浙江大学量子化学实验

量子化学计算中的紫外光谱

实
验
报
告

参加学生：叶青杨（3210100360）

指导老师：刘迎春

浙江大学化学实验教学中心

2023 年 10 月 11 日

量子化学计算中的紫外光谱

叶青杨 (3210100360), 求化 2101，指导教师：刘迎春

1 实验目的

1.1 对方法和基组优化参数

1.2 计算指定物质的紫外光谱并分析

2 实验步骤

通过大量的尝试，我们得出，完成紫外光谱计算的方法步骤为：

1. 使用较高精度的方法完成 opt
2. 使用 opt 后的结构，不使用 opt 直接进行 TD-SCF/CIS 计算。

3 实验结果

完成了如 table 1 中的计算。

Molecule	Method	Basis Set/Note
山梨酸	opt freq B3LYP	6-311G
	opt freq TD CAM-B3LYP	aug-cc-pvtz (failed)
	opt freq TD B3LYP	6-311G
	opt freq CIS	6-311G
	td=(nstates=12) CAM-B3LYP	aug-cc-pvtz (failed)
	td CAM-B3LYP	aug-cc-pvtz
阿司匹林	opt freq B3LYP	6-311G
	opt freq TD B3LYP	6-311G
	opt freq CIS	aug-cc-pvtz (failed)
	opt freq CIS	6-311G
	opt freq TD CAM-B3LYP	aug-cc-pvdz (failed)
	opt freq TD CAM-B3LYP	cc-pvtz (failed)
	opt freq CIS	6-31G
	td CAM-B3LYP	aug-cc-pvtz (failed)
	td CAM-B3LYP	aug-cc-pvtz
	td=(nstates=15) CAM-B3LYP	aug-cc-pvtz
	CIS	aug-cc-pvtz (failed)
	td HF	aug-cc-pvtz (failed)
苯甲酸	opt freq B3LYP	6-311G
	opt freq B3LYP	6-311G
	opt freq TD B3LYP	6-311G
	opt freq TD B3LYP	6-311G
	opt freq CIS	6-311G
	td CAM-B3LYP	aug-cc-pvtz
	td=(nstates=20) CAM-B3LYP	aug-cc-pvtz (failed)
甲烷	opt freq B3LYP	6-311G
	opt freq TD CAM-B3LYP	aug-cc-pvdz (error)
	opt freq TD B3LYP	3-21G (error)
	td B3LYP	6-311G
	td CAM-B3LYP	aug-cc-pvtz
	opt freq CIS	6-311G (error)
	opt freq TD HF	6-311G (error)
	opt freq TD PM6	
	opt freq TD B3LYP	6-311G (error)
	opt freq CIS	6-311G (error)
	opt freq TD CAM-B3LYP	aug-cc-pvdz (failed)
	opt freq TD CAM-B3LYP	6-311G (error)
	opt freq TD B3LYP	6-311G (failed)
	opt freq TD CAM-B3LYP	aug-cc-pvdz (error)
	opt freq TD CAM-B3LYP	aug-cc-pvtz (error)
	td=(nstates=15) CAM-B3LYP	aug-cc-pvtz
	td CAM-B3LYP	aug-cc-pvtz

Table 1: 各种分子的计算方法和基组总结

选取了部分最优的高精度计算的成功的计算结果

Figure 1: 山梨酸 td CAM-B3LYP aug-cc-pvtz

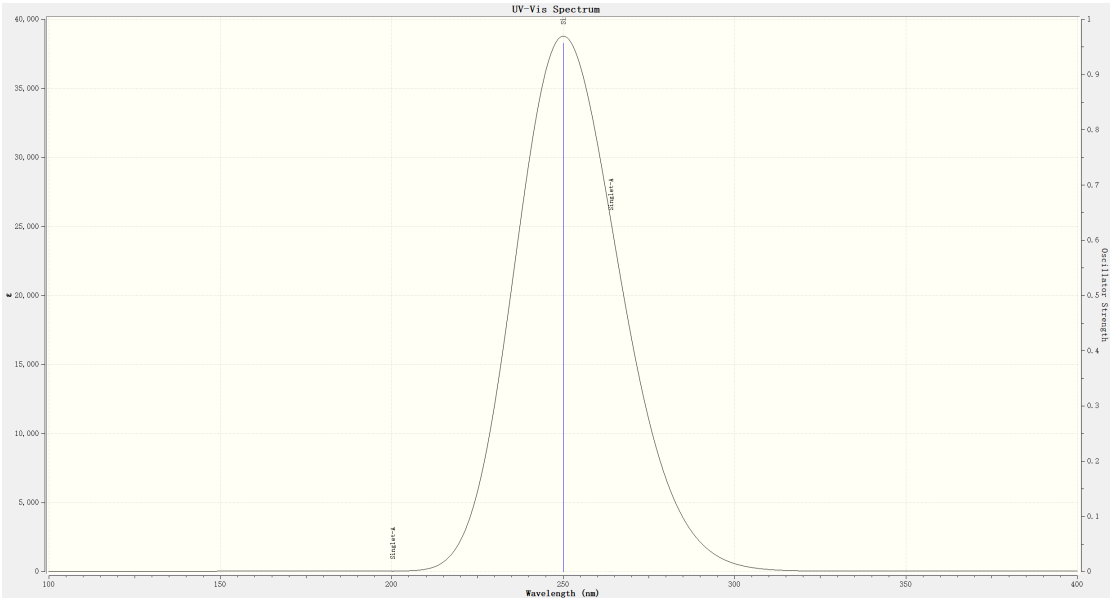


Figure 2: 阿司匹林 td=(nstates=15) CAM-B3LYP aug-cc-pvtz

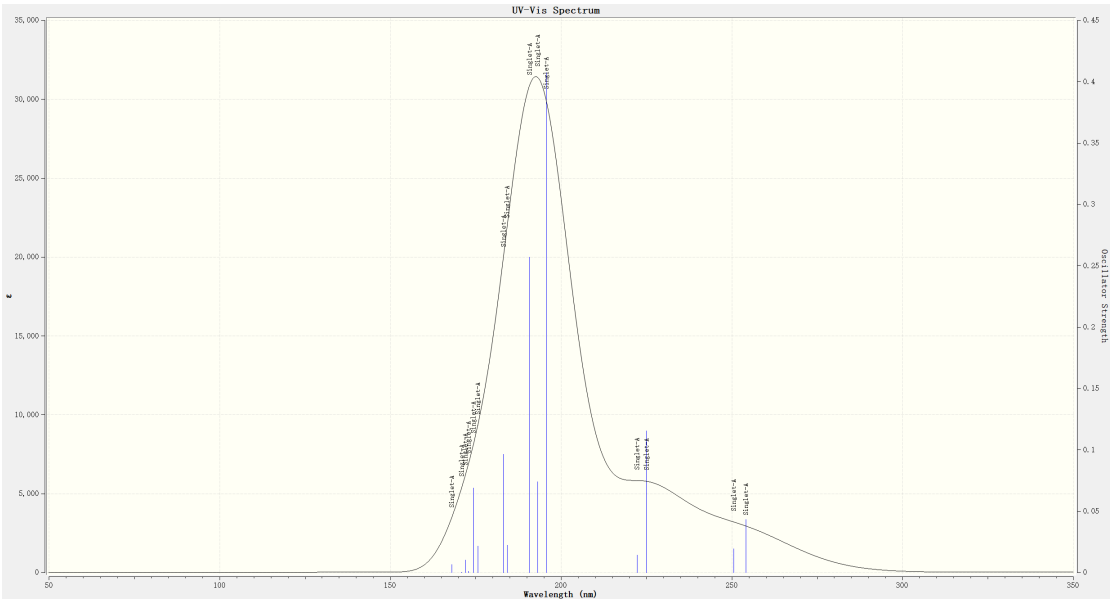


Figure 3: 苯甲酸 td CAM-B3LYP aug-cc-pvtz

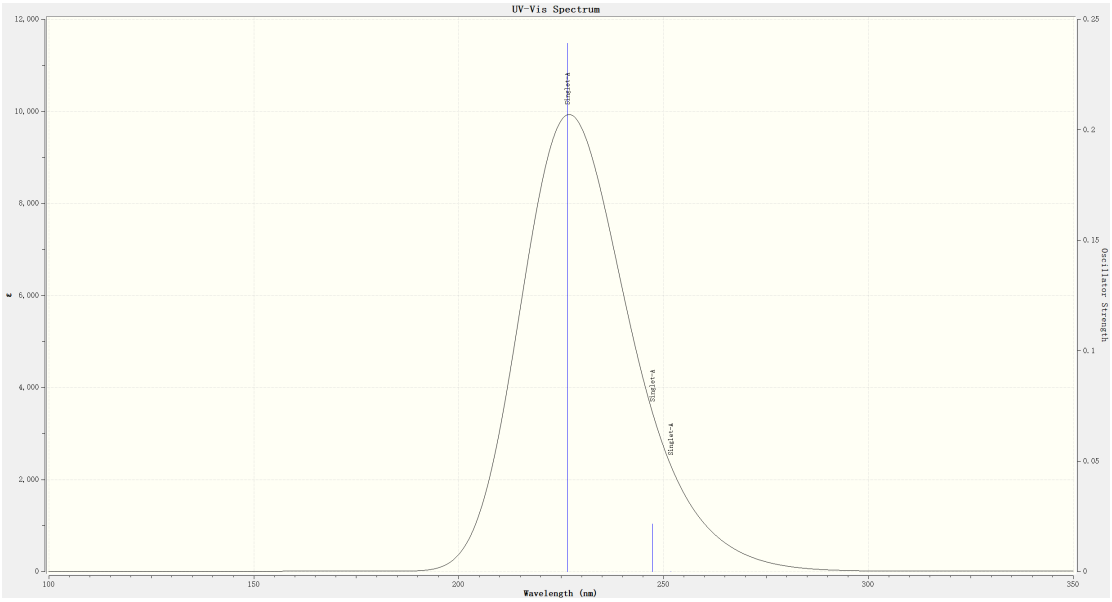
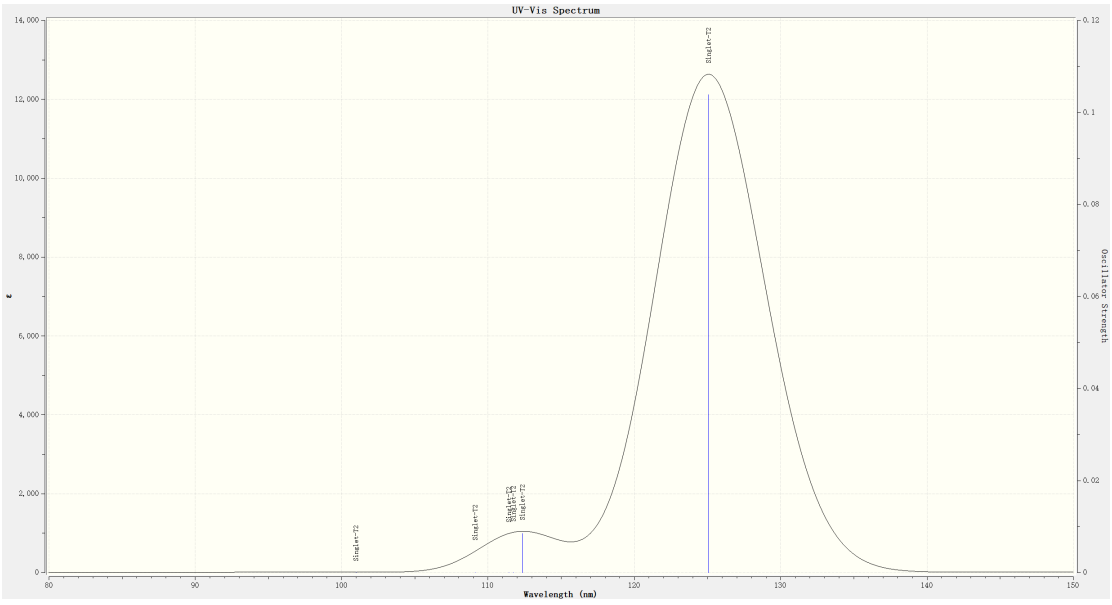


Figure 4: 甲烷 td=(nstates=15) CAM-B3LYP aug-cc-pvtz



物质	实验数据	PPT 数据	最终计算数据
山梨酸	254nm	255nm	250nm
阿司匹林	276nm	230nm 275nm	192nm 235nm 255nm
苯甲酸	220-240nm 250-280nm	140nm 161nm 208nm	226nm 247nm 208nm
甲烷	125nm	90nm	112nm 125nm

Table 2: 实验、PPT 和最终计算数据对比

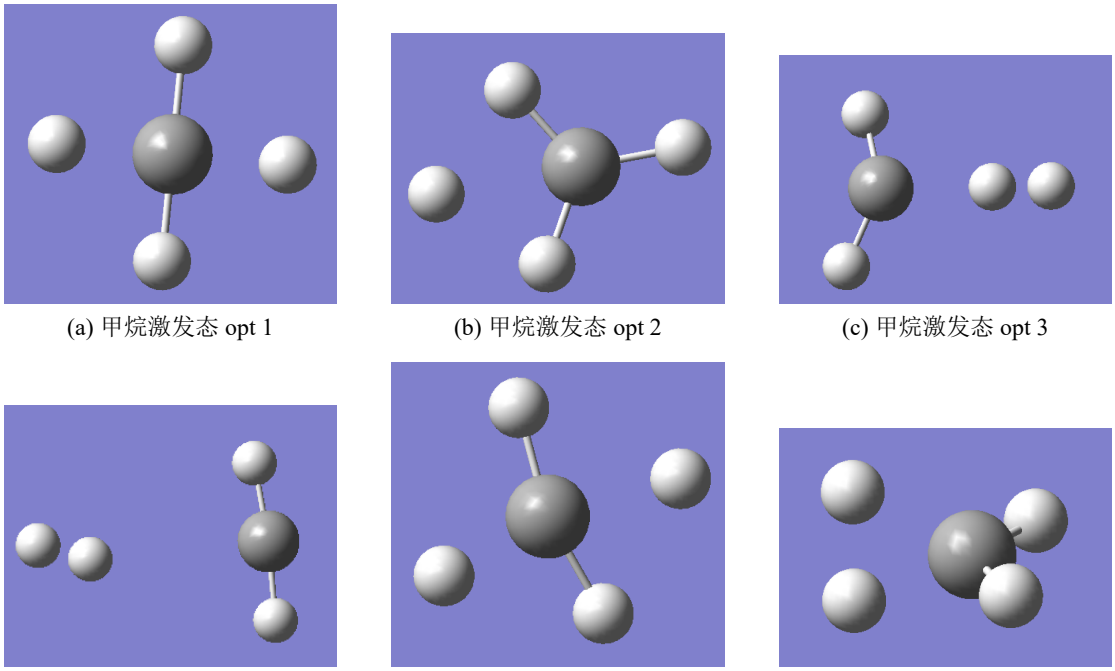
经过三组数据的比对，本次计算实验的结果基本符合预期，除了没有足够的时间资源计算较多 state 的情况。

4 实验分析与结论

我们发现，使用 CIS 与低精度的基组使用 TD-SCF 计算出的结果会与实验结果产生较大的误差，因此我们选用了适用于激发态的 td cam-b3lyp/aug-cc-pvtz 作为计算方法，在这个方法下我们可以花费较长的计算时间完成较高精度的计算。

另外，我们可以发现，在 PPT 中展示的计算方法完全是错误的。同时进行 opt 和 TD-SCF 计算得到的结果是激发态的结构优化，这并不是我们期望的结果。例如甲烷，会在这种条件下，使用不同方法，得到各种大部分是不收敛的激发态结构（error）。其中一些例子如下。

Figure 5: 甲烷激发态 error



(a) 甲烷激发态 opt 1 (b) 甲烷激发态 opt 2 (c) 甲烷激发态 opt 3

(d) 甲烷激发态 opt 4 (e) 甲烷激发态 opt 5 (f) 甲烷激发态 opt 6

failed 描述的是由于花费时间过长或者核心调度出现 bug 而失败的计算方案。

紫外光谱的数据来源都较早且不规范，而且小于 200nm 的峰的实验数据非常少见，因此选择高精度基组是有必要的。

特别需要提到的是，PPT 中的甲烷的峰很有可能是错误的，并没有文献支持这个数据，而在某些没有引用的资料中出现了 125nm，这和本次计算实验中的高精度计算结果是一致的。

需要额外注意的是，如果当前的体系结构较为复杂，可以使用 td=(nstates=？) 来指定计算？个激发态，在对比实验中我们已经证实了在某些情况下？的值会对谱图产生显著的影响。由于计算资源有限，我们仅对部分化合物进行了较高的？的计算。

5 附录

```python
# app.py 用来识别log文件的完成情况并分组
import os
import csv
from collections import defaultdict

def extract_calculation_method(filename):
    with open(filename, 'r', encoding='utf-8') as file:
        lines = file.readlines()
    for line in lines:
        if line.startswith('#'):
            calculation_method = line[1:].strip()  # 返回除了 '#' 外的所有内
                容
            return calculation_method
    return None

def get_calculation_status(filename):
    with open(filename, 'r', encoding='utf-8') as file:
        content = file.read()
    if 'Error termination' in content:
        return ' (error)'
    elif 'Job cpu time' in content:
        return ''
    else:
        return ' (failed)'

def main():
    directory = '.'  # Assuming the current directory, change this to the
        directory path
    files = os.listdir(directory)
    log_files = [f for f in files if f.endswith('.LOG')]
    gjf_files = [f for f in files if f.endswith('.gjf')]
    gjf_files_dict = {f.split('.')[0].lower(): f for f in gjf_files}  # 将文
        件名转换为小写

    categorized_results = defaultdict(list)

    for log_file in log_files:
        base_name = log_file.split('.')[0].lower()  # 将文件名转换为小写
        gjf_file = gjf_files_dict.get(base_name)
        if gjf_file:
            category_key = base_name.split('-')[1]  # 以第三个字符作为分类键
            calculation_method = extract_calculation_method(gjf_file)
            status = get_calculation_status(log_file)
            categorized_results[category_key].append((log_file, f"{
                calculation_method}{status}"))
        else:
            print(f"No matching gjf file found for log_file: {log_file}")

    for category, results in categorized_results.items():
        with open(f'results_{category}.csv', 'w', newline='', encoding='utf-8
            ') as file:
            writer = csv.writer(file)
            writer.writerow(['Filename', 'Calculation Method'])
            writer.writerows(results)

if __name__ == '__main__':
    main()
```