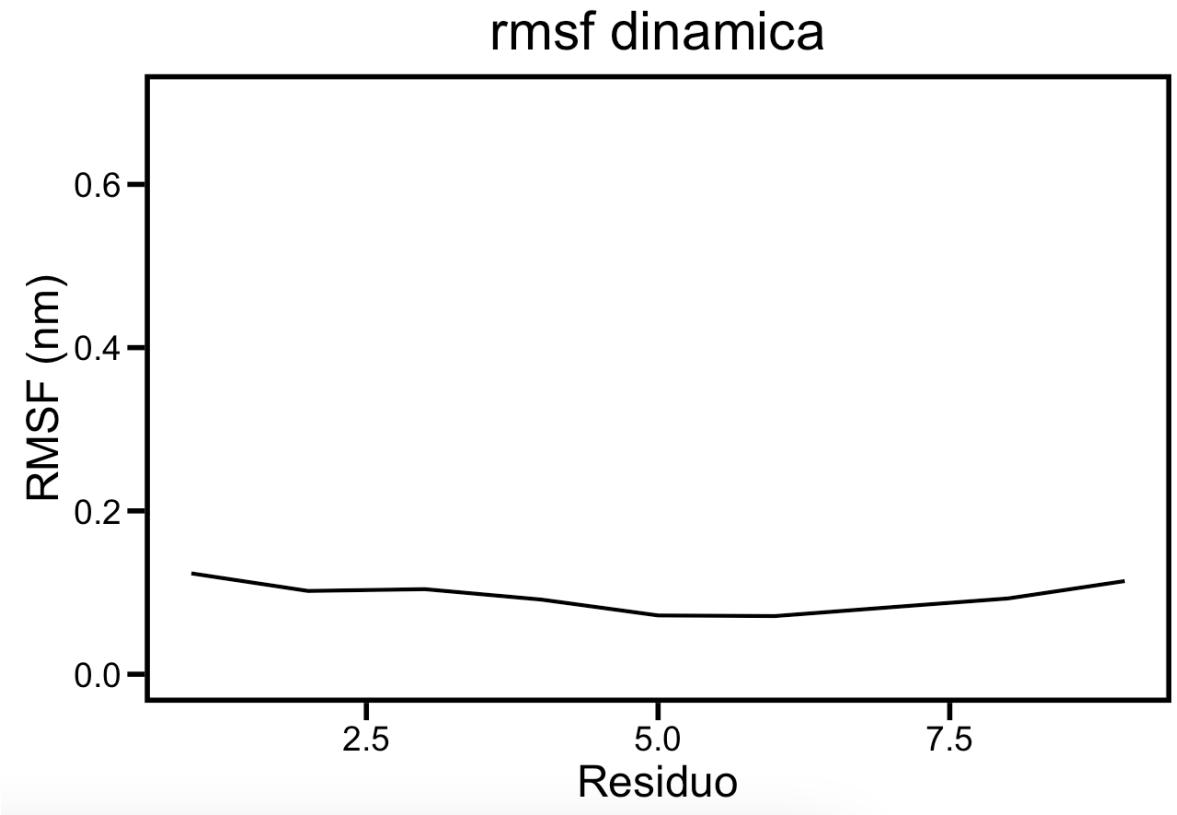
Agora, vamos criar um gráfico usando R. Execute o seguinte código no RStudio:

```
library(ggplot2)

rmsd = read.table(file = "rmsd.xvg", sep = "", dec = ".")
ggplot(rmsd, aes(x = V1, y = V2)) +
  geom_line(size = 0.75) +
  xlab("Tempo (ns)") +
  ylab("RMSD (nm)") +
  labs(title ="rmsd dinamica") +
  theme(axis.title = element_text(size = 18),
        axis.text = element_text(size = 14, colour = "black"),
        panel.border = element_rect(size = 2, fill = NA),
        panel.background = element_rect(fill = "white"),
        panel.grid = element_blank(),
        axis.ticks = element_line(colour = "black", size = 1),
        axis.ticks.length.y = unit(.25, "cm"),
        axis.ticks.length.x = unit(.25, "cm"),
        plot.margin = margin(0.5,1,0.5,0.5, "cm"),
        plot.title = element_text(size = rel(2), hjust = 0.5)) +
  coord_cartesian(ylim = c(0, 0.7))
ggsave("rmsd.png", dpi = 600, height = 5, width = 8)
```




Agora, para ver o gráfico de RMSF, use:

```
library(ggplot2)

rmsf = read.table(file = "rmsf_residue.xvg", sep = "", dec = ".")
ggplot(rmsf, aes(x = V1, y = V2)) +
  geom_line(size = 0.75) +
  xlab("Residuo") +
  ylab("RMSF (nm)") +
  labs(title ="rmsf dinamica") +
  theme(axis.title = element_text(size = 18),
        axis.text = element_text(size = 14, colour = "black"),
        panel.border = element_rect(size = 2, fill = NA),
        panel.background = element_rect(fill = "white"),
        panel.grid = element_blank(),
        axis.ticks = element_line(colour = "black", size = 1),
        axis.ticks.length.y = unit(.25, "cm"),
        axis.ticks.length.x = unit(.25, "cm"),
        plot.margin = margin(0.5,1,0.5,0.5, "cm"),
        plot.title = element_text(size = rel(2), hjust = 0.5)) +
  coord_cartesian(ylim = c(0, 0.7))
ggsave("rmsf.png", dpi = 600, height = 5, width = 8)
```


rmsf dinamica
RMSF (nm)
0.6
0.4
0.2
0.0
2.5
5.0
7.5
Residuo