J. Korean Inst. Electr. Electron. Mater. Eng.
Vol. 35, No. 6, pp. 556-561 November 2022
DOI: https://doi.org/10.4313/JKEM.2022.35.6.3
ISSN 1226-7945(Print), 2288-3258(Online)

Tutorial Status Report

Pymatgen 패키지를 이용한 구조 생성 및 제일원리계산에의 적용

이대형, 서동화

울산과학기술원 에너지화학공학과

초록: 밀도범함수이론(density functional theory, DFT)이 등장한 이래로, 이를 재료과학에 적용하여 에너지 재료 및 반도체와 같은 전자재료들의 연구개발에 활발하게 활용되고 있다. 하지만 DFT 계산 프로그램을 실행할 때 필요한 입력 파일 생성 시 여러 가지 소재들에 대해 동일한 계산 조건을 맞춰 주고 파라미터들을 알맞게 설정해 줘야 올바른 계산 결과 비교가 가능한데, 이런 부분들에 대해 진입 장벽이 높다는 어려움이 있다. 이에 본 논문에서는 Python Materials Genomics (pymatgen) 파이썬 패키지를 이용해 분자 및 결정구조를 다루고 널리 사용되는 DFT 계산 프로그램인 Vienna Ab initio Simulation Package (VASP) 및 Gaussian 입력 파일 생성에 대해 소개하고자 한다. 이를 통해 해당 프로그램에 대한 전문적인 지식이 많지 않더라도 보다 일관적인 계산 조건에서 결과들을 손쉽게 수행할 수 있게 되기를 기대한다.

키워드: 파이썬, Pymatgen, 밀도범함수이론, 자동화, VASP, Gaussian

Creating Structure with Pymatgen Package and Application to the First-Principles Calculation

Dae-Hyung Lee and Dong-Hwa Seo

School of Energy and Chemical Engineering, Ulsan National Institute of Science and Technology(UNIST), Ulsan 44919, Korea

(Received September 1, 2022; Revised September 27, 2022; Accepted September 27, 2022)

Abstract: Computational material science as an application of Density Functional Theory (DFT) to the discipline of material science has emerged and applied to the research and development of energy materials and electronic materials such as semi-conductor. However, there are a few difficulties, such as generating input files for various types of materials in both the same calculating condition and appropriate parameters, which is essential in comparing results of DFT calculation in the right way. In this tutorial status report, we will introduce how to create crystal structures and to prepare input files automatically for the Vienna Ab initio Simulation Package (VASP) and Gaussian, the most popular DFT calculation programs. We anticipate this tutorial makes DFT calculation easier for the ones who are not experts on DFT programs.

Keywords: Python, Pymatgen, DFT, Automation, VASP, Gaussian

1. 서 론

■ 양자역학 이론이 등장한 이후 슈뢰딩거 방정식을 통해 원자 내 전자의 에너지 준위를 계산해 내는 것이 가능해졌으나, 다전자 원자의 경우, 그리고 원자의 개수가 많은 분자 혹은 벌크 소재들의 경우에는 그 계산이 너무 복잡해 양자역학에 기반한 이론적인 접근이 불가능

✉ **Dong-Hwa Seo; dseo@unist.ac.kr**

했다. 그러나 전자밀도 기반 근사를 통해 다전자 슈뢰딩거 방정식을 비교적 간단하게 만든 Kohn-Sham 방정식 및 밀도범함수이론(density functional theory, DFT)의 등장으로 인해 분자 및 결정질 소재들에 대해서도 양자역학 기반의 전산모사 기법이 가능하게 되었다 [1-3]. 이에 따라 전산재료과학 분야가 발전하기 시작했으며, 현재까지 소재의 물성 예측 및 스크리닝 [4], 전자구조 계산 [5] 등 여러 분야에 적용되고 있다.

- 대표적인 DFT 계산 프로그램으로는 Vienna Ab initio Simulation Package(VASP)가 있다. VASP는 평면파 바탕함수를 이용해 파동함수를 표현함으로써 고체 결정질 소재들의 계산에 특화되어 있음에도 단위 격자를 잡는 것에 따라 분자 계산도 충분히 해낼 수 있다는 장점이 있다. 하지만, 해당 프로그램 및 바탕 이론에 대해 익숙하지 않은 사람들에게는 계산에 필요한 입력 파일들을 생성하거나 DFT 계산 조건들인 INCAR 태그들을 설정하는 것에 진입장벽이 높다는 어려움이 있고, 이에 익숙한 사람들에게도 대량병렬 제일원리계산을 수행하는 경우 수많은 구조들에 대한 입력 파일들을 하나씩 만드는 것이 쉽지 않다는 어려움이 있다.
- 본 tutorial status report에서는 이러한 어려움을 해결하기 위해 Python Materials Genomics(pymatgen) 파이썬 패키지 [6]를 이용해 결정구조를 생성하고 다루는 법, 여러 가지 계산 및 functional들에 대해 자동으로 VASP 입력 파일을 생성하는 예제에 대해 다루고자 한다. 추가적으로, 분자구조에 대한 DFT 계산 프로그램인 Gaussian의 입력 파일 생성에 대해서도 간단히 소개하고자 한다. 이를 통해 숙련되지 않은 사람들도 좀 더 일관된 DFT 계산 조건으로 편리하고 효율적인 계산을 수행할 수 있기를 기대한다.

2. 본 문

2.1 Pymatgen 패키지 설치 및 POTCAR 셋업

- Pymatgen 패키지를 사용하기 위해서는 먼저 파이썬이 설치되어야 한다. 파이썬 설치 방법에는 여러 가지가 있지만, miniconda와 같은 conda 환경에서의 설치를 권장한다. 각 운영체제마다 miniconda 설치에 관한 내용은 관련 웹페이지 [7]에서 확인할 수 있다. Conda 환경 설치가 완료되면 pymatgen을 위한 가상환경 설정도 가능하며, base 환경에 pymatgen을 설치해도 무방하다. 하지만, base 환경에 pymatgen을 설치하는 경우 추후 다른 패키지들과 numpy 등의 버전 충돌을 일으킬 가능성이 있기 때문에, pymatgen이 설치될 가상환경을 구축해 설치하는 것을 권장한다. Pymatgen 패키지 설치에 관한 자세한 내용은 관련 웹페이지 [8]에서 확인할 수 있다.
- Pymatgen 패키지는 pip 및 conda를 이용해 아래의 명령어에 의해 간단하게 설치가 가능하며, pip보다는 conda를 이용해 설치하는 것을 권장한다.

```
pip install pymatgen
혹은
conda install --channel conda-forge pymatgen
```

Pymatgen을 설치한 후 VASP POTCAR 파일을 생성하기 위해서 아래의 명령어로 유사포텐셜 파일들의 위치를 입력해 줘야 한다.

```
pmg config -p '유사포텐셜 파일들이 위치한 경로' 'pymatgen의 POTCAR 파일 경로'
이후
pmg config --add PMG_VASP_PSP_DIR 'pymatgen의 POTCAR 파일 경로'
```

여기에서 '유사포텐셜 파일들이 위치한 경로'는 Ac, Ag 등의 원소 별 POTCAR 디렉토리들이 있는 경로이며, 'pymatgen의 POTCAR 파일 경로'는 pymatgen용 POTCAR 파일들이 생성될 경로로써 어떠한 경로를 입력해 주어도 무방하나, 최상위 경로에 디렉토리를 하나 따로 만들어 해당 경로를 사용하는 것을 권장한다.

2.2 분자 및 결정구조 생성 및 저장

- Pymatgen 패키지는 버전에 따라 사용법이 달라지거나 클래스 및 하위 메서드가 추가, 삭제되는 경우가 많다. 따라서 pymatgen 버전, 웹페이지 [9]를 수시로 확인하는 것을 권장한다. 본 tutorial status report에서는 2022.8.23 버전의 pymatgen 패키지를 사용하였으며, 소개된 기능 외의 여러 가지 기능들을 pymatgen documentation [10]에서 확인할 수 있다.
- Pymatgen에서 구조를 다루는 클래스는 크게 Molecule과 Structure가 있다. Molecule 클래스는 주기경계조건(periodic boundary condition, PBC)이 없는 분자구조를 다루며 분자를 구성하는 원자와 원자의 좌표만

을 가지고 만들 수 있고, Structure 클래스는 PBC가 존재하는 결정구조를 다루며 추가로 결정 격자에 대한 정보가 있어야 만들 수 있다. 그림 1, 2는 Molecule 및 Structure 클래스를 이용해 H_2O 구조와 NaCl 구조를 생성하는 예시를 보여준다. Molecule과 Structure 클래스는 생성한 구조를 똑같은 방법으로 다룰 수 있기 때문에, 이후부터는 Structure 클래스를 중심으로 설명하고자 한다.

- 그림 2에서의 예시와 같이 생성한 구조를 to 메서드를 사용해 파일로 저장할 수 있다.
  S는 Structure 클래스의 인스턴스, 즉 구조 데이터이며, filename은 저장할 구조 파일의 경로를, fmt는 구조 파일의 형식을 의미한다. Fmt 옵션으로는 cif, poscar, cssr, json, xsf, mcsqs, prismatic, yaml, fleur-inpgen 이 있다.
- 그림 1, 2와 같이 각 구조들을 직접 생성하는 것 외에도 그림 3에서와 같이 from_file 메서드를 사용해 구조 파일에서 구조를 읽어 Molecule 및 Structure 클래스로 변환할 수 있다. Scaling_matrix 인자를 [2,2,2]처럼 단순히 스칼라 형식으로 나타내는 것뿐만 아니라, [[1,1,0], [0,1,0], [2,3,1]]처럼 벡터 형식으로 나타내는 것도 가능하다.

2.3 구조 다루기

- Structure 클래스의 인스턴스, 즉 결정구조 데이터에서 슈퍼셀을 만들고 각 원자들을 치환, 추가, 제거하거나 miller plane들의 면간 거리, 나아가 X선 회절(X-ray diffraction, XRD) 패턴까지 계산해 낼 수 있다. 먼저, 그림 3과 같이 make_supercell 및 get_primitive_structure 메서드를 이용해 슈퍼셀 및 원시셀을 얻을 수 있다.
- Structure의 인스턴스에 인덱싱을 해 줌으로써 각 원자 및 이온들의 정보를 알 수 있다. 또한, 모든 Na^+ 이온들을 Li^+ 이온들로 간단하게 치환할 수 있으며, sites, lattice 등의 메서드들을 이용해 해당 결정구조에 대한 여러 가지 정보들을 알아낼 수 있다. 이 예시들을 그림 4에 나타내었다.
- 그림 5와 같이 pymatgen의 XRDCalculator 클래스를 사용해 주어진 구조의 XRD 패턴을 얻을 수 있다. 이때, X-ray의 wavelength는 숫자로 직접 입력할 수도 있고, 'CuKa'와 같이 pymatgen에서 지원하는 available radiation을 사용할 수도 있다. 그림 2에서 구조를 생성할 때 실험에서 얻은 격자상수 및 구조에 대한 정보

```python
from pymatgen.core import Molecule

coords = [[0.000000, 0.000000, 0.000000],
          [0.758602, 0.000000, 0.504284],
          [0.758602, 0.000000, -0.504284]]
h2o = Molecule(species = ['O', 'H', 'H'], coords = coords)
print(h2o)
```

```
Full Formula (H2 O1)
Reduced Formula: H2O
Charge = 0.0, Spin Mult = 1
Sites (3)
0 O     0.000000     0.000000     0.000000
1 H     0.758602     0.000000     0.504284
2 H     0.758602     0.000000    -0.504284
```

Fig. 1. Python code which generates molecular structure with Molecule class in pymatgen and its result.

```python
from pymatgen.core import Lattice, Structure

lat = Lattice.cubic(a = 5.64)

na_1, na_2, na_3, na_4 = [0, 0, 0], [0.5, 0.5, 0], [0.5, 0, 0.5], [0, 0.5, 0.5]
cl_1, cl_2, cl_3, cl_4 = [0.5, 0, 0], [1, 0.5, 0], [0.5, 0.5, 0.5], [0, 0, 0.5]
s = Structure(lattice = lat,
              species = ['Na+', 'Na+', 'Na+', 'Na+', 'Cl-', 'Cl-', 'Cl-', 'Cl-'],
              coords = [na_1, na_2, na_3, na_4, cl_1, cl_2, cl_3, cl_4])
print(s)
s.to(filename = 'NaCl.vasp', fmt='poscar')
```

```
Full Formula (Na4 Cl4)
Reduced Formula: NaCl
abc   :   5.640000   5.640000   5.640000
angles:  90.000000  90.000000  90.000000
pbc   :       True       True       True
Sites (8)
  #  SP     a    b    c
---  ----  ---  ---  ---
  0  Na+   0    0    0
  1  Na+   0.5  0.5  0
  2  Na+   0.5  0    0.5
  3  Na+   0    0.5  0.5
  4  Cl-   0.5  0    0
  5  Cl-   1    0.5  0
  6  Cl-   0.5  0.5  0.5
  7  Cl-   0    0    0.5
```

Fig. 2. Python code which generates periodic crystal structure with Structure class in pymatgen and its result.

```python
s = Structure.from_file('NaCl.vasp')
print("Original unit cell : {}".format(s.composition))

s.make_supercell(scaling_matrix=[2,2,2])
print("2x2x2 supercell : {}".format(s.composition))

prim = s.get_primitive_structure()
print("Primitive cell : {}".format(prim.composition))
```

```
Original unit cell : Na4 Cl4
2x2x2 supercell : Na32 Cl32
Primitive cell : Cl1 Na1
```

Fig. 3. Python code which generates supercell and primitive cell with methods in Structure class and its result.

```python
s = Structure.from_file('NaCl.vasp')

s['Na'] = 'Li'
first_species, sites, s_lat, sg_info = s[0], s.sites, s.lattice, s.get_space_group_info()
miller_index_135 = s.get_miller_index_from_site_indexes(site_ids = [1,3,5])
d_spacing = s_lat.d_hkl(miller_index = [0,1,0])

print("The first species : {}".format(first_species))
print("Sites of NaCl : ")
for site in sites :
    print(site)
print("Lattice : ")
print(s_lat)
print("Space group : {} (#{})".format(sg_info[0], sg_info[1]))
print("Miller index of plane generated by s[1], s[3], and s[5] : {}".format(miller_index_135))
print("d-spacing for (0, 1, 0) plane : {}(Angstrom)".format(d_spacing))
```

```
The first species : [0. 0. 0.] Li
Sites of NaCl :
[0. 0. 0.] Li
[2.82 2.82 0.  ] Li
[2.82 0.   2.82] Li
[0.   2.82 2.82] Li
[2.82 0.   0.  ] Cl
[5.64 2.82 0.  ] Cl
[2.82 2.82 2.82] Cl
[0.   0.   2.82] Cl
Lattice :
5.640000 0.000000 0.000000
0.000000 5.640000 0.000000
0.000000 0.000000 5.640000
Space group : Fm-3m (#225)
Miller index of plane generated by s[1], s[3], and s[5] : (0, 1, 0)
d-spacing for (0, 1, 0) plane : 5.64(Angstrom)
```

Fig. 4. Python code handling crystal structure with methods in Structure class and its result.

```python
from pymatgen.analysis.diffraction.xrd import XRDCalculator

s = Structure.from_file('NaCl.vasp')
xrd = XRDCalculator(wavelength = 'CuKa')
xrd.show_plot(s)
```

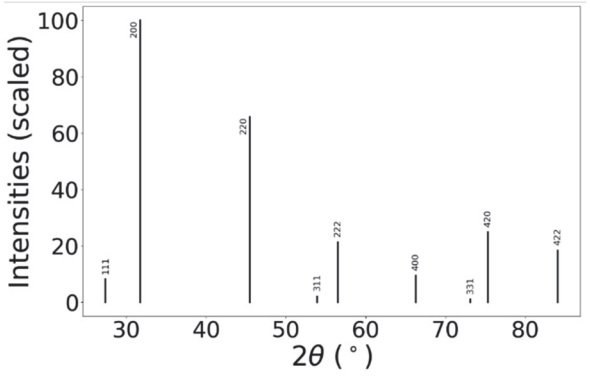

Fig. 5. Python code calculating XRD pattern with XRDCalculator class in pymatgen and its result.

[11]를 사용했으므로 pymatgen을 통해 얻은 XRD 패턴과 실제 실험 XRD 패턴은 피크의 위치와 세기가 일치하는 것을 확인할 수 있다. 하지만 pymatgen을 통해 얻은 XRD 패턴은 실제 실험 XRD 패턴이 가지고 있는 피크의 넓이 정보를 가지고 있지 않다. 또한, DFT 계산을 통해 얻은 구조의 XRD 패턴과 실험적인 XRD 패턴을 비교할 때는 격자상수가 실험 결과와 일치하게 계산되었는지 확인해야 하는데, 이는 일반적으로 LDA의 경우 격자상수를 작게 계산하고 GGA의 경우 격자상수를 크게 계산하기 때문이다 [12,13].

2.4 DFT relaxation VASP 입력 파일 생성

- 앞서 생성한 결정구조와 pymatgen.io.vasp.sets 모듈의 클래스들을 사용해 여러가지 DFT functional 계산의 입력 파일들을 자동으로 생성할 수 있다. 여기서 POTCAR 파일을 생성하기 위해서는 앞선 단계의 POTCAR 셋업이 먼저 되어 있어야 한다. Pymatgen.io.vasp.sets 모듈에는 MPRelaxSet과 같은 GGA+U relaxation 계산의 입력 파일들을 생성하는 클래스, MPHSERelaxSet과 같은 HSE06 계산의 입력 파일들을 생성하는 클래스, MITMDSet과 같은 제일원리 분자동역학(Ab Initio Molecular Dynamics, AIMD) 계산의 입력 파일들을 생성하는 클래스 등 여러 functional 및 계산들의 입력 파일을 생성하주는 클래스들이 있다. 이들의 사용법은 모두 동일한데, 결정구조와 여러 Set 클래스들을 이용해 계산에 대한 set을 먼저 정의한 후, write_input 메서드를 사용해 해당 계산에 대한 입력 파일들을 생성할 수 있다. 계산 set들을 정의할 때 user_incar_settings, user_potcar_functional, user_potcar_settings, user_kpoints_settings 등의 인자들을 조절해 각 입력 파일들의 파라미터들을 세부적으로 조정할 수 있다. 그림 6(a) PBE GGA+U 계산과 (b) HSE06 계산에 대한 VASP 입력 파일들을 생성하는 예시이다. 이외의 더 많은 DFT functional set들의 종류 및 사용법은 pymatgen 웹페이지 [14]에서 확인할 수 있다.
- MITRelaxSet은 Ceder 그룹의 high-throughput 계산 프로젝트 [15]에서 사용된 파라미터들을 바탕으로 입력 파일들을 생성해 준다. 적은 수의 valence electron들을 사용하도록 설정되므로, 빠르게 계산 결

(a)
```python
from pymatgen.io.vasp.sets import MITRelaxSet

s = Structure.from_file('LiCoO2.vasp')
incar_setting = {"ALGO" : "Fast",
                 "NSW" : "199"}
potcar_setting = {"Li" : "Li_sv", "Co" : "Co", "O" : "O"}
kpoints_setting = {"grid_density" : "500"}
relax = MITRelaxSet(s,  user_incar_settings = incar_setting,
                        user_potcar_functional = "PBE",
                        user_potcar_settings = potcar_setting,
                        user_kpoints_settings = kpoints_setting)
relax.write_input('.')
```

(b)
```python
from pymatgen.io.vasp.sets import MPHSERelaxSet

s = Structure.from_file('LiCoO2.vasp')
hse = MPHSERelaxSet(s, user_potcar_functional = 'PBE_54')
hse.write_input('.')
```

Fig. 6. Python code generating VASP input files for (a) GGA+U calculation and (b) HSE06 calculation.

과를 얻을 수 있다는 장점이 있다. MPRelaxSet은 Materials Project [16] 데이터베이스의 계산 데이터들과 같은 파라미터들을 바탕으로 입력 파일들을 생성해 준다. Materials Project 데이터베이스에 사용된 계산들과 설정을 공유하기 때문에 계산 결과를 비교하거나 Materials Project의 데이터와 합쳐 상태도를 구축하는 것이 가능하다는 장점이 있다.

2.5 AIMD 계산 VASP 입력 파일 생성

■ MITMDSet 혹은 MPMDSet은 AIMD 계산에 대한 VASP 입력 파일들을 생성해 준다. 이 MDSet들은 여러 가지 온도에 대한 AIMD 계산 입력 파일들을 한 번에 만들어 주지 않는다. 따라서, 본 AIMD 계산에 앞서 heatup 과정을 거치는 경우, 혹은 여러 온도에 대해서 AIMD 계산을 진행하는 경우 여러 온도에 대한 입력 파일들을 각각 생성해 주어야 한다. 그림 7은 100 K에서 500 K까지 heatup 계산 및 500 K에서 500 step 동안 AIMD 계산을 하는 데 필요한 VASP 입력 파일 생성 예시이다.

2.6 NEB 계산 VASP 입력 파일 생성

■ Nudged elastic band (NEB) 계산을 하기 위해서는 원자 확산 이전 및 이후의 구조와 확산 경로, 즉 이미지들이 필요하다. Pymatgen의 add-on 패키지인 pymatgen-analysis-diffusion 패키지 [17]는 확산 현상에 대한 계산 및 분석에 특화된 패키지인데, 다음과 같은 명령어로 설치할 수 있으며 관련 GitHub 웹페이지 [18]를 참고할 수 있다.

```
pip install pymatgen-analysis-diffusion
```

■ Pymatgen-analysis-diffusion 패키지의 IDPPSolver 클래스를 이용해 확산 이전 및 이후의 구조를 가지고 NEB 이미지들을 생성할 수 있다. 또한, 생성한 NEB 이미지들과 MITNEBSet으로부터 NEB 계산의 VASP 입력 파일들을 생성할 수 있다. MITNEBSet은 MITRelaxSet의 파라미터들을 기반으로 NEB 계산의 VASP 입력 파일들을 생성해 준다. 그림 8(a)는 확산 전, 후의 구조 로부터 NEB 이미지들을 만들고 이로부터 VASP 입력 파일들을 생성하는 예시이며, 그림 8(b)는 IDPPSolver 클래스로부터 생성된 NEB 이미지, 즉 초기 확산 경로이다. 그림 8의 예시에서는 2022.7.21 버전의 pymatgen-nalysis-diffusion 패키지를 사용하였다.

```python
from pymatgen.io.vasp.sets import MITMDSet

s = Structure.from_file('LiCoO2.vasp')
heatup = MITMDSet(s, start_temp=100, end_temp=500, nsteps=400,
                  user_incar_settings={"SMASS" : "-1"})

thermo = MITMDSet(s, start_temp=500, end_temp=500, nsteps=500,
                  user_incar_settings={"SMASS" : "0", "POTIM" : "1"})

heatup.write_input('/heatup')
thermo.write_input('/thermo')
```

Fig. 7. Python code generating VASP input files for AIMD calculation in heatup stage and at 500 K temperature.

(a)

```python
from pymatgen.analysis.diffusion.neb.pathfinder import IDPPSolver
from pymatgen.io.vasp.sets import MITNEBSet

init_st = Structure.from_file('POSCAR_initial')
final_st = Structure.from_file('POSCAR_final')

solver = IDPPSolver.from_endpoints(endpoints = [init_st, final_st], nimages = 7)
new_path = solver.run(maxiter=5000, tol=1e-5, gtol=1e-3, species=['Na'])

neb = MITNEBSet(new_path)
neb.write_input('.')
```

(b)

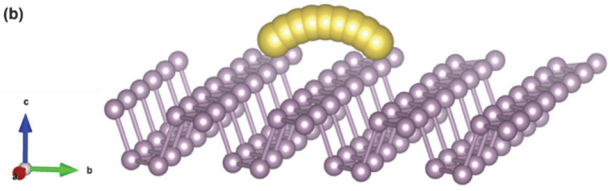

Fig. 8. (a) Python code generating NEB images from structures before and after diffusion, and VASP input files for NEB calculation and (b) Generated initial NEB diffusion pathway.

2.7 Gaussian 입력 파일 생성

■ Pymatgen은 본문에서 주로 다룬 VASP 외에도 Gaussian, ABINIT, LAMMPS, CP2K 등의 다른 프로그램들에 대한 입력 파일 생성 기능도 제공한다. 그림 9

```python
from pymatgen.io.gaussian import GaussianInput
from pymatgen.core import Molecule

coords = [[0, 0, 0], [0, 1.2, 0]]
o2 = Molecule(species = ['O', 'O'], coords = coords)

g_input = GaussianInput(mol = o2,
                        spin_multiplicity = 3,
                        functional = 'b3lyp',
                        basis_set = '6-31G',
                        link0_parameters = {"%nproc" : 8, "%mem" : '2000MW'},
                        dieze_tag = '#p')

g_input.write_file('INPUT.gjf', cart_coords=True)
```

```
%mem=2000MW
%nproc=8
#p b3lyp/6-31G

O2

0 3
O 0.000000 0.000000 0.000000
O 0.000000 1.200000 0.000000
```

Fig. 9. Python code generating Gaussian input files for B3LYP calculation of triplet O_2 molecule and its result.

는 GaussianInput 클래스를 이용해 삼중항 상태의 O_2 분자에 대한 B3LYP 계산의 입력 파일을 생성하는 예시이다. Link0_parameters 인자를 조절해 cpu의 수, 메모리 예측값 등을 자유롭게 조절할 수 있다. Pymatgen을 통해 다룰 수 있는 프로그램 리스트 및 사용법은 pymatgen 웹페이지 [19]에서 확인할 수 있다.

3. 결 론

- 제일원리계산은 현상의 예측, 가설 및 모델의 검증, 스크리닝 등 소재 개발 분야에서 널리 사용되고 있지만, 진입 장벽이 높고 해당 분야에 전문지식이 많지 않은 사람들이 쉽게 접근하기 어렵다는 문제점이 있다. 이에 본 tutorial status report에서는 분자 및 결정구조를 다루는 파이썬 패키지인 pymatgen을 소개하고, 이를 이용해 VASP 및 Gaussian 입력 파일들을 자동으로 생성하는 방법을 소개하였다. Pymatgen 패키지를 통해 DFT 계산 프로그램의 입력 파일들을 생성하면 해당 분야에 전문적인 지식이 없더라도 여러 가지 소재들에 대해 동일한 조건의 계산 입력 파일들을 손쉽게 만들 수 있고, 계산 설정들을 원하는 대로 조절할 수 있기 때문에 이러한 어려움들을 극복할 수 있다.
- 본 tutorial status report에서는 간단한 결정구조, 분자구조 생성 및 다루는 법과 VASP, Gaussian 입력 파일들을 생성하는 데 그쳤지만, pymatgen은 배위수 분석, Materials Project 데이터베이스와 연동 및 상태도 구축 등 다른 유용한 기능들을 지원할 뿐만 아니라, ABINIT, LAMMPS, CP2K 등과 같은 다른 여러 제일원리계산 프로그램들의 입력 파일 생성, 결과 파일 분석 등의 기능도 지원한다. 따라서 pymatgen 패키지는 초심자들도 쉽게 제일원리계산을 이용한 연구를 수행할 수 있게 해 주며, 이를 통해 전산재료과학 분야의 발전 및 소재 개발에 긍정적인 영향을 끼칠 것으로 기대한다.

ORCID

Dong-Hwa Seo https://orcid.org/0000-0002-7200-7186

감사의 글

This work was supported by the 2019 Research Fund(1.190150.01) of UNIST.

REFERENCES

[1] P. Hohenberg and W. Kohn, *Phys. Rev.*, **136**, B864 (1964). [DOI: https://doi.org/10.1103/PhysRev.136.B864]

[2] W. Kohn and L. J. Sham, *Phys. Rev.*, **140**, A1133 (1965). [DOI: https://doi.org/10.1103/PhysRev.140.A1133]

[3] J. Hafner, *J. Comput. Chem.*, **29**, 2044 (2008). [DOI: https://doi.rg/10.1002/jcc.21057]

[4] S. Kim, M. Aykol, V. I. Hegde, Z. Lu, S. Kirklin, J. R. Croy, M. M. Thackeray, and C. Wolverton, *Energy Environ. Sci.*, **10**, 2201 (2017). [DOI: https://doi.org/10.1039/C7EE01782K]

[5] D. -H. Seo, J. Lee, A. Urban, R. Malik, S. Y. Kang, and G. Ceder, *Nat. Chem.*, **8**, 692 (2016). [DOI: https://doi.org/10.1038/nchem. 524]

[6] S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson, and G. Ceder, *Comput. Mater. Sci.*, **68**, 314 (2013). [DOI: https://doi.org/10.016/j.commatsci.2012.10.028]

[7] WEBSITE: https://docs.conda.io/en/latest/miniconda.html, accessed date: September 2022.

[8] WEBSITE: https://pymatgen.org/installation.html, accessed date: September 2022.

[9] WEBSITE: https://pymatgen.org/index.html, accessed date: September 2022.

[10] WEBSITE: https://pymatgen.org/modules.html, accessed date: September 2022.

[11] M. Rabiei, A. Palevicius, A. Dashti, S. Nasiri, A. Monshi, A. Vilkauskas, and G. Janusas, *Mater.*, **13**, 4380 (2020). [DOI: https://doi.org/10.3390/ma13194380]

[12] S. Narasimhan and S. D. Gironcoli, *Phys. Rev. B*, **65**, 064302 (2002). [DOI: https://doi.org/10.1103/PhysRevB.65.064302]

[13] G. I. Csonka, J. P. Perdew, A. Ruzsinszky, P.H.T. Philipsen, S. Lebègue, J. Paier, O. A. Vydrov, and J. G. Ángyán, *Phys. Rev. B*, **79**, 155107 (2009). [DOI: https://doi.org/10.1103/PhysRevB.9.155107]

[14] WEBSITE: https://pymatgen.org/pymatgen.io.vasp.sets.html, accessed date: September 2022.

[15] A. Jain, G. Hautier, C. J. Moore, S. P. Ong, C. C. Fischer, T. Mueller, K. A. Persson, and G. Ceder, *Comput. Mater. Sci.*, **50**, 2295 (2011). [DOI: https://doi.org/10.1016/j.commatsci.2011. 2.023]

[16] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson, *APL Mater.* **1**, 011002 (2013). [DOI: https://doi.org/0.1063/1.4812323]

[17] Z. Deng, Z. Zhu, I. -H. Chu, and S. P. Ong, *Chem. Mater.*, **29**, 281 (2017). [DOI: https://doi.org/10.1021/acs.chemmater.6b02648]

[18] WEBSITE: https://github.com/materialsvirtuallab/pymatgen-analysis-diffusion, accessed date: September 2022.

[19] WEBSITE: https://pymatgen.org/pymatgen.io.html, accessed date: September 2022.