使用 LightDock 和 MolAICal 进行蛋白质-肽分子对接教程—也适用于蛋白质-蛋白质对接

Qifeng Bai
电子邮箱: molaical@yeah.net
主页: https://molaical.github.io 或 https://molaical.gitlab.io

1. 引言

MolAICal 可以使用开源的 LightDock [1] 软件(https://lightdock.org)并通过 MM/GBSA 或 MM/PBSA 方法进一步提高生物大分子对接的准确性。LightDock 是一个开源(GPL-3.0 许可证)的对接框架，用于蛋白质-蛋白质、蛋白质-肽、蛋白质-RNA 和蛋白质-DNA 复合物。它特别擅长处理柔性且具有挑战性的案例(例如，瞬时相互作用、低亲和力复合物，或需要主链/侧链柔性的系统)，并作为一个可扩展平台用于测试新的评分函数、约束条件或优化策略。LightDock 采用了萤火虫群优化(GSO)算法 [2]，这是一种最初为多模态函数优化开发的受生物启发的群体智能方法。本教程使用胰高血糖素样肽-1 受体(GLP-1R)和艾塞那肽(exendin-4)(第一个获 FDA 批准的 GLP-1R 激动剂)作为示范实例(PDB ID: 7LLL.pdb)。

本教程仅适用于在 Linux 操作系统上完成！ 只有部分软件可以在 Windows 上完成。用户可以通过 SSH shell 或其他工具在 Linux 和 Windows 之间传输文件，以便更好地观察运行结果。

2. 材料

2.1. 软件需求

1) MolAICal: https://molaical.github.io or https://molaical.gitlab.io
2) NAMD (CPU 版本): https://www.ks.uiuc.edu/Research/namd/
   (注意: 本教程可以使用 NAMD 2.x、3.x 或更高版本运行。例如，如果您使用 NAMD 3.x 版本，命令"namd3"将替代本教程中的命令"namd2"。对于更高版本的 NAMD，用户可以使用与前面示例类似的替换方式。)
3) PyMol: https://github.com/cgohlke/pymol-open-source-wheels
4) VMD: https://www.ks.uiuc.edu/Research/vmd

2.2. 示例文件

1) 所有必要的教程文件可从以下网址下载:
   https://gitee.com/molaical/tutorials/tree/master/022-protein_pp_docking

3. 操作流程

3.1. 分子准备

为了解决 Linux 中的库依赖问题，Linux 版本的 MolAICal (Windows 版本的 MolAICal 没有采用容器)采用了基于容器的方法。如果需要调用外部程序，建议在 MolAICal 容器内安装这些程序。如果不需要外部程序，可以忽略此步骤。本教程需要外部程序 NAMD 和 VMD，具体步骤如下：

1. 首先，将文件复制到 MolAICal 容器中

```
# 进入容器文件系统（将进入 "/root" 目录）
#> molaical.exe -eset shell in

# 将 VMD 和 NAMD 安装包从本地机器复制到容器中，'cp' 命令的第一部分（源路径）
# 位于本地主机，第二部分（目标路径）在容器内；VMD 和 NAMD 软件包可通过以下命
# 令移入容器。
#> cp /home/user/<本地文件> /root/soft

# 退出容器文件系统
#> exit
```

2. 其次，进入 MolAICal 容器的虚拟环境

```
#> molaical.exe -eset sys run molaical
```

注：molaical 是容器名称。

3. 在 MolAICal 容器虚拟环境中安装软件的方式与在本地主机上安装相同。以下以安装 VMD 和 NAMD 为例：

1) 安装 NAMD：

解压 NAMD 文件（假设解压后的文件夹名为 namdcpu），然后使用以下命令将其路径告知 MolAICal：

```
#> molaical.exe -call set -n NAMD -p "/root/soft/namdcpu/namd3"
```

注：请将上述 VMD 和 NAMD 的路径替换为您系统中的实际路径。-n 后面的 "VMD" 和 "NAMD"（大小写不敏感）是固定的标识符。为确保 MM/GBSA 结果的可重复性，建议使用 NAMD 的 CPU 版本，因为 CUDA 版本中的 seed 参数似乎对结果可重复性无效。

2) 安装 VMD：

按以下步骤操作：

◆ **解压 VMD 文件：**

```
#> tar -xzvf vmd-xxx.tar.gz
```

注：请将上述路径替换为您系统中的实际路径。

◆ **修改 VMD 解压目录中名为 configure 的文件中的安装路径：**

```
# 默认值：
$install_bin_dir="/usr/local/bin";
$install_library_dir="/usr/local/lib/$install_name";
```

```
# 修改为:
$install_bin_dir="/root/soft/vmd193/bin";
$install_library_dir="/root/soft/vmd193/lib/$install_name";
```

◆ **安装 VMD：**

```
#> cd vmd-xxx
#> ./configure LINUXAMD64
#> cd src
#> make install
```

注：请运行 `./configure` 并根据所用计算机选择正确的类型，此处为 `"LINUXAMD64"`。

◆ **然后使用以下命令将 VMD 路径告知 MolAICal：**

```
#> molaical.exe -call set -n VMD -p "/root/soft/vmd193/bin/vmd"
```

至此，NAMD 和 VMD 在 MolAICal 容器内的安装与配置已完成。为防止 MolAICal 出现问题时丢失已安装的程序（例如 VMD 和 NAMD），请参阅**附录 1** 中的第 **3 步**。

✧ **记得使用 exit 命令退出 MolAICal 虚拟环境，返回本地计算机进行计算（主要是为了省去文件拷贝步骤；在容器内运行也可行，但需手动将数据从本地计算机复制到 MolAICal 容器中）。**

```
#> exit
```

假设用户已使用以下说明在 MolAICal 中配置了 VMD 和 NAMD 的内部命令:

```
#> molaical.exe -call set -n VMD -p "/root/soft/vmd193/bin/vmd"
#> molaical.exe -call set -n NAMD -p "/root/soft/namdcpu/namd3"
```

注意: 请将上面 VMD 和 NAMD 的路径替换为用户系统上的实际路径。"-n"后面的字符串"VMD"和"NAMD"(不区分大小写)对于 VMD 和 NAMD 的路径是固定的。为了确保 MM/GBSA 结果的可重现性，建议使用 NAMD 的 CPU 版本，因为在 NAMD 的 CUDA 版本中，"seed"参数对可重现性似乎无效。请注意，Windows 版与 Linux 版的 MolAICal 配置存在差异：Linux 版本采用基于 udocker 容器的技术方案，需在容器内部完成设置，而 Windows 版本则无需此步骤。

打开教程材料文件夹"022-protein_pp_docking":

#> cd 022-protein_pp_docking

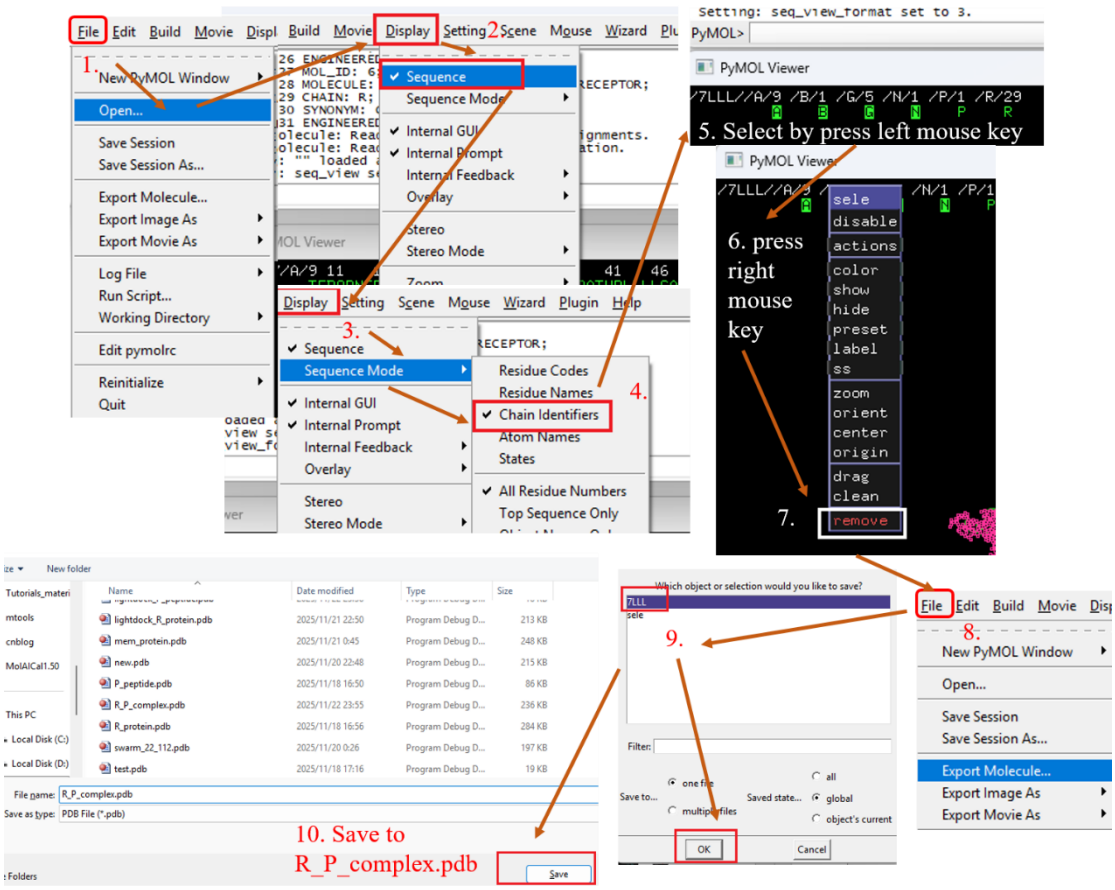

图 1

文件"R_P_complex.pdb" (链名为"P"和"R")、"R_protein.pdb" (链名为"R")和"P_peptide.pdb" (链名为"P") 是从胰高血糖素样肽-1 受体(GLP-1R)和艾塞那肽-4(exendin-4)（第一个获 FDA 批准的 GLP-1R 激动剂)的复合物(PDB ID: 7LLL.pdb）中通过 PyMol 提取的，

步骤如下: 将 PDB 文件 7LLL.pdb 加载到 PyMol 中，并使用**图 1** 的步骤分别保存"R_P_complex.pdb"、"R_protein.pdb"和"P_peptide.pdb"。

激动剂艾塞那肽-4 是一种肽。**如果用户想要进行蛋白质-蛋白质对接，请用蛋白质文件替换肽文件，因为肽可以被视为小蛋白质。**

选项: 有时，两个不同分子的链却有相同的名称(例如，都命名为链 A)。在这种情况下，用户可以通过 MolAICal 调用 VMD 分别重命名两个分子(例如，作为链 A 和链 B)。如果这种情况没有发生，请跳过此步骤和以下命令。命令如下:

```
#> molaical.exe -call run -c vmdargs -i -pdb R_protein.pdb -args none set sel [atomselect top all], \\\$sel set chain A, \\\$sel writepdb R_protein_A.pdb
```

```
#> molaical.exe -call run -c vmdargs -i -pdb P_peptide.pdb -args none set sel [atomselect top all], \\\$sel set chain B, \\\$sel writepdb P_peptide_B.pdb
```

注意:

- 它包括外部程序的所有命令行参数，但'**-i**'参数必须是最后指定的，而所有其他标识符(例如，'-call'、'-c'等)必须在'-i'之前。
- '**-pdb**'表示将 pdb 文件加载到 VMD 中。
- **-c:** 如果其值为"vmdargs"，它将在 Tk 控制台中运行 VMD 命令一次。
- '-args'后面的字符串表示可以在 VMD 的 Tk 控制台中运行的命令。
  - '-args'后的第一个参数可以是 VMD 的包名称或'none'。如果'-args'后第一个参数是'none'，它将省略加载包。例如，如果'-args'后第一个参数是'autopsf'，它将在 VMD 的 Tk 控制台中运行命令"package require autopsf"。如果'-args'后第一个参数是'none'，它将不会运行命令"package"。
  - '-args'后面的字符串包含一个可以在 VMD 的 Tk 控制台中执行的命令，每个逗号**','**之前；换句话说，逗号字符代替了在 VMD 的 Tk 控制台中输入每个命令并按 Enter 键的过程。这使得只需一个命令就可以执行多行命令。
  - 一些特殊字符，例如字符"$"，需要使用转义字符""。对于特殊字符，MolAICal 使用三个转义字符""(即"\")。

选项：LightDock 中的 DFIRE 评分函数需要标准氨基酸残基及其原子名称，因此在使用 DFIRE 评分函数进行对接前，必须检查并修复受体或配体。修复命令如下：

```
#> molaical.exe -call run -c sfile -i lmfix_rec.py -i protein.pdb -o protein_fixed.pdb
```

此处该 PDB 文件正常，无需通过上述选项处理。

在本教程中，未使用上述选项；使用文件"R_protein.pdb"(链名为"R")和"P_peptide.pdb"(链名为"P")，因为它们具有不同的链名。

3.2. 制作残基约束文件

为残基创建约束文件类似于识别活性口袋中的关键残基位点，分子将在其周围进行对接。用户可以根据文献介绍或直接根据经验自定义这些关键残基。

约束文件中残基的表示格式为:

```
Chain.Residue_Name.Residue_Number[.Residue_Insertion]
```

注意: 这里，Residue_Insertion 是一个可选的单字符标识符，附加到标准残基编号后面，通常用于 PDB 文件中区分插入残基而不重新编号整个序列。在生物信息学和结构生物学中，插入代码对于准确跟踪和描述特定蛋白质残基至关重要，特别是在处理序列变异或比较时。

当省略 Residue_Insertion 时，一般格式为:

```
Chain.Residue_Name.Residue_Number
```

或者

```
Chain.Residue_Name.Residue_Number P
```

注意: 标签"P"表示此约束是被动的(与没有标签的主动约束相反)。除非完全理解其含义，否则不推荐使用被动残基约束。

在这里，MolAICal 用于生成受体(蛋白质)在配体(肽)4.0Å 范围内的残基列表文件(命名为'rec.dat')作为关键残基，如下所示:

```
#> molaical.exe -call run -c sfile -i get_res.tcl -pdb 7LLL.pdb -args R P 4.0 rec.dat R
```

注意: 以下 1st、2nd、3rd 和 4th 分别表示第一个参数、第二个参数、第三个参数和第四个参数，依此类推。

- '-call': 与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。
- -c: 如果其值为"sfile"，它将运行 MolAICal 的 VMD 脚本，这里调用脚本文件'get_res.tcl'。
- '-pdb'表示将 pdb 文件加载到 VMD 中。
- 1st: 所选分子的链名；
- 2nd: 用于选择的参考分子的链名；
- 3rd: 在指定距离内从参考分子选择的所选分子的残基；
- 4th: 保存的文件名；
- 5th: 指定分子类型，'R'代表受体，'L'代表配体。

类似地，MolAICal 用于生成配体(肽)在受体(蛋白质)4.0Å 范围内的残基列表文件(命名为'lig.dat')作为关键残基，如下所示:

```
#> molaical.exe -call run -c sfile -i get_res.tcl -pdb 7LLL.pdb -args P R 4.0 lig.dat L
```

将'rec.dat'和'lig.dat'合并到名为'restraints.list'的约束文件中，如下所示:

```
#> molaical.exe -call run -c ecommand -i cat rec.dat lig.dat > restraints.list
```

在许多情况下，配体的具体细节是未知的。因此，本教程仅使用受体(第一个分子)的关键残基进行约束(并根据经验手动删除不必要的氨基酸)，命令如下:

```
#> molaical.exe -call run -c ecommand -i cp rec.dat restraints.list
```

3.3. 分子对接设置

首先，运行群体(swarm)生成的设置，如下所示:

```
#> molaical.exe -call run -c ldock -i lightdock3_setup.py R_protein.pdb P_peptide.pdb --noxt --noh --now -sp -rst restraints.list
```

- '-call': 与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。
- -c: 如果其值为"ldock"，它将通过调用"LightDock"运行分子对接(蛋白质-蛋白质、蛋白质-肽、蛋白质-DNA 或蛋白质-RNA)。
- "lightdock3_setup.py"后的第一个和第二个参数分别是受体和配体文件。
- --noxt: 如果启用此选项，LightDock 将忽略 OXT 原子。这对于几种不理解这种特殊类型原子的评分函数很有用。
- --noh: 如果启用此选项，LightDock 将忽略氢原子。这对于 dfire、fastdfire 或 dfire2 评分函数(以及其他)相关。仅删除以 H 字符开头的原子，因此不识别 1H 等类型。
- --now: 如果启用此选项，晶体水分子将被删除。
- --anm: 如果启用此选项，将激活 ANM 模式，并使用 ANM(通过 ProDy)建模主链柔性。该参数通过 ANM（使用 ProDy）启用骨架柔性建模，可能导致对接结果的结构畸变。**除非实施更优构象采样（如能量最小化），否则应避免使用。**替代方案：预生成多重构象以规避对接过程中的骨架柔性问题。
- -membrane: 启用时，此标志考虑受体伙伴沿 Z 轴对齐，并过滤掉与跨膜域不兼容的群体。为此，它使用提供的受体约束残基。
- -transmembrane: 参数"-transmembrane"具有与"-membrane"相反的功能。启用时，此标志考虑受体伙伴沿 Z 轴对齐，并过滤掉不在膜内的群体。
- -sp: 如果启用(默认为 False)，将在 init 文件夹中写入调试额外文件。
- -r, -rst restraints_file: 如果提供了 restraints_file，在设置和模拟过程中将考虑残基约束。如果没有提供 restraints_file，设置和模拟将集中在整个受体上。

- 它将生成名为'**init**'的文件夹，其中包含代表群体萤火虫的 PDB 格式分子，用于进一步对接。
- 文件'lightdock_R_protein.pdb'和'lightdock_P_peptide.pdb'是由 LightDock 从'R_protein.pdb'和'P_peptide.pdb'重新生成的

通过 PyMol 打开'lightdock_R_protein.pdb'和'init'文件夹中所有前缀为"starting_positions*"的 PDB 文件(见图 2):

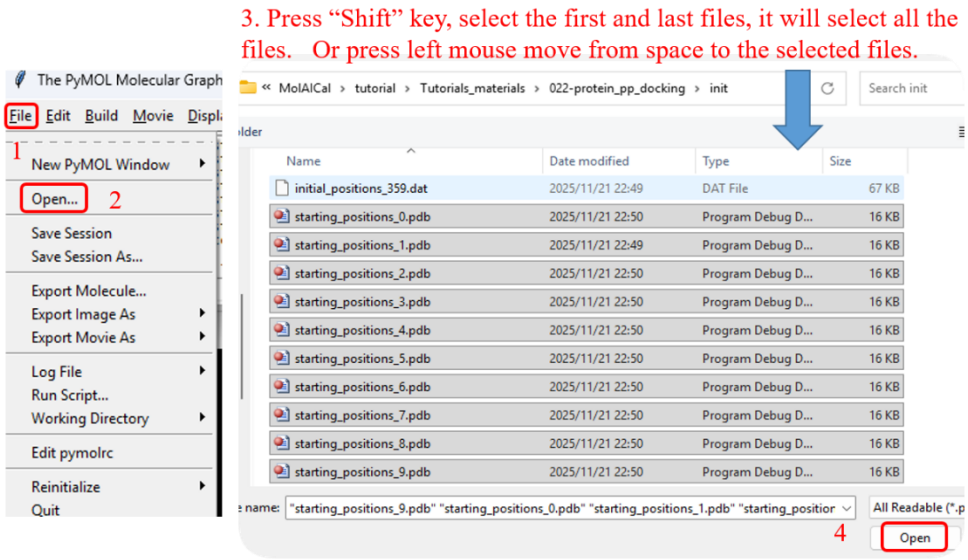

图 2

请参见图 3 的结果。很明显，许多萤火虫位于膜内区域，而配体肽位于膜蛋白 7 个跨膜螺旋中心面向细胞外的口袋中(如图 3 所示)。**这些膜内区域的萤火虫不仅不合理，而且在计算过程中消耗大量计算资源和时间，可能不利于对接结果。**

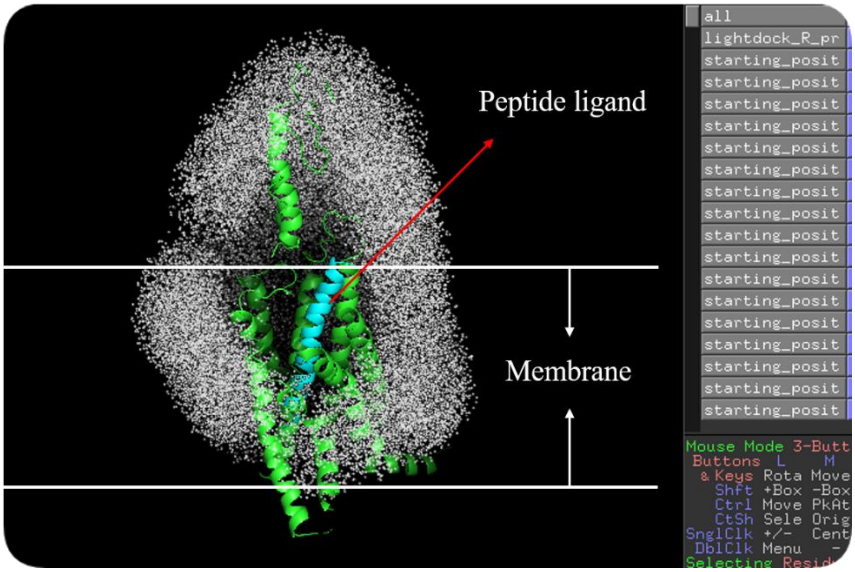

图 3

在这种情况下，MolAICal 提供了一种简单的膜生成方法，以最小化不合理膜内萤火虫的生成：

3.3.1 (选项) 在受体中设置膜

如果目标受体不是膜蛋白，请跳过此步骤！

3.3.1.1 (选项)

首先，使用 VMD 打开"lightdock_R_protein.pdb"。通过 VMD，可以在 x、y 和 z 方向上直观检查受体。确定膜蛋白的跨膜区域(面向细胞外侧)是否大致沿 z 轴对齐，如果已经对齐，则无需进一步操作。如果没有大致沿 z 轴对齐，请使用 MolAICal 调整其方向以与 z 轴对齐。命令如下:

```
#> molaical.exe -call run -c sfile -i align_axis.tcl -pdb R_protein.pdb -args protein 2 new.pdb
```

- '-call': 与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。
- -c: 如果其值为"sfile"，它将运行 MolAICal 的 VMD 脚本，这里调用脚本文件'align_axis.tcl'。
- '-pdb'表示将 pdb 文件加载到 VMD 中。
- 1st: 原子选择(与 VMD 的 Tk 控制台中"atomselect"命令相同)。如果有一个以上的字母(例如，protein 和 chain B)，它将使用逗号","表示空格" "(例如，protein,and,chain,B)；
- 2nd: 如果蛋白质水平对齐(平躺)，将 0 更改为 2；
- 3rd: 输出文件名；
- 4th: VMD 的"transaxis"所选轴，可以是 x、y、z 或""。""可以等于无字符，包括用于输入的空格。

文件"new.pdb"沿 z 轴对齐，将用于以下步骤。

3.3.1.2 一旦确保受体大致沿 z 轴对齐。在受体上识别两个用于膜生成的位置。

- ✧ 首先，在 VMD 中打开"R_protein.pdb"
- ✧ 其次，按住数字键'1'，然后用鼠标大致点击膜的内边缘和外边缘附近。这些位置可以根据实验要求进行调整，主要是限制膜之间萤火虫的生成。
- ✧ 获取这两个位置的 z 轴值，并将它们用作范围[min, max]的最小值和最大值(如图 4 所示)

这里，[min, max]=[-29.0, 10.0]。这些只是粗略的数字。用户可以根据研究目的进行调整。然后，运行以下命令进行膜生成:

```
#> molaical.exe -call run -c sfile -i gen_mem.tcl -pdb new.pdb -args protein -29.0 10.0 mempro.pdb
```

- ◆ '-call': 与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。
- ◆ -c: 如果其值为"sfile"，它将运行 MolAICal 的 VMD 脚本，这里调用脚本文件'gen_mem.tcl'。
- ◆ '-pdb'表示将 pdb 文件加载到 VMD 中。
- ◆ 1st: 原子选择(与 VMD 的 Tk 控制台中"atomselect"命令相同)。如果有一个以上的字母(例如，protein 和 chain B)，它将使用逗号","表示空格" "(例如，protein,and,chain,B)；
- ◆ 2nd: 膜生成的最小 z 轴；
- ◆ 3rd: 膜生成的最大 z 轴；
- ◆ 4th: 保存的输出文件名；
- ◆ 5th: delta 值用于[($max - $delta), $max]，表示用于确定沿 z 轴区域的最大和最小 x,y 值的范围值。这确保膜横截面中的膜颗粒不会位于膜蛋白内部。默认值: 3.0；
- ◆ 6th: 边界框的安全缓冲区(设置 0 以严格使用蛋白质限制)。默认值: -2.0；
- ◆ 7th: 膜填充(埃)。默认值: 15.0；
- ◆ 8th: 颗粒之间的最小间距。默认值: 3.5；
- ◆ 9th: 颗粒之间的最大间距。默认值: 6.0；
- ◆ 10th: 与蛋白质原子保持的距离(埃)。默认值: 4.0。

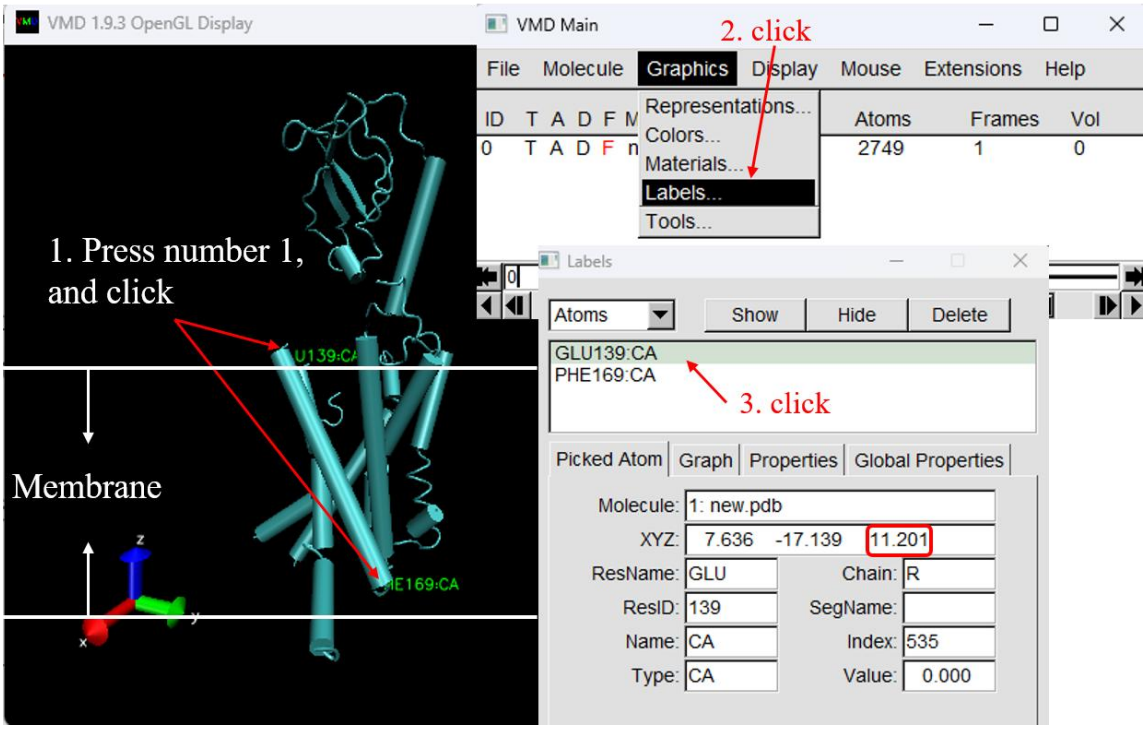

图 4

3.3.1.3 它将生成名为"mempro.pdb"的文件，其中包含受体和膜分子。

- **选项:** 删除先前生成的文件(如果不删除，将导致一些错误)。如果先前生成的文件不存在，请跳过此步骤。

```
#> molaical.exe molaical.exe -call run -c ecommand -i rm -rf init swarm_* lightdock_*
```

重新运行群体（swarm）生成的设置(添加参数: -membrane):

```
#> molaical.exe -call run -c ldock -i lightdock3_setup.py mempro.pdb P_peptide.pdb -membrane --noxt
--noh --now -sp -rst restraints.list
```

注意: **-transmembrane**: 参数"–transmembrane"具有与"–membrane"相反的功能。启用时，此标志考虑受体伙伴沿 Z 轴对齐，并过滤掉不在膜内的群体。

很明显，生成的群体数量比以前少。通过 PyMol 打开'lightdock_mempro.pdb'和'init'文件夹中所有前缀为"starting_positions*"的 PDB 文件(如图 2 所示的方法)。重新生成的群体萤火虫如图 5 所示；值得注意的是，膜之间没有生成群体，这是合理的。

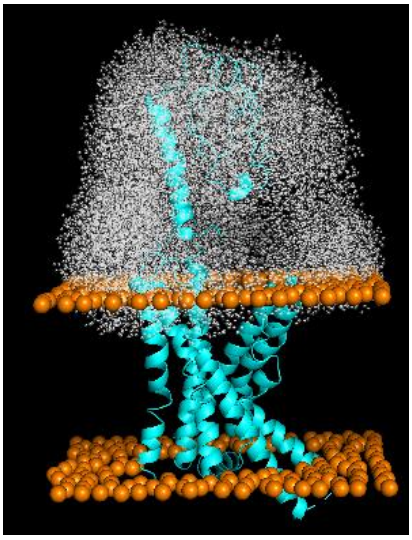

图 5

3.4. 分子对接模拟

其次，运行模拟:

```
#> molaical.exe -call run -c ldock -i lightdock3.py setup.json 100 -s fastdfire -c 12 -min
```

- '-call': 与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。
- -c: 如果其值为"ldock"，它将通过调用"LightDock"运行分子对接(蛋白质-蛋白质、蛋白质-肽、蛋白质-DNA 或蛋白质-RNA)。
- "lightdock3.py"后的第一个和第二个参数分别是设置步骤生成的配置文件和模拟的步数(这里是 100 步)。
- -s SCORING_FUNCTION: 使用此标志的默认评分函数(DFIRE，快速 C 实现 fastdfire)。接受评分函数的名称或包含多个评分函数名称和权重的文件。
- -c CORES: 默认情况下，使用硬件上可用的 CPU 核心总数来运行模拟，但可以通过此选项指定不同数量的 CPU 核心。
- -min: 启用时，-min 选项将使用 SciPy 的 scipy.optimize.fmin_powell 算法，在每个群优化步骤中对最高分粒子执行局部最小化。

3.5. 生成模型、聚类、排名和过滤

模拟完成后，MolAICal 调用脚本生成模型、聚类、排名和过滤结果，如下所示:

```
#> molaical.exe -call run -c sfile -i lrank.sh 1::=mempro.pdb 2::=P_peptide.pdb 3::=R 4::=P 5::=15
6::=restraints.list 13::=0.7
```

- '-call': 与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。
- '**::=**': 这里，它用于按指定顺序为参数赋值。
- '-c': 如果其值为"sfile"，它将运行 MolAICal 的 VMD 脚本。
- '1::=': 这是第一个输入分子(受体)的路径。必需参数。

- '2::=': 这是第二个输入分子(配体)的路径。必需参数。
- '3::=': 受体伙伴(第一个分子)上的链。默认值是'R'。
- '4::=': 配体伙伴(第二个分子)上的链。默认值是'P'。
- '5::=': 使用的 CPU 核心数。默认值是 12。
- '6::=': 包含约束的文件。默认值是'restraints.list'。
- '13::=': 过滤步骤中至少具有此分数的原生接触的结构。默认值是 0.4。为了识别与受体活性位点中更多关键残基结合的配体，此处将约束比例设置为 0.7。
- 其他参数使用默认值。更多信息，请参见 MolAICal 手册。

聚类和过滤过程完成后，将生成一个名为**"filtered"**的新目录。该目录保存所有符合默认 70%过滤标准的预测结构。在其中，将找到一个名为"rank_filtered.list"的文件，该文件根据 LightDock DFIRE (fastdfire)分数对这些结构进行排名，其中更高(更正)的分数表示更好的排名。

表 1

Name	Score	Percentages
swarm_75_173.pdb	31.008	0.714
swarm_60_55.pdb	30.682	0.786
s_swarm_22_3.pdb	25.149	0.400
swarm_97_172.pdb	19.769	0.786

注意: 与> **13::=0.7** 对应的百分比通过 len(contacts_receptor & restraints_receptor) + len(contacts_ligand & restraints_ligand) / (len(restraints_receptor) + len(restraints_ligand))计算。分子命名格式：swarm_{id_swarm}_{id_glowworm}.pdb，其中："id_swarm" 对应"setup"阶段生成的 swarm 文件夹名称；"id_glowworm" 指该 swarm 文件夹内 PDB 格式的分子名称。

将"R_P_complex.pdb"(天然结合构象)和表 1 中的分子加载到 PyMol 中，将"R_P_complex.pdb"与所需的分子对齐(对齐方法见图 6)。对接姿势明显重现了与参考结构("R_P_complex.pdb")的结构相似性。

表 1 显示了对接结果的分数。**表 1** 中的"s_swarm_22_3.pdb"候选化合物通过膜插入深度的变化生成，作为本教程中比较分析的案例。

此外，用户可以将最佳的一个或指定排名靠前的一些分子放入新文件夹"filtered/candidates"，命令如下:

```
#> molaical.exe -call run -c sfile -i mrank.py -ft 1
```

- -t: 复制到候选者的顶部分子数量(默认: 0)

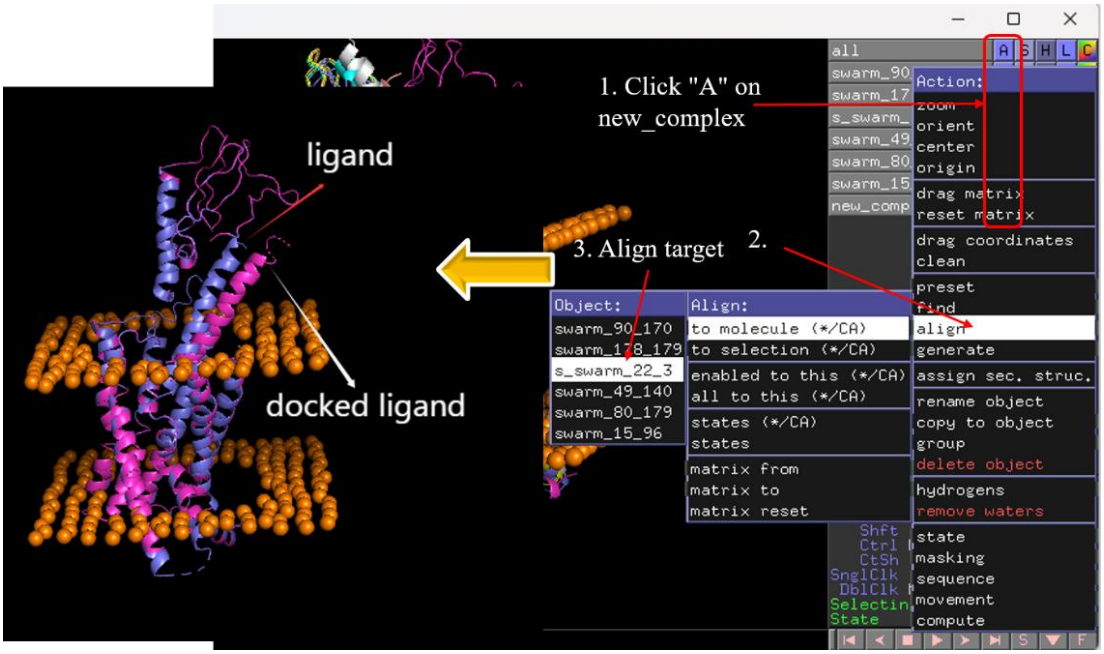

图 6

3.6. 用于进一步优化的 MM/GBSA 计算

MolAICal 计算 MM/GBSA 和 MM/PBSA 结合自由能。根据文献证据和经验验证，MM/GBSA 在蛋白质-蛋白质和蛋白质-配体对接系统中表现出优于 MM/PBSA 的性能。然而，这并不意味着 MM/GBSA 在所有系统中都普遍优于 MM/PBSA。我们建议研究人员根据其特定的生物分子系统和计算目标选择最佳方法。在这里，工作流程采用 MM/GBSA 计算对接姿势的结合亲和力，天然复合物结构'R_P_complex.pdb'作为比较 MM/GBSA 计算中的基准参考。

```
#> molaical.exe -call run -c sfile -i mmgbpbsa_batch.sh 1::=R 2::=P
```

- '-call': 与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。
- '::=': 这里，它用于按指定顺序为参数赋值。
- '-c': 如果其值为"sfile"，它将运行 MolAICal 的 VMD 脚本。
- '1::=': 第一个分子的链。默认值是'R'。
- '2::=': 第二个分子的链。默认值是'P'。
- 其他参数使用默认值。更多信息，请参见 MolAICal 手册。

注意：

1）首次运行时将自动生成参数文件**"cal_mmgbpbsa.sh"**。若该文件已存在，后续运行不会覆盖，以目录中现有版本为准。用户可自定义 **"cal_mmgbpbsa.sh"** 中的参数，此脚本遵循 MolAICal 格式，支持逐行批量执行 MolAICal 命令（命令行通常以 molaical.exe 开头）

2）MolAICal 目前仅支持标准蛋白质残基和核酸的 MM/GBSA 与 MM/PBSA 自动计算。对于小分子和非标准氨基酸，需额外使用 CHARMM 力场，可通过修改目录中的"cal_mmgbpbsa.sh"脚本并添加"-top"和"-ff"等参数实现。同时，如果需要 MM/PBSA 计算，修改"cal_mmgbpbsa.sh"第 5 行的注释即可切换到 MM/PBSA 的计算。更多细节请参考 MolAICal 手册及对应的 MM/GBSA 和/MM/PBSA 教程。

对'./mmgbpbsa/candidates'中的所需分子进行排名或提取，命令如下:

1. 仅打印 MM/GB/PBSA 的排名结果

```
#>molaical.exe -call run -c sfile -i mrank.py
```

2. 将结果排名靠前的 2 个分子复制到'./mmgbpbsa/candidates'

```
#> molaical.exe -call run -c sfile -i mrank.py -t 2
```

- -t: 复制到候选者的顶部分子数量(默认: 0)

它将显示如下结果:

```
Name                G_binding (kcal/mol)     ±        SE          SD
-------------------------------------------------------------------------------------------------------------------------
R_P_complex.pdb          -48.4655        +/-       0.0140       0.0989
swarm_97_172.pdb         -37.9548        +/-       0.0408       0.2884
swarm_60_55.pdb          -37.1118        +/-       0.0227       0.1605
swarm_75_173.pdb         -36.8318        +/-       0.0047       0.0334
s_swarm_22_3.pdb         -11.4960        +/-       0.0164       0.1163
```

在一般情况下，G_binding 的值越负越好。"R_P_complex.pdb"是天然构象。这表明 MM/GBSA 是评估两个分子之间结合亲和力的良好方法。

注意：本教程提供详细的多步骤指导，**主要用于教学目的。**教程中的步骤已更新，但示例结果未同步刷新；若你的输出与教程不同，**以你的实际结果为准。**

若程序被异常终止（例如通过键盘快捷键 "Ctrl+C" 或其他非标准方式），**大量残留进程将持续失控运行。**为解决此问题，用户可执行以下命令，删除当前目录下所有关联进程：

```
#> molaical.exe -eset pdclean
```

重要提示：

- **重复执行：**需多次运行此命令以确保彻底清除相关进程。
- **严重警告：**该操作会同时终止同目录下其他正在运行的程序进程。
- 执行前必须严格验证：
  - 若当前目录存在多个运行中的程序，须明确确认哪些目录关联进程可安全删除。
  - 切勿在共享目录/生产环境目录中执行此命令，除非经过充分风险评估与明确验证。

参考文献

1. Jiménez-García, B.; Roel-Touris, J.; Romero-Durana, M.; Vidal, M.; Jiménez-González, D.; Fernández-Recio, J., LightDock: a new multi-scale approach to protein-protein docking. Bioinformatics 2018, 34 (1), 49-55.
2. Krishnanand, K. N.; Ghose, D., Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. Swarm Intelligence 2009, 3 (2), 87-124.

附录 1

在 MolAICal 容器内安装外部程序（推荐，适用于 Linux 版本的 MolAICal）

请注意，Windows 版与 Linux 版的 MolAICal 配置存在差异：Linux 版本采用基于 udocker 容器的技术方案，需在容器内部完成设置，而 Windows 版本则无需此步骤。

1. 首先，将文件复制到 MolAICal 容器中

```
# 进入容器文件系统（将进入 "/root" 目录）
#> molaical.exe -eset shell in

# 将 VMD 和 NAMD 安装包从本地机器复制到容器中，'cp' 命令的第一部分（源路径）
# 位于本地主机，第二部分（目标路径）在容器内；VMD 和 NAMD 软件包可通过以下命
# 令移入容器。
#> cp /home/user/<本地文件> /root/soft

# 退出容器文件系统
#> exit
```

2. 其次，进入 MolAICal 容器的虚拟环境

```
#> molaical.exe -eset sys run molaical
```

注：molaical 是容器名称。

3. 在 MolAICal 容器虚拟环境中安装软件的方式与在本地主机上安装相同。以下以安装 VMD 和 NAMD 为例：

1) 安装 NAMD：

解压 NAMD 文件（假设解压后的文件夹名为 namdcpu），然后使用以下命令将其路径告知 MolAICal：

```
#> molaical.exe -call set -n NAMD -p "/root/soft/namdcpu/namd3"
```

注：请将上述 VMD 和 NAMD 的路径替换为您系统中的实际路径。-n 后面的 "VMD" 和 "NAMD"（大小写不敏感）是固定的标识符。为确保 MM/GBSA 结果的可重复性，建议使用 NAMD 的 CPU 版本，因为 CUDA 版本中的 seed 参数似乎对结果可重复性无效。

2) 安装 VMD：

按以下步骤操作：

- **解压 VMD 文件：**

```
#> tar -xzvf vmd-xxx.tar.gz
```

注：请将上述路径替换为您系统中的实际路径。

- **修改 VMD 解压目录中名为 configure 的文件中的安装路径：**

```
# 默认值：
$install_bin_dir="/usr/local/bin";
$install_library_dir="/usr/local/lib/$install_name";
```

```
# 修改为:
$install_bin_dir="/root/soft/vmd193/bin";
$install_library_dir="/root/soft/vmd193/lib/$install_name";
```

◆ **安装 VMD：**

```
#> cd vmd-xxx
#> ./configure LINUXAMD64
#> cd src
#> make install
```

注：请运行 `./configure` 并根据所用计算机选择正确的类型，此处为 `"LINUXAMD64"`。

◆ **然后使用以下命令将 VMD 路径告知 MolAICal：**

```
#> molaical.exe -call set -n VMD -p "/root/soft/vmd193/bin/vmd"
```

至此，NAMD 和 VMD 在 MolAICal 容器内的安装与配置已完成。

✧ **记得使用 exit 命令退出 MolAICal 虚拟环境，返回本地计算机进行计算（主要是为了省去文件拷贝步骤；在容器内运行也可行，但需手动将数据从本地计算机复制到 MolAICal 容器中）。**

```
#> exit
```

3) 保存并加载已修改的容器（可选）

为保留容器内所做的更改，并避免日后重新安装软件（因为一旦容器被删除，所有数据都会丢失），可执行以下操作：

✧ **获取容器 ID 和名称：**

```
#> molaical.exe -eset sys ps
```

✧ **克隆该容器：**

```
#> molaical.exe -eset sys export --clone -o molaicalv2.tar molaical
```

注：`molaical` 是容器名称，也可使用容器 ID。如果报错，那可能是因为挂载的文件系统的权限问题，可以忽略。

✧ **假如需要，导入克隆的容器：**

```
#> molaical.exe -eset sys import --clone --name molaicalv2 molaicalv2.tar
```

✧ **将新导入的容器名称更改为默认容器名 `"molaical"`，原容器重命名为 `"bakmolaical"`：**

```
#> molaical.exe -eset sys rename molaical bakmolaical
#> molaical.exe -eset sys rename molaicalv2 molaical
```

注意：容器（动态）是从镜像（静态）生成的。软件安装等操作必须在动态容器中进行；如果克隆了该容器，恢复时仍基于原始底层镜像。

更多详情请参阅 MolAICal 用户手册，或运行以下命令获取帮助：

```
#> molaical.exe -eset sys --help
```