用强大的GROMACS分析工具分析VASP的动力学结果

之前的教程里我们介绍了一个 `XDATCAR_toolkit.py` 工具将 `VASP` 的 `AIMD` 的轨迹转成 `PDB` 的格式，再借用 `VMD` 和 `MD Analysis` 软件包分析了 `RDF`（径向分布函数）和 `RMSD`（均方根偏差）。但是VASP软件只输出了最后一帧的原子速度，因此无法得到与速度相关的性质，比如速度自相关函数等。我们借鉴樊哲勇老师博客中的方法(http://blog.sciencenet.cn/blog-3102863-1159419.html)，通过向后差分的方法近似得到每一帧的原子速度。原理如下：

$$v_t = \frac{x(t)-x(t-\Delta t)}{\Delta t}$$

这样就舍去了第一帧的速度。如果想要得到比较精确的原子速度，可以通过牛顿定律求得t时刻的速度v_t:

$$v_t\Delta t + \frac{1}{2}a\Delta t^2 = x(t+\Delta t) - x(t), \text{其中} a = \frac{F}{m}$$

而不同时刻的原子受力，原子位置在 `OUTCAR` 中均有输出。可惜的是， `PDB文件` 不能承载速度信息，因此我们用 `c++` 重新写了一个 `VASP2GRO` 程序，可以快速从 `OUTCAR` 中读取所需要的信息，并自动计算**第2帧到最后一帧**的原子位置和原子速度，输出到 `GROMACS` 的轨迹 `gro` 中。与之前的教程一样，我们对**周期性**进行了处理，以保证原子速度不会出现异常情况。 教程的使用如下,在 `linux` 的终端中输入：

```
chmod u+x VASP2GRO.exe
./VASP2GRO.exe
```

程序自动会读取 `OUTCAR` 的信息，并输出带有原子位置和原子速度的 `gro` 轨迹，一个 `伪的` 拓扑文件 `XDATCAR.top` 和一个 `伪的` 输入文件 `XDATCAR.mdp` 。这个不能用于进行分子动力学计算，但是对我们使用 `GROMACS` 的分析功能已经绰绰有余了。接下来我们就使用它强大的分析功能帮助我们分析 `VASP` 的动力学结果。首先需要安装 `GROMACS` ，如果你是 `Ubuntu` 或者 `Centos` 的用户，你可以使用 `apt-get install gromacs` 或者 `yum install gromacs` 在线安装，你也可以使用参考 `Sob` 的并行安装方法安装（http://sobereva.com/457）。安装后可以输入 `gmx` 或者 `gmx_mpi` 测试是否安装好。 接下来我们使用 `GROMACS` 的功能处理我们转化得到的轨迹。先使用 `trjconv` 功能将得到的 `gro` 文件转成二进制的 `trr` 文件。如果你是并行安装的，需要将 `gmx` 替换成 `gmx_mpi` 。

```
gmx trjconv -f XDATCAR.gro -o XDATCAR.trr
```

再使用预编译引擎将输入文件打包成一个输入 `tpr` 文件：

```
gmx grompp -f XDATCAR.mdp -c XDATCAR.gro -p XDATCAR.top -o em.tpr
```

接着我们使用 `GROMACS` 神奇的分组功能将 `H` 原子和 `O` 原子分成一组以便后续处理。

```
gmx make_ndx -f XDATCAR.gro
```

默认的只有`3`组，包含了所有的原子。我们输入`a H`就会添加一组由`H`原子构成的一组，输入`a O`添加`O`原子组，再输入`q`保存退出。生成的`index.ndx`就包含了每个组包含的原子的原子序号。如果想合并`O`原子组`H`原子组，可以输入`3 | 4`回车就会生成第`5`组。

```
Reading structure file
Going to read 0 old index file(s)
Analysing residue names:
There are:    24      Other residues
Analysing residues not classified as Protein/DNA/RNA/Water and splitting into
groups...

  0 System              :    24 atoms
  1 Other               :    24 atoms
  2 MOL                 :    24 atoms

 nr : group      '!': not  'name' nr name   'splitch' nr    Enter: list
groups
 'a': atom       '&': and  'del' nr         'splitres' nr   'l': list
residues
 't': atom type  '|': or   'keep' nr        'splitat' nr    'h': help
 'r': residue              'res' nr         'chain' char
 "name": group             'case': case sensitive           'q': save and
quit
 'ri': residue index

> a H

Found 16 atoms with name H

  3 H                   :    16 atoms

> a O

Found 8 atoms with name O

  4 O                   :     8 atoms

> q
```

我们尝试做`O原子`的速度自相关函数。此时需要带上`index.ndx`以区分不同的原子组

```
gmx velacc -f XDATCAR.trr -o vacf.xvg -n index.ndx
```

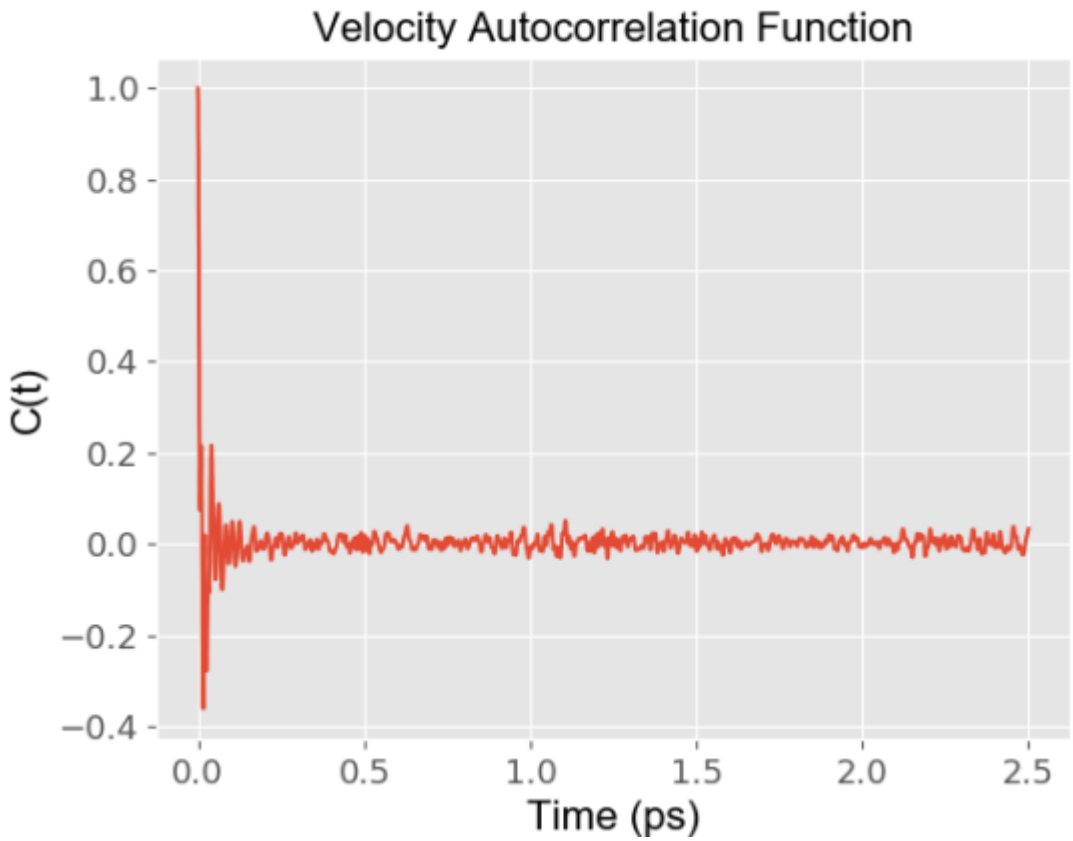
输入`4`来得到`O`原子的速度自相关函数。

```
Command line:
  gmx velacc -f XDATCAR.trr -o vacf.xvg -n index.ndx

Group     0 (         System) has    24 elements
Group     1 (          Other) has    24 elements
Group     2 (            MOL) has    24 elements
Group     3 (              H) has    16 elements
Group     4 (              O) has     8 elements
Select a group: 4
Selected 4: 'O'
trr version: GMX_trn_file (single precision)
```

得到的`vacf.xvg`是个含有时间，速度自相关函数值的数据文件，可以利用`xmgrace`作图，也可以手动提取后用`Origin Pro`作图。这里我们提供了一个脚本`xvg.py`，使用`Python`的`Matplotlib`作图：

```
import numpy as np
from matplotlib import pyplot as plt
import sys
import os

#------------------ FONT_setup ----------------------
font = {'family' : 'Arial',
    'color'  : 'black',
    'weight' : 'normal',
    'size' : 16.0,
    }


if len(sys.argv) <2:
    #print('')
    if sys.version[0]=='2':
        input=raw_input
    file=input("please with xvg file name-->")
else:
    file=sys.argv[1]
plt.style.use("ggplot")

if not os.path.exists(file):
    print("No file exists")
    sys.exit(0)

data=[]
with open(file,'r') as reader:
    for index,line in enumerate(reader):
        if "title" in line:
```

```
            title = eval(' '.join(line.split()[2:]))
        elif "xaxis" in line and   "label" in line:
            xlabel=eval(' '.join(line.split()[3:]))
        elif "yaxis" in line and   "label" in line:
            ylabel=eval(' '.join(line.split()[3:]))
        elif ((not line.startswith("@")) and (not line.startswith("&"))\
            and (not line.startswith("#"))):
            data.append(list(map(float,line.split())))
datas=np.array(data)
plt.plot(datas[:,0],datas[:,1:])

plt.ylabel(ylabel,fontdict=font)
plt.xlabel(xlabel,fontdict=font)
plt.xticks(fontsize=font['size']-3)
plt.yticks(fontsize=font['size']-3)
plt.title(title,fontdict=font)

plt.show()
```



`GROMACS`的强大不止于此，李继存老师的博客上有详细的教程可供学习。(https://jerkwin.github.io/GMX/GMXprg/)。

可执行文件`VASP2GRO.exe`和其源代码可以在我的`Github`下载。（https://github.com/tamaswells/VASP_script/tree/master/VASP2GRO ）