This supplementary data file is the part of the following article.

Journal of Modeling and Simulation of Materials
ISSN: 2582-2365
Volume 7, Issue 1, pp. 1-17, 2025
DOI: https://doi.org/10.21467/jmsm.7.1.1-17

SUPPLEMENTARY FILE

Supplemental Material

Additional Data

Appendix A. Z-matrices of the isomers

(a) (E, E)-1,2,3,4-TCBD

```
c
 c   1 cc2
cl    1 clc3       2 clcc3
 c    2 cc4        1 ccc4        3 dih4
cl    2 clc5       1 clcc5       3 dih5
 c    4 cc6        2 ccc6        1 dih6
cl    4 clc7       6 clcc7       2 dih7
 h    6 hc8        4 hcc8        7 dih8
cl    6 clc9       4 clcc9       7 dih9
 h    1 hc10       3 hccl10      2 dih10

cc2=        1.335000
clc3=       1.740000
clcc3=       120.000
cc4=        1.500000
ccc4=        120.000
dih4=          0.000
clc5=       1.740000
clcc5=       120.000
dih5=        180.000
cc6=        1.335000
ccc6=        120.000
dih6=        180.000
clc7=       1.740000
clcc7=       120.000
dih7=        180.000
hc8=        1.089000
hcc8=        120.000
dih8=          0.000
clc9=       1.740000
clcc9=       120.000
dih9=        180.000
hc10=       1.070000
hccl10=      120.000
dih10=       180.000
```

(b) (Z, Z)-1,2,3,4-TCBD

```
c
 c   1 cc2
 h    1 hc3        2 hcc3
 c    2 cc4        1 ccc4         3 dih4
cl    2 clc5       1 clcc5        3 dih5
cl    1 clc6       3 clch6        2 dih6
 c    4 cc7        2 ccc7         1 dih7
cl    4 clc8       7 clcc8        2 dih8
cl    7 clc9       4 clcc9        8 dih9
 h    7 hc10       4 hcc10        8 dih10

cc2=         1.335000
hc3=         1.089000
hcc3=        120.000
cc4=         1.500000
ccc4=        120.000
dih4=          0.000
clc5=        1.740000
clcc5=       120.000
dih5=        180.000
clc6=        1.740000
clch6=       120.000
dih6=        180.000
cc7=         1.335000
ccc7=        120.000
dih7=        180.000
clc8=        1.740000
clcc8=       120.000
dih8=        180.000
clc9=        1.740000
clcc9=       120.000
dih9=          0.000
hc10=        1.089000
hcc10=       120.000
dih10=       180.000
```

(c) (E, Z)-1,2,3,4-TCBD

```
c
 c   1 cc2
 h    1 hc3        2 hcc3
 c    2 cc4        1 ccc4         3 dih4
cl    2 clc5       1 clcc5        3 dih5
cl    1 clc6       3 clch6        2 dih6
 c    4 cc7        2 ccc7         1 dih7
cl    4 clc8       7 clcc8        2 dih8
cl    7 clc9       4 clcc9        8 dih9
 h    7 hc10       9 hccl10       4 dih10

cc2=        1.335000
hc3=        1.089000
hcc3=       120.000
cc4=        1.500000
ccc4=       120.000
dih4=         0.000
clc5=       1.740000
clcc5=      120.000
dih5=       180.000
clc6=       1.740000
clch6=      120.000
dih6=       180.000
cc7=        1.335000
ccc7=       120.000
dih7=       180.000
clc8=       1.740000
clcc8=      120.000
dih8=       180.000
clc9=       1.740000
clcc9=      120.000
dih9=       180.000
hc10=       1.070000
hccl10=     120.000
dih10=      180.000
```

(d) (Z)-1,1,2,4-TCBD

```
c
 c   1 cc2
 h    1 hc3        2 hcc3
 c    2 cc4        1 ccc4         3 dih4
cl    1 clc5       3 clch5        2 dih5
 h    2 hc6        1 hcc6         3 dih6
 c    4 cc7        2 ccc7         1 dih7
cl    4 clc8       7 clcc8        2 dih8
cl    7 clc9       4 clcc9        8 dih9
cl    7 clc10      4 clcc10       8 dih10

cc2=         1.335000
hc3=         1.089000
hcc3=        120.000
cc4=         1.500000
ccc4=        120.000
dih4=          0.000
clc5=        1.740000
clch5=       120.000
dih5=        180.000
hc6=         1.070000
hcc6=        120.000
dih6=        120.000
cc7=         1.335000
ccc7=        120.000
dih7=        180.000
clc8=        1.740000
clcc8=       120.000
dih8=        180.000
clc9=        1.740000
clcc9=       120.000
dih9=          0.000
clc10=       1.740000
clcc10=      120.000
dih10=       180.000
```

(e) (Z, Z)-1,2,3,4-TCBD

```
c
 c   1 cc2
cl    1 clc3       2 clcc3
 c    2 cc4        1 ccc4        3 dih4
 h    2 hc5        1 hcc5        3 dih5
 c    4 cc6        2 ccc6        1 dih6
cl    4 clc7       6 clcc7       2 dih7
cl    6 clc8       4 clcc8       7 dih8
cl    6 clc9       4 clcc9       7 dih9
 h    1 hc10       3 hccl10      2 dih10

cc2=         1.335000
clc3=        1.740000
clcc3=     120.000
cc4=         1.500000
ccc4=      120.000
dih4=        0.000
hc5=         1.070000
hcc5=      120.000
dih5=      120.000
cc6=         1.335000
ccc6=      120.000
dih6=      180.000
clc7=        1.740000
clcc7=     120.000
dih7=      180.000
clc8=        1.740000
clcc8=     120.000
dih8=        0.000
clc9=        1.740000
clcc9=     120.000
dih9=      180.000
hc10=        1.070000
hccl10=    120.000
dih10=     180.000
```

(f) 1,1,4,4-TCBD

```
c
 c   1 cc2
cl    1 clc3       2 clcc3
 c    2 cc4        1 ccc4        3 dih4
 h    2 hc5        1 hcc5        3 dih5
 c    4 cc6        2 ccc6        1 dih6
cl    1 clc7       3 clccl7      2 dih7
 h    4 hc8        2 hcc8        1 dih8
cl    6 clc9       4 clcc9       2 dih9
cl    6 clc10      4 clcc10      2 dih10

cc2=         1.335000
clc3=        1.740000
clcc3=       120.000
cc4=         1.500000
ccc4=        120.000
dih4=          0.000
hc5=         1.070000
hcc5=        120.000
dih5=        120.000
cc6=         1.335000
ccc6=        120.000
dih6=        180.000
clc7=        1.740000
clccl7=      120.000
dih7=        180.000
hc8=         1.070000
hcc8=        120.000
dih8=        300.000
clc9=        1.740000
clcc9=       109.471
dih9=        180.000
clc10=       1.740000
clcc10=      109.471
dih10=       300.000
```

(g) (E)-1,1,3,4-TCBD

```
c
 c   1 cc2
cl    1 clc3       2 clcc3
 c    2 cc4        1 ccc4        3 dih4
 h    2 hc5        1 hcc5        3 dih5
 c    4 cc6        2 ccc6        1 dih6
cl    1 clc7       3 clccl7      2 dih7
cl    4 clc8       2 clcc8       1 dih8
cl    6 clc9       4 clcc9       2 dih9
 h    6 hc10       4 hcc10       9 dih10

cc2=        1.335000
clc3=       1.740000
clcc3=      120.000
cc4=        1.500000
ccc4=       120.000
dih4=         0.000
hc5=        1.070000
hcc5=       120.000
dih5=       120.000
cc6=        1.335000
ccc6=       120.000
dih6=       180.000
clc7=       1.740000
clccl7=     120.000
dih7=       180.000
clc8=       1.740000
clcc8=      120.000
dih8=       300.000
clc9=       1.740000
clcc9=      109.471
dih9=       300.000
hc10=       1.070000
hcc10=      109.471
dih10=      120.000
```

(h) (Z)-1,1,3,4-TCBD

```
c
 c   1 cc2
cl    1 clc3       2 clcc3
 c    2 cc4        1 ccc4        3 dih4
 h    2 hc5        1 hcc5        3 dih5
 c    4 cc6        2 ccc6        1 dih6
cl    1 clc7       3 clccl7      2 dih7
cl    4 clc8       2 clcc8       1 dih8
cl    6 clc9       4 clcc9       2 dih9
 h    6 hc10       4 hcc10       2 dih10

cc2=         1.335000
clc3=        1.740000
clcc3=       120.000
cc4=         1.500000
ccc4=        120.000
dih4=          0.000
hc5=         1.070000
hcc5=        120.000
dih5=        120.000
cc6=         1.335000
ccc6=        120.000
dih6=        180.000
clc7=        1.740000
clccl7=      120.000
dih7=        180.000
clc8=        1.740000
clcc8=       120.000
dih8=        300.000
clc9=        1.740000
clcc9=       109.471
dih9=        180.000
hc10=        1.070000
hcc10=       109.471
dih10=       300.000
```

(i) 1,1,2,3-TCBD

```
c
 c   1 cc2
cl    1 clc3       2 clcc3
 c    2 cc4        1 ccc4        3 dih4
cl    2 clc5       1 clcc5       3 dih5
 c    4 cc6        2 ccc6        1 dih6
cl    1 clc7       3 clccl7      2 dih7
cl    4 clc8       2 clcc8       1 dih8
 h    6 hc9        4 hcc9        2 dih9
 h    6 hc10       4 hcc10       2 dih10

cc2=         1.335000
clc3=        1.740000
clcc3=       120.000
cc4=         1.500000
ccc4=        120.000
dih4=          0.000
clc5=        1.740000
clcc5=       120.000
dih5=        120.000
cc6=         1.335000
ccc6=        120.000
dih6=        180.000
clc7=        1.740000
clccl7=      120.000
dih7=        180.000
clc8=        1.740000
clcc8=       120.000
dih8=        300.000
hc9=         1.070000
hcc9=        109.471
dih9=        300.000
hc10=        1.070000
hcc10=       109.471
dih10=        60.000
```

Appendix B. IR data from the simulation

1. TCBDa

freq, cm^{-1}	IR_Intensity, Infrared transition intensities in km/mol	Relative intensity
166	0	w
184	0	w
193	0	w
212	1	w
246	0	w
290	4	w
322	5	w
367	2	w
396	0	w
634	0	w
688	0	w
801	181	s
832	47	m
836	0	w
842	0	w
860	104	m
1104	0	w
1358	7	w
1404	0	w
1586	51	m
1645	0	w
3252	0	w
3252	21	m

Kaniz Fatema, *J. Mod. Sim. Mater.;* Vol. 7 Issue 1, pp: 1-17, 2025

2. TCBDb

freq, cm-1	IR_Intensity, Infrared transition intensities in km/mol	Relative intensity
34	0	w
119	2	w
135	0	w
192	0	w
227	0	w
249	0	w
288	0	w
332	0	w
435	3	w
566	21	m
580	0	w
639	0	w
727	106	s
814	55	m
830	0	w
891	143	s
911	0	w
1176	0	w
1302	7	w
1312	0	w
1627	81	m
1657	0	w
3275	25	m
3275	0	w

3. TCBDc

freq, cm-1	IR_Intensity, Infrared transition intensities in km/mol	Relative intensity
140	0	w
159	0	w
207	0	w
210	1	w
231	0	w
277	0	w
377	0	w
401	3	w
455	11	w
604	1	w
654	17	w
810	133	s
818	9	w
829	72	m
834	43	m
904	22	m
1131	20	m
1312	0	w
1378	6	w
1607	48	m
1650	12	w
3252	15	w
3285	12	w

4. TCBDd

freq, cm-1	IR_Intensity, Infrared transition intensities in km/mol	Relative intensity
66	0	w
103	1	w
125	0	w
169	0	w
211	0	w
251	0	w
294	0	w
353	1	w
385	2	w
393	0	w
567	0	w
618	54	m
766	76	m
828	4	w
876	104	s
929	62	m
954	37	m
1198	11	w
1233	8	w
1307	4	w
1607	67	m
1671	34	m
3218	3	w
3239	9	w

5. TCBDe

freq, cm-1	IR_Intensity, Infrared transition intensities in km/mol	Relative intensity
13	1	w
90	1	w
134	1	w
197	0	w
228	0	w
247	2	w
307	0	w
361	2	w
430	1	w
476	12	m
585	4	w
657	26	m
749	27	m
777	37	m
806	103	s
924	111	s
945	0	w
1157	18	m
1287	2	w
1369	30	m
1587	58	m-s
1680	27	m
3190	9	w
3218	4	w

Kaniz Fatema, *J. Mod. Sim. Mater.;* Vol. 7 Issue 1, pp: 1-17, 2025

6. TCBDf

freq, cm-1	IR_Intensity, Infrared transition intensities in km/mol	Relative intensity
46	1	w
89	2	w
92	1	w
183	0	w
237	0	w
307	1	w
318	0	w
377	0	w
471	5	w
475	13	w-m
487	0	w
608	80	m
688	0	w
886	0	w
887	287	s
889	26	w-m
928	0	w
1185	0	w
1249	55	m
1315	0	w
1629	165	s
1677	0	w
3206	0	w
3214	14	w-m

7. TCBDg

freq, cm-1	IR_Intensity, Infrared transition intensities in km/mol	Relative intensity
11	0	w
114	1	w
155	0	w
185	0	w
211	1	w
228	0	w
310	0	w
366	0	w
466	0	w
479	15	w-m
553	7	w
648	21	w-m
770	26	m
808	33	m
820	84	m
879	9	w
897	176	s
1146	75	m
1338	3	w
1360	6	w
1603	42	m
1677	57	m
3206	6	w
3265	17	w-m

Kaniz Fatema, *J. Mod. Sim. Mater.;* Vol. 7 Issue 1, pp: 1-17, 2025

8. TCBDh

freq, cm-1	IR_Intensity, Infrared transition intensities in km/mol	Relative intensity
25	0	w
102	1	w
119	1	w
201	0	w
231	0	w
251	0	w
281	0	w
362	0	w
462	1	w
547	6	w
584	40	m
646	34	m
677	39	m
789	12	w
875	28	m
879	75	s
908	82	s
1183	59	s
1307	33	m
1375	6	w
1611	95	s
1677	25	m
3189	1	w
3238	14	w

9. TCBDi

freq, cm-1	IR_Intensity, Infrared transition intensities in km/mol	Relative intensity
39	1	w
77	0	w
149	1	w
168	0	w
205	1	w
267	0	w
303	0	w
389	0	w
399	1	w
487	8	w
551	3	w
607	14	w-m
681	53	m
715	27	m
820	98	s
913	93	s
938	49	m
1035	3	w
1192	86	s
1441	6	w
1638	46	m
1706	17	w-m
3188	0	w
3279	1	w

Appendix C. Python Codes

1. Python file for geometry optimization and vibrational frequency calculation with B3LYP/6-31G level DFT method*[1]

```
"""
Lab3starter_OptimizationAndFrequencies.py

This code uses Psi4 to perform a geometry optimization for TCBD isomers, and
at the DFT (b3lyp/6-31G*) level, and then compute the vibrational
frequencies.

This is a starter code for Lab3.

Origin CHE 525, S21 Problem Development
Author: Tom Allison, Ben Levine
"""
#%%
# Import modules ======================================================
import psi4
import numpy as np
import CHE525_Vib as vib
import pandas as pd

#%%
# Set up Psi4 =========================================================

psi4.core.clean()
psi4.core.clean_options()
psi4.set_memory('4000 MB')  # Can make this much larger on Seawulf, each
compute node has more than 100 GB RAM
psi4.set_num_threads(4)     # Can make this much larger on Seawulf, each
compute node can support 28 threads.
                            # But it doesn't help much for small molecules...
psi4.core.set_output_file('TCBD.dat', False) #this command sets psi4's output
to a file. Comment this line out if you want to see the output on the
terminal.

#%%
# Define QC method to use and angles to scan ==========================

method = 'b3lyp/6-31G*'  # it is 'b3lyp-d3bj/cc-pvdz' for isomers d, e, and f

#%%
# Set up Z-matrix string and perform initial optimization =============
TCBD_string= """
    c
 c   1 cc2
cl    1 clc3        2 clcc3
 c    2 cc4         1 ccc4          3 dih4
cl    2 clc5        1 clcc5         3 dih5
 c    4 cc6         2 ccc6          1 dih6
cl    4 clc7        6 clcc7         2 dih7
 h    6 hc8         4 hcc8          7 dih8
cl    6 clc9        4 clcc9         7 dih9
 h    1 hc10        3 hccl10        2 dih10
```

```
cc2=         1.335000
clc3=        1.740000
clcc3=       120.000
cc4=         1.500000
ccc4=        120.000
dih4=          0.000
clc5=        1.740000
clcc5=       120.000
dih5=        180.000
cc6=         1.335000
ccc6=        120.000
dih6=        180.000
clc7=        1.740000
clcc7=       120.000
dih7=        180.000
hc8=         1.089000
hcc8=        120.000
dih8=          0.000
clc9=        1.740000
clcc9=       120.000
dih9=        180.000
hc10=        1.070000
hccl10=      120.000
dih10=       180.000"""

# Define molecule
TCBD = psi4.geometry(TCBD_string)

# Run the geometry optimization
E0, wfn0 = psi4.optimize(method, molecule = TCBD, return_wfn = True) #
perform initial optimization

# This option will tell Psi4 that, upon calculating the frequencies, it
# should output the a molden file that will include information so you can
# visualize the normal modes.
psi4.set_options({"normal_modes_write": True})

# This options provides Psi4 with the name for the file.  (Psi4 will append
# some additional characters to the file name.)
psi4.set_options({"writer_file_label": "tcbd-dft"})

# Compute the virbational frequencies. Note that Psi4 will remember the
# last calculation that you did, and by default will start using that
# (optimized) geometry.
psi4.frequencies(method)

# Compute IR intensities using using CHE525_vib package
# It may be helpful to adjust the step size eps to make sure the results are
# stable with respect to eps, especially if you are getting strange numbers!
vib_results = vib.dipder(method, TCBD, eps = 0.01)

# Display results at terminal
print(vib_results['om'])
print(vib_results['IR_Intensity'])

# Save data fo file with Pandas. Very easy when you already have a Python
```

```
dictionary!
df = pd.DataFrame(data = vib_results) # inialize DataFrame object using
vib_results dictionary
df.to_csv('vib_results.csv') # Save .csv file with the data.
```

2. Python file for geomtry optimization and vibrational frequency calculation with B3LYP-D3BJ/cc-pVDZ level DFT method[1]

```
"""
Lab3starter_OptimizationAndFrequencies.py

This code uses Psi4 to perform a geometry optimization for TCBD isomers, and
at the DFT (b3lyp/6-31G*) level, and then compute the vibrational
frequencies.

This is a starter code for Lab3.

Origin CHE 525, S21 Problem Development
Author: Tom Allison, Ben Levine
"""
#%%
# Import modules ======================================================
import psi4
import numpy as np
import CHE525_Vib as vib
import pandas as pd

#%%
# Set up Psi4 =========================================================

psi4.core.clean()
psi4.core.clean_options()
psi4.set_memory('4000 MB')  # Can make this much larger on Seawulf, each
compute node has more than 100 GB RAM
psi4.set_num_threads(4)    # Can make this much larger on Seawulf, each
compute node can support 28 threads.
                           # But it doesn't help much for small molecules...
psi4.core.set_output_file('TCBD.dat', False) #this command sets psi4's output
```

```
to a file. Comment this line out if you want to see the output on the
terminal.



#%%
# Define QC method to use and angles to scan ============================

method = 'b3lyp-d3bj/cc-pvdz'

#%%
# Set up Z-matrix string and perform initial optimization ==============
TCBD_string= """
    c
 c   1 cc2
cl    1 clc3        2 clcc3
 c    2 cc4         1 ccc4          3 dih4
 h    2 hc5         1 hcc5          3 dih5
 c    4 cc6         2 ccc6          1 dih6
cl    4 clc7        6 clcc7         2 dih7
cl    6 clc8        4 clcc8         7 dih8
cl    6 clc9        4 clcc9         7 dih9
 h    1 hc10        3 hccl10        2 dih10

cc2=        1.335000
clc3=       1.740000
clcc3=      120.000
cc4=        1.500000
ccc4=       120.000
dih4=         0.000
hc5=        1.070000
hcc5=       120.000
dih5=       120.000
cc6=        1.335000
ccc6=       120.000
dih6=       180.000
clc7=       1.740000
clcc7=      120.000
dih7=       180.000
```

```
clc8=       1.740000
clcc8=      120.000
dih8=         0.000
clc9=       1.740000
clcc9=      120.000
dih9=       180.000
hc10=       1.070000
hccl10=     120.000
dih10=      180.000"""

# Define molecule
TCBD = psi4.geometry(TCBD_string)

# Run the geometry optimization
E0, wfn0 = psi4.optimize(method, molecule = TCBD, return_wfn = True) #
perform initial optimization

# This option will tell Psi4 that, upon calculating the frequencies, it
# should output the a molden file that will include information so you can
# visualize the normal modes.
psi4.set_options({"normal_modes_write": True})

# This options provides Psi4 with the name for the file.  (Psi4 will append
# some additional characters to the file name.)
psi4.set_options({"writer_file_label": "tcbd-dft"})

# Compute the virbational frequencies. Note that Psi4 will remember the
# last calculation that you did, and by default will start using that
# (optimized) geometry.
psi4.frequencies(method)

# Compute IR intensities using using CHE525_vib package
# It may be helpful to adjust the step size eps to make sure the results are
# stable with respect to eps, especially if you are getting strange numbers!
vib_results = vib.dipder(method, TCBD, eps = 0.01)

# Display results at terminal
print(vib_results['om'])
```

```
print(vib_results['IR_Intensity'])


# Save data fo file with Pandas. Very easy when you already have a Python
dictionary!
df = pd.DataFrame(data = vib_results) # inialize DataFrame object using
vib_results dictionary
df.to_csv('vib_results.csv') # Save .csv file with the data.
```

3. Python file for getting data for vibrational frequency and IR intensity calculation[2]

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
CHE525_Vib.py

This package contains functions for easily extracting a molecule's dipole
moment
and calculating the derivative of the dipole moment, and resulting "IR
intensities"
for a molecule.

Origin: Stony Brook University CHE525 Problem development, S21
Author: Tom Allison
"""

import psi4
import numpy as np
import qcelemental as qcel
#from qcelemental import Datum

def get_dipole(wfn_in):
    """This function returns a 3-component numpy array for dipole moment
extracted
       from  a psi4 wave function object. It doesn't calculate anything. It
just
       pulls the data out of the wfn in a more convenient format."""
    psi4.oeprop(wfn_in,'DIPOLE')
    D = np.zeros( (3,) ) # initialize empty numpy array to store dipole
vector components
    D[0] = wfn_in.variable(' DIPOLE X')
    D[1] = wfn_in.variable(' DIPOLE Y')
    D[2] = wfn_in.variable(' DIPOLE Z')
    D = (0.3934303)*D # convert to atomic units e a_0
    return D

def dipder(method, mol, eps = 0.01):
    """Dipole Derivatives

       This function performs vibrational analysis for a psi4 molecule
object.
       It fills a gap in standard Psi4, providing dipole derivatives and IR
       intensities. The derivatives are calculated using finite difference
```

```
via
       displacing the molecule geometry a small amount in the direction of
       the normal mode coordinates.

       Required input arguments:
       method = QC method to use for the calculation, e.g. b3lyp/6-31G*
       mol = psi4 molecule object

       Optional argument eps lets you set the (fractional) step size to take
for
       the finite difference. Default value of 0.01 takes a step size of 1%
of the normal mode.
       Making this too large reduces the accuracy of the finite difference as
an approximation
       of the derivative. Making this too small can run into problems with
numerical accuracy
       in the dipole calculation. 1% is a compromise that works reasonably
well
       for many cases.

       This code returns a dictionary with the following keys:

       'dipder': Dipole derivatives for small change in normal mode
coordinate. In units of dipole [au] * length [au]. Similar to what psi4
usually calculates for RHF
       'om': Vibrational frequencies in cm^{1}. Excludes rotations and
translations.
       'IR_Intensity': Infrared transition intensities in km/mol

       """
    uconv_kmmol = (qcel.constants.get("Avogadro constant") * np.pi * 1.e-3 *
qcel.constants.get("electron mass in u") *\
                   qcel.constants.get("fine-structure constant")**2 *
qcel.constants.get("atomic unit of length") / 3)    # code taken from
https://github.com/psi4/psi4/blob/master/psi4/driver/qcdb/vib.py

    XYZ0 = mol.geometry() #psi4 matrix object, initial geometry
    XYZ0_str = mol.save_string_xyz()
    mol = psi4.geometry(XYZ0_str) # redefine molecule in terms of XYZ matrix
since having problems when molecule defined in terms of Z-matrix ?!
    XYZ1 = mol.geometry() # initialize displaced geometry
    XYZ0_np = XYZ0.to_array() #extract XYZ matrix  in numpy format

    #initialize dictionary with empty lists
    results = {}
    results['dipder'] = [] #dipole derivatives in a.u.
    results['om'] = []     #frequencies in cm^{-1}
    results['IR_Intensity'] = [] # IR intensities in km/mol

    E, wfn0 = psi4.frequency(method, molecule = mol, return_wfn = True)
    mol.fix_orientation(True) # turn reorientation off! Without this you will
get nonsense if psi4 reorients the molecule between dipole calculations!
    D0 = get_dipole(wfn0)
    print('Initial dipole:')
    print(D0)
    TRV = wfn0.frequency_analysis['TRV'].data
    w = wfn0.frequency_analysis['w'].data
```

```
    om = wfn0.frequency_analysis['omega'].data

    for j in np.arange(len(om)):
        if TRV[j] == 'V':
            wn = w[:,j] #extract normal mode coordinates for normal mode n.
            #print(wn)
            results['om'].append(om[j])
            print('Working on mode ' + str(j) + ' with frequency ' +
str(om[j]))
            disp_mat = np.zeros(np.shape(XYZ0))
            for i in np.arange(np.shape(XYZ0)[0]):
                disp_mat[i,:] = eps*wn[3*i:(3*i+3)]
            XYZ1_np = XYZ0_np + disp_mat
            XYZ1 = XYZ1.from_array(XYZ1_np)
            mol.set_geometry(XYZ1) #update geometry
            E, wfn1 = psi4.energy(method, molecule = mol, return_wfn = True)
            D1 = get_dipole(wfn1)
            print('D1 = ')
            print(D1)
            dDdw = (D1-D0)/(eps) # calculate dmu/deps. Note that this is not
dD/dX or dD/dW, but instead (nearly) equivalent to the dot product of psi4's
normal dipder with w.
            results['dipder'].append(dDdw)
            results['IR_Intensity'].append(uconv_kmmol*np.dot(dDdw,dDdw))
#Under construction! Need to get prefactor right!

    return results
```

References

(1) *sbu-che525-ss24/Lab3/Lab3starter.py at main · blevine37/sbu-che525-ss24*. https://github.com/blevine37/sbu-che525-ss24/blob/main/Lab3/Lab3starter.py (accessed 2024-05-03).

(2) *sbu-che525-ss24/Lab3/CHE525_Vib.py at main · blevine37/sbu-che525-ss24*. https://github.com/blevine37/sbu-che525-ss24/blob/main/Lab3/CHE525_Vib.py (accessed 2024-05-03).