ZEO++대량시뮬레이션 코드

경로 및 변수 설정 코드

```
import pandas as pd
import numpy as np
import os
import sys
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
import subprocess
from joblib import Parallel, delayed

QMOF_cif_PATH = "\\\\wsl.localhost\\Ubuntu\\home\\dydtkddhk
dwk\\PYRASPA\\databases\\qmof_database\\relaxed_structures
\\relaxed_structures"
Output_save_PATH = "\\\\wsl.localhost\\Ubuntu\\home\\dydtkd
dhkdwk\\ZEO++\\0908QMOF"
# 표면적 계산에 사용된 탐침은 질소 분자의 TraPPE 모델에서 추출된 Lenn
ard-Jones σ 파라미터 값(직경 3.31 Å)을 사용 (CoREMOF 논문 참고)
COREMOF_PROBE_DIAMETER= 3.31 # Surface Area 측정시 필요한 Prob
e 직경 옹스트롬
COREMOF_PROBE_RADIUS =COREMOF_PROBE_DIAMETER/2   # 옹스트롬
ZEO_PATH = "\\\\wsl.localhost\\Ubuntu\\home\\dydtkddhkdwk
\\ZEO++\\zeo++-0.3".replace("\\", "/")
cifs = os.listdir(QMOF_cif_PATH)
cifs = [ x.replace(".cif", "") for x in cifs if x.endswith
(".cif")]
cifs_df = pd.DataFrame({"filename" : cifs})
```

커맨드 데이터프레임 만드는 코드

```
PORE_samples = cifs_df.copy()
isexistcif = []
for i in PORE_samples["filename"]:
    if i in cifs:
        isexistcif.append(True)
    else:
        isexistcif.append(False)
PORE_samples["isexistcif"] = isexistcif
PORE_samples = PORE_samples[PORE_samples["isexistcif"] == T
rue]
PORE_samples["cif_abspath"] = (QMOF_cif_PATH +"\\" +  PORE_
samples["filename"] + ".cif").str.replace("\\", "/").str.re
place("//wsl.localhost/Ubuntu", "")
PORE_samples["Output_save_PATH" ] = (Output_save_PATH +
"\\res\\"+ PORE_samples["filename"] + ".res").str.replace
("\\", "/").str.replace("//wsl.localhost/Ubuntu", "")
PORE_samples["command"] = (ZEO_PATH+"/network -ha -res " +
PORE_samples["Output_save_PATH"] + " " + PORE_samples["cif_
abspath"]).str.replace("\\", "/").str.replace("//wsl.localh
ost/Ubuntu", "")
PORE_samples = PORE_samples.reset_index()
PORE_samples.to_csv("0908QMOF/POREcommands.csv")

PORE_Vloume_samples = cifs_df.copy()
isexistcif = []
for i in PORE_Vloume_samples["filename"]:
    if i in cifs:
        isexistcif.append(True)
    else:
        isexistcif.append(False)
PORE_Vloume_samples["isexistcif"] = isexistcif
PORE_Vloume_samples = PORE_Vloume_samples[PORE_Vloume_sampl
es["isexistcif"] == True]
PORE_Vloume_samples["cif_abspath"] = (QMOF_cif_PATH +"\\" +
PORE_Vloume_samples["filename"] + ".cif").str.replace("\\",
"/").str.replace("//wsl.localhost/Ubuntu", "")
PORE_Vloume_samples["Output_save_PATH" ] = (Output_save_PAT
H + "\\volpo\\"+ PORE_Vloume_samples["filename"] + ".volp
```

```
o").str.replace("\\", "/").str.replace("//wsl.localhost/Ubu
ntu", "")
PORE_Vloume_samples["command"] = (ZEO_PATH+"/network -ha -v
olpo %s %s 10000 "%(COREMOF_PROBE_RADIUS,COREMOF_PROBE_RADI
US) + PORE_Vloume_samples["Output_save_PATH"] + " " + PORE_
Vloume_samples["cif_abspath"]).str.replace("\\", "/").str.r
eplace("//wsl.localhost/Ubuntu", "")
PORE_Vloume_samples = PORE_Vloume_samples.reset_index()
commands = PORE_Vloume_samples["command"]
PORE_Vloume_samples.to_csv("PORE_VOLUMEcommands.csv")

Surface_samples =cifs_df.copy()
isexistcif = []
for i in Surface_samples["filename"]:
    if i in cifs:
        isexistcif.append(True)
    else:
        isexistcif.append(False)
Surface_samples["isexistcif"] = isexistcif
Surface_samples = Surface_samples[Surface_samples["isexistc
if"] == True]
Surface_samples["cif_abspath"] = (QMOF_cif_PATH +"\\" +  Su
rface_samples["filename"] + ".cif").str.replace("\\", "/").
str.replace("//wsl.localhost/Ubuntu", "")
Surface_samples["Output_save_PATH" ] = (Output_save_PATH +
"\\sa\\"+ Surface_samples["filename"] + ".sa").str.replace
("\\", "/").str.replace("//wsl.localhost/Ubuntu", "")
Surface_samples["command"] = (ZEO_PATH+"/network -ha -sa %s
%s 2000 "%(COREMOF_PROBE_RADIUS,COREMOF_PROBE_RADIUS) + Sur
face_samples["Output_save_PATH"] + " " + Surface_samples["c
if_abspath"]).str.replace("\\", "/").str.replace("//wsl.loc
alhost/Ubuntu", "")
Surface_samples = Surface_samples.reset_index()
Surface_samples.to_csv("Surfacecommands.csv")

Accessible_volume_samples = cifs_df.copy()
isexistcif = []
for i in Accessible_volume_samples["filename"]:
```

```
    if i in cifs:
        isexistcif.append(True)
    else:
        isexistcif.append(False)
Accessible_volume_samples["isexistcif"] = isexistcif
Accessible_volume_samples = Accessible_volume_samples[Acces
sible_volume_samples["isexistcif"] == True]
Accessible_volume_samples["cif_abspath"] = (QMOF_cif_PATH
+"\\" +  Accessible_volume_samples["filename"] + ".cif").st
r.replace("\\", "/").str.replace("//wsl.localhost/Ubuntu",
"")
Accessible_volume_samples["Output_save_PATH" ] = (Output_sa
ve_PATH + "\\vol\\"+ Accessible_volume_samples["filename"]
+ ".vol").str.replace("\\", "/").str.replace("//wsl.localho
st/Ubuntu", "")
Accessible_volume_samples["command"] = (ZEO_PATH+"/network
-ha -vol %s %s 5000 "%(0.0000001,0.0000001) + Accessible_vo
lume_samples["Output_save_PATH"] + " " + Accessible_volume_
samples["cif_abspath"]).str.replace("\\", "/").str.replace
("//wsl.localhost/Ubuntu", "")
Accessible_volume_samples = Accessible_volume_samples.reset
_index()
Accessible_volume_samples.to_csv("Accessible_volume_comman
d.csv")
```

만든 커맨드들을 일괄 불러와서 wsl이나 리눅스 환경에서 subprocess 및 joblib을 이용해 대량 실시해주는 코드

```
import pandas as pd
import numpy as np
import os
import sys
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
import subprocess
from joblib import Parallel, delayed
```

```
# 함수 정의: 명령어를 실행하고 실패한 경우 예외 처리
def run_wsl_command(command, count):
    try:
        wsl_command_split = ['wsl'] + command.split()
        result = subprocess.run(wsl_command_split, capture_
output=True, text=True)
        return None  # 성공적으로 실행된 경우
    except Exception as e:
        return (command, count)  # 실패한 경우 명령어와 카운트 반
환

commands = pd.concat([
    pd.read_csv("./0908QMOF/Surfacecommands.csv")["comman
d"],
pd.read_csv("./0908QMOF/Accessible_volume_command.csv")["co
mmand"]
# ,pd.read_csv("./0908QMOF/PORE_VOLUMEcommands.csv")["comma
nd"]
,pd.read_csv("./0908QMOF/POREcommands.csv")["command"]])
commands =  pd.DataFrame(commands).reset_index(drop=True)
["command"]
    # 병렬 실행을 위한 설정
num_cores = 6  # 사용할 CPU 코어 수
# 명령어 리스트(commands)에서 병렬 처리 수행
results = Parallel(n_jobs=num_cores)(
    delayed(run_wsl_command)(command, idx) for idx, command
in enumerate(tqdm(commands, total = len(commands)))
)
# 실패한 명령어 모으기
failed_commands = [result for result in results if result i
s not None]
# 실패한 명령어 출력
print(f"총 실패한 명령어 수: {len(failed_commands)}")
for failed_command in failed_commands:
    print(f"Failed command: {failed_command}")
```