МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КПІ ім. Ігоря Сікорського

Фізико-технічний інститут

$$\psi_k = \sum_\mu c_{\mu k}\chi_\mu \quad \text{(LCAO)}$$
$$\hat{F}\varphi_i = \varepsilon_i\varphi_i$$
$$S_{\mu\nu} = \int \chi_\mu^*(\mathbf{r})\chi_\nu(\mathbf{r})\,d\mathbf{r}$$

С. М. Пономаренко

Квантово-механічні методи обчислення

Використання PySCF

$$\left[-\frac{1}{2}\nabla_i^2 + \sum_{j\neq i}\frac{1}{r_{ij}} + V_{\text{nuc}}(i)\right]\varphi_i = \varepsilon_i\varphi_i$$
$$\Psi = \det\begin{pmatrix} \varphi_1(1) & \varphi_2(1) & \cdots & \varphi_N(1) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(N) & \varphi_2(N) & \cdots & \varphi_N(N) \end{pmatrix}$$

КИЇВ 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

С. М. Пономаренко

Квантово-механічні методи обчислення
Використання PySCF

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для здобувачів ступеня магістра за спеціальностями Е6 «Прикладна фізика та наноматеріали»

КИЇВ
КПІ ім. Ігоря Сікорського
2025

Зміст

1 Властивості молекул

Після розрахунку електронної структури молекули природним кроком є обчислення її спостережуваних властивостей: коливальних спектрів, оптичних переходів, електричних та магнітних характеристик. PySCF надає потужні інструменти для розрахунку цих властивостей на рівні теорії Хартрі–Фока та пост-ХФ методів.

1.1. Коливальні властивості молекул

1.1.1. Теоретичні основи

Коливання атомів у молекулі відбуваються навколо рівноважної геометрії. Для малих відхилень потенційна енергія апроксимується квадратичною функцією:

$$E(\mathbf{R}) \approx E(\mathbf{R}_0) + \frac{1}{2}\sum_{ij} H_{ij}\Delta R_i \Delta R_j,$$

де H_{ij} — елементи матриці других похідних енергії (гесіану):

$$H_{ij} = \left.\frac{\partial^2 E}{\partial R_i \partial R_j}\right|_{\mathbf{R}_0}$$

Гесіан має розмірність $3N \times 3N$ для молекули з N атомів та містить повну інформацію про коливальні властивості.

1.1.2. Обчислення гесіану

PySCF може обчислювати гесіан аналітично або чисельно:

h2o_hessian.py

```
# ============================================================
# h2o_hessian.py
# Обчислення гесіану (матриці других похідних) для H2O
# ============================================================

from pyscf import gto, scf
from pyscf.hessian import rhf as rhf_hess
```

```python
# Створюємо молекулу води
mol = gto.M(
    atom="""
    O  0.0000  0.0000  0.1173
    H  0.0000  0.7572 -0.4692
    H  0.0000 -0.7572 -0.4692
    """,
    basis="6-31g",
    unit="angstrom",
)

print("Молекула H2O")
print("=" * 60)

# Спочатку SCF розрахунок
mf = scf.RHF(mol)
mf.kernel()

print("\nОбчислення гесіану...")
# Аналітичний гесіан (точний і швидкий)
hess = mf.Hessian()
h = hess.kernel()

print(f"Розмірність гесіану: {h.shape}")
print(f"Гесіан симетричний: {np.allclose(h, h.T)}")

# Виведемо кілька елементів
print("\nПриклад елементів гесіану (Ha/bohr²):")
print(f"H[0,0] = {h[0,0]:.6f}")  # ∂²E/∂x₁²
print(f"H[0,1] = {h[0,1]:.6f}")  # ∂²E/∂x₁∂y₁
print(f"H[1,1] = {h[1,1]:.6f}")  # ∂²E/∂y₁²
```

Важливі моменти:

- Аналітичний гесіан доступний для RHF, UHF, RKS, UKS
- Чисельний гесіан використовує скінченні різниці: $H_{ij} \approx [E(R_i + h) - 2E(R_i) + E(R_i - h)]/h^2$
- Аналітичний метод точніший та швидший
- Гесіан обчислюється в декартових координатах

Структура гесіану для H_2O:

- Розмірність: 9×9 (3 атоми $\times$ 3 координати)
- Симетричний: $H_{ij} = H_{ji}$
- Позитивно визначений у мінімумі енергії
- Має 6 нульових власних значень (трансляції та обертання)

1.1.3. Нормальні моди коливань

Для знаходження нормальних мод диагоналізуємо масо-зважений гесіан:

$$\mathbf{M}^{-1/2}\mathbf{H}\mathbf{M}^{-1/2}\mathbf{L} = \mathbf{L}\Lambda,$$

де $\mathbf{M}$ — діагональна матриця мас атомів, $\mathbf{L}$ — власні вектори (нормальні моди), Λ — власні значення.

Частоти коливань:

$$\omega_k = \sqrt{\lambda_k}, \quad \nu_k = \frac{\omega_k}{2\pi c}$$

h2o_frequencies.py

```python
# ============================================================
# h2o_frequencies.py
# Розрахунок частот коливань для H2O
# ============================================================

from pyscf import gto, scf
from pyscf.hessian import thermo
import numpy as np

# Молекула води (оптимізована геометрія)
mol = gto.M(
    atom="""
    O  0.0000  0.0000  0.1173
    H  0.0000  0.7572 -0.4692
    H  0.0000 -0.7572 -0.4692
    """,
    basis="6-31g",
    unit="angstrom",
)

print("Розрахунок коливальних частот H2O")
print("=" * 60)

# SCF розрахунок
mf = scf.RHF(mol)
mf.kernel()

# Обчислюємо гесіан
hess = mf.Hessian()
h = hess.kernel()

# Аналіз частот
freq_info = thermo.harmonic_analysis(mol, h)

print("\nКоливальні частоти:")
print("-" * 60)
print(f"{'№':<5} {'Частота (см⁻¹)':<20} {'Тип'}")
print("-" * 60)

# Виведемо тільки справжні коливання (без трансляцій/обертань)
modes = freq_info['freq_wavenumber']
for i, freq in enumerate(modes):
    if freq > 100:  # Фільтруємо дуже малі частоти
        mode_type = "коливання"
        print(f"{i+1:<5} {freq:>15.1f}     {mode_type}")

print("\nПорівняння з експериментом:")
print("ν₁ (симетр. валент.):  ~3657 см⁻¹")
print("ν₂ (деформаційна):     ~1595 см⁻¹")
print("ν₃ (антисим. валент.): ~3756 см⁻¹")
print("\nПримітка: RHF/6-31G завищує частоти на ~10-15%")
```

Аналіз результатів для H_2O:

Мода	Тип	ν (см$^{-1}$)	Опис
ν_1	A_1	3657	Симетричне валентне
ν_2	A_1	1595	Деформаційне (ножиці)
ν_3	B_2	3756	Антисиметричне валентне

Порівняння з експериментом:

- RHF/6-31G: завищує частоти на 10–15%
- RHF/cc-pVTZ: завищує на 8–12%
- B3LYP/cc-pVTZ: відхилення $< 5\%$ (з масштабуванням)
- Типовий масштабуючий фактор для RHF: 0.89

1.1.4. ІЧ-спектр: інтенсивності

Інтенсивність ІЧ-поглинання визначається зміною дипольного моменту:

$$I_k \propto \left| \frac{\partial \mu}{\partial Q_k} \right|^2,$$

де Q_k — нормальна координата k-ї моди, μ — дипольний момент.

Правила відбору:

- ІЧ-активна: $\partial\mu/\partial Q_k \neq 0$
- Залежить від симетрії молекули
- Для H_2O (група C_{2v}): всі три моди ІЧ-активні

h2o_ir_spectrum.py

```python
# ============================================================
# h2o_ir_spectrum.py
# Розрахунок ІЧ-спектру (частоти + інтенсивності)
# ============================================================

from pyscf import gto, scf
from pyscf.hessian import rhf as rhf_hess
from pyscf.prop import infrared
import numpy as np

mol = gto.M(
    atom="""
    O  0.0000  0.0000  0.1173
    H  0.0000  0.7572 -0.4692
    H  0.0000 -0.7572 -0.4692
    """,
    basis="6-31g",
    unit="angstrom",
)

print("Розрахунок ІЧ-спектру H2O")
print("=" * 60)

# SCF розрахунок
mf = scf.RHF(mol)
mf.kernel()
```

```
# Обчислюємо похідні дипольного моменту
# та частоти одночасно
ir_data = infrared.rhf.Infrared(mf)
ir_data.kernel()

print("\nІЧ-активні моди:")
print("-" * 70)
print(f"{'Мода':<8} {'ν (см⁻¹)':<15} {'Інтенсивність (км/моль)':<25}")
print("-" * 70)

for i, (freq, intensity) in enumerate(zip(ir_data.freq, ir_data.ir_inten)):
    if freq > 100:  # Тільки справжні коливання
        print(f"{i+1:<8} {freq:>12.1f}    {intensity:>20.2f}")

print("\nПримітка:")
print("- Інтенсивність залежить від зміни дипольного моменту")
print("- Деформаційна мода (ножиці) найінтенсивніша для H2O")


# ============================================================
# h2o_raman_activity.py
# Розрахунок Раман-активності (потребує похідних поляризовності)
# ============================================================

from pyscf import gto, scf, lib
from pyscf.hessian import thermo
from pyscf.prop import polarizability
import numpy as np

mol = gto.M(
    atom="""
    O  0.0000  0.0000  0.1173
    H  0.0000  0.7572 -0.4692
    H  0.0000 -0.7572 -0.4692
    """,
    basis="6-31g",
    unit="angstrom",
)

print("Розрахунок Раман-активності H2O")
print("=" * 60)

# SCF розрахунок
mf = scf.RHF(mol)
mf.kernel()

# Обчислюємо статичну поляризовність
alpha = polarizability.rhf.Polarizability(mf).polarizability()

print("\nСтатична поляризовність (au³):")
print("α□□ = {:.4f}".format(alpha[0, 0]))
print("α□□ = {:.4f}".format(alpha[1, 1]))
print("α□□ = {:.4f}".format(alpha[2, 2]))

# Середня поляризовність
alpha_mean = np.trace(alpha) / 3
print(f"\nСередня поляризовність: {alpha_mean:.4f} au³")

# Для повного Раман-спектру потрібні похідні поляризовності
```

```
# по нормальних координатах (складніший розрахунок)
print("\nПримітка:")
print("Повний Раман-спектр потребує обчислення ∂α/∂Q")
print("Це вимагає чисельного диференціювання або CPHF")
```

Типові інтенсивності для H_2O:

Мода	ν (см$^{-1}$)	I (км/моль)
ν_1 (симетр. валент.)	3657	5
ν_2 (деформаційна)	1595	73
ν_3 (антисим. валент.)	3756	58

Фізична інтерпретація:

- Деформаційна мода найінтенсивніша (велика зміна μ)
- Симетричне валентне — найслабше (компенсація)
- Інтенсивність залежить від полярності зв'язків

1.1.5. Раманівський спектр

На відміну від ІЧ, активність у Раман-спектрі визначається поляризовністю α:

$$I_k^{\text{Raman}} \propto \left| \frac{\partial \alpha}{\partial Q_k} \right|^2$$

Правила відбору:

- Раман-активна: $\partial\alpha/\partial Q_k \neq 0$
- Відрізняється від ІЧ (принцип взаємовиключення для центросиметричних)
- Для H_2O: всі три моди також Раман-активні

h2o_raman_activity.py

```
# ============================================================
# h2o_raman_activity.py
# Розрахунок Раман-активності (потребує похідних поляризовності)
# ============================================================

from pyscf import gto, scf, lib
from pyscf.hessian import thermo
from pyscf.prop import polarizability
import numpy as np

mol = gto.M(
    atom="""
    O  0.0000  0.0000  0.1173
    H  0.0000  0.7572 -0.4692
    H  0.0000 -0.7572 -0.4692
    """,
    basis="6-31g",
    unit="angstrom",
)
```

```
print("Розрахунок Раман-активності H2O")
print("=" * 60)

# SCF розрахунок
mf = scf.RHF(mol)
mf.kernel()

# Обчислюємо статичну поляризовність
alpha = polarizability.rhf.Polarizability(mf).polarizability()

print("\nСтатична поляризовність (au³):")
print("α𝑥𝑥 = {:.4f}".format(alpha[0, 0]))
print("α𝑦𝑦 = {:.4f}".format(alpha[1, 1]))
print("α𝑧𝑧 = {:.4f}".format(alpha[2, 2]))

# Середня поляризовність
alpha_mean = np.trace(alpha) / 3
print(f"\nСередня поляризовність: {alpha_mean:.4f} au³")

# Для повного Раман-спектру потрібні похідні поляризовності
# по нормальних координатах (складніший розрахунок)
print("\nПримітка:")
print("Повний Раман-спектр потребує обчислення ∂α/∂Q")
print("Це вимагає чисельного диференціювання або CPHF")


# ============================================================
# h2o_thermochemistry.py
# Термохімічні поправки (ZPE, ентальпія, ентропія)
# ============================================================

from pyscf import gto, scf
from pyscf.hessian import thermo

mol = gto.M(
    atom="""
    O  0.0000  0.0000  0.1173
    H  0.0000  0.7572 -0.4692
    H  0.0000 -0.7572 -0.4692
    """,
    basis="6-31g",
    unit="angstrom",
)

print("Термохімічний аналіз H2O")
print("=" * 60)

# SCF розрахунок
mf = scf.RHF(mol)
e_elec = mf.kernel()

# Гесіан та частоти
hess = mf.Hessian()
h = hess.kernel()

# Термохімічні функції при T=298.15 K, p=1 atm
results = thermo.thermo(mf, h, 298.15, 101325)

print("\nЕлектронна енергія:")
```

```
print(f"E(elec) = {e_elec:.6f} Ha")
print(f"        = {e_elec * 627.509:.2f} ккал/моль")

print("\nПоправки при 298.15 K:")
print(f"Нульова коливальна енергія (ZPE): {results['ZPE']:.6f} Ha")
print(f"Термічна поправка до енергії:     {results['E_thermal']:.6f} Ha")
print(f"Термічна поправка до ентальпії:   {results['H_thermal']:.6f} Ha")

print("\nПовна енергія Гіббса:")
print(f"G(298K) = E(elec) + ZPE + H_thermal - T*S")
print(f"        = {results['G_total']:.6f} Ha")

print("\nЕнтропія:")
print(f"S = {results['S']:.3f} кал/(моль·K)")

print("\nРозклад ZPE по модах:")
for i, freq in enumerate(results['freqs']):
    if freq > 100:
        zpe_mode = 0.5 * freq * 1.4388  # см⁻¹ -> ккал/моль
        print(f"  Мода {i+1}: {freq:.1f} см⁻¹ → ZPE = {zpe_mode:.2f} ккал/моль")
```

Типові активності для H_2O:

Мода	ν (см$^{-1}$)	Раман-активність (Å^4/amu)
ν_1	3657	1.8
ν_2	1595	0.4
ν_3	3756	3.2

1.1.6. Термохімічні поправки

Використовуючи частоти коливань, можна обчислити термодинамічні функції в наближенні гармонічного осцилятора та жорсткого ротатора.

Нульова коливальна енергія (ZPE)

$$E_{\text{ZPE}} = \sum_k \frac{1}{2}\hbar\omega_k = \sum_k \frac{hc\nu_k}{2}$$

Для H_2O:

$$E_{\text{ZPE}} \approx \frac{1}{2}(3657 + 1595 + 3756)\ \text{см}^{-1} \times 1.439\ \text{ккал/(моль·см}^{-1}) \approx 13.3\ \text{ккал/моль}$$

Термічні поправки

При температурі T коливальний внесок:

$$E_{\text{vib}}(T) = \sum_k \frac{\hbar\omega_k}{\exp(\hbar\omega_k / k_B T) - 1}$$

Повна енергія при T:

$$E(T) = E_{\text{elec}} + E_{\text{ZPE}} + E_{\text{vib}}(T) + E_{\text{rot}}(T) + E_{\text{trans}}(T)$$

h2o_thermochemistry.py

```python
# ============================================================
# h2o_thermochemistry.py
# Термохімічні поправки (ZPE, ентальпія, ентропія)
# ============================================================

from pyscf import gto, scf
from pyscf.hessian import thermo

mol = gto.M(
    atom="""
    O  0.0000  0.0000  0.1173
    H  0.0000  0.7572 -0.4692
    H  0.0000 -0.7572 -0.4692
    """,
    basis="6-31g",
    unit="angstrom",
)

print("Термохімічний аналіз H2O")
print("=" * 60)

# SCF розрахунок
mf = scf.RHF(mol)
e_elec = mf.kernel()

# Гесіан та частоти
hess = mf.Hessian()
h = hess.kernel()

# Термохімічні функції при T=298.15 K, p=1 atm
results = thermo.thermo(mf, h, 298.15, 101325)

print("\nЕлектронна енергія:")
print(f"E(elec) = {e_elec:.6f} Ha")
print(f"        = {e_elec * 627.509:.2f} ккал/моль")

print("\nПоправки при 298.15 K:")
print(f"Нульова коливальна енергія (ZPE): {results['ZPE']:.6f} Ha")
print(f"Термічна поправка до енергії:     {results['E_thermal']:.6f} Ha")
print(f"Термічна поправка до ентальпії:   {results['H_thermal']:.6f} Ha")

print("\nПовна енергія Гіббса:")
print(f"G(298K) = E(elec) + ZPE + H_thermal - T*S")
print(f"        = {results['G_total']:.6f} Ha")

print("\nЕнтропія:")
print(f"S = {results['S']:.3f} кал/(моль·K)")

print("\nРозклад ZPE по модах:")
for i, freq in enumerate(results['freqs']):
    if freq > 100:
        zpe_mode = 0.5 * freq * 1.4388  # см⁻¹ -> ккал/моль
        print(f"  Мода {i+1}: {freq:.1f} см⁻¹ → ZPE = {zpe_mode:.2f} ккал/моль")
```

```python
# ============================================================
# h2co_tddft.py
# Розрахунок УФ-спектру формальдегіду методом TDDFT
# ============================================================

from pyscf import gto, scf, dft, tddft

# Молекула формальдегіду H2CO
mol = gto.M(
    atom="""
    C  0.0000  0.0000  0.0000
    O  0.0000  0.0000  1.2050
    H  0.0000  0.9428 -0.5876
    H  0.0000 -0.9428 -0.5876
    """,
    basis="aug-cc-pvdz",
    unit="angstrom",
)

print("TDDFT розрахунок для H2CO (формальдегід)")
print("=" * 60)

# Основний стан: DFT з функціоналом CAM-B3LYP
mf = dft.RKS(mol)
mf.xc = "cam-b3lyp"  # Range-separated функціонал
mf.kernel()

print(f"\nЕнергія основного стану: {mf.e_tot:.6f} Ha")

# TDDFT для збуджених станів
# Розраховуємо перші 5 синглетних збуджень
td = tddft.TDDFT(mf)
td.nstates = 5
td.kernel()

print("\nЗбуджені стани (синглети):")
print("-" * 80)
print(f"{'Стан':<8} {'ΔE (eV)':<12} {'λ (нм)':<12} {'f':<12} {'Характер'}")
print("-" * 80)

# Конвертуємо Hartree -> eV -> nm
au2ev = 27.2114
for i, e in enumerate(td.e):
    energy_ev = e * au2ev
    wavelength = 1240 / energy_ev  # eV -> nm
    osc_str = td.oscillator_strength()[i]

    # Простий аналіз характеру
    if i == 0:
        char = "n→π*"
    elif i == 1:
        char = "π→π*"
    else:
        char = "Рідберг/змішаний"

    print(f"S_{i+1:<6} {energy_ev:>10.3f}  {wavelength:>10.1f}  "
          f"{osc_str:>10.4f}  {char}")

print("\nПримітка:")
print("- Сила осцилятора f показує інтенсивність переходу")
```

```
print("- n→π* перехід слабкий (заборонений за симетрією)")
print("- π→π* перехід сильний (дозволений)")
```

Типові поправки для H_2O при 298.15 K:

- Електронна енергія: $E_{\text{elec}} = -76.067$ Ha
- ZPE: $+0.021$ Ha (13.3 ккал/моль)
- Термічна поправка: $+0.003$ Ha (1.9 ккал/моль)
- Ентропія: $S = 45.1$ кал/(моль·K)

1.2. Електронні переходи та УФ-видимі спектри

1.2.1. Теорія збуджених станів

Для розрахунку електронних переходів використовуємо методи:

- **CIS** (Configuration Interaction Singles) — базовий
- **TDHF/TDDFT** (Time-Dependent HF/DFT) — точніший
- **EOM-CCSD** (Equation-of-Motion CCSD) — високоточний

Енергія збудження n-го стану:

$$\omega_n = E_n - E_0$$

Сила осцилятора (інтенсивність):

$$f_n = \frac{2}{3}\omega_n |\langle \Psi_0 | \hat{\mu} | \Psi_n \rangle|^2$$

1.2.2. TDDFT розрахунок для формальдегіду

Розглянемо молекулу формальдегіду H☒CO як приклад:

h2co_tddft.py

```
# ============================================================
# h2co_tddft.py
# Розрахунок УФ-спектру формальдегіду методом TDDFT
# ============================================================

from pyscf import gto, scf, dft, tddft

# Молекула формальдегіду H2CO
mol = gto.M(
    atom="""
    C  0.0000  0.0000  0.0000
    O  0.0000  0.0000  1.2050
    H  0.0000  0.9428 -0.5876
    H  0.0000 -0.9428 -0.5876
    """,
    basis="aug-cc-pvdz",
    unit="angstrom",
)
```

```
print("TDDFT розрахунок для H2CO (формальдегід)")
print("=" * 60)

# Основний стан: DFT з функціоналом CAM-B3LYP
mf = dft.RKS(mol)
mf.xc = "cam-b3lyp"  # Range-separated функціонал
mf.kernel()

print(f"\nЕнергія основного стану: {mf.e_tot:.6f} Ha")

# TDDFT для збуджених станів
# Розраховуємо перші 5 синглетних збуджень
td = tddft.TDDFT(mf)
td.nstates = 5
td.kernel()

print("\nЗбуджені стани (синглети):")
print("-" * 80)
print(f"{'Стан':<8} {'ΔE (eV)':<12} {'λ (нм)':<12} {'f':<12} {'Характер'}")
print("-" * 80)

# Конвертуємо Hartree -> eV -> nm
au2ev = 27.2114
for i, e in enumerate(td.e):
    energy_ev = e * au2ev
    wavelength = 1240 / energy_ev  # eV -> nm
    osc_str = td.oscillator_strength()[i]

    # Простий аналіз характеру
    if i == 0:
        char = "n→π*"
    elif i == 1:
        char = "π→π*"
    else:
        char = "Рідберг/змішаний"

    print(f"S_{i+1:<6} {energy_ev:>10.3f}  {wavelength:>10.1f}  "
          f"{osc_str:>10.4f}  {char}")

print("\nПримітка:")
print("- Сила осцилятора f показує інтенсивність переходу")
print("- n→π* перехід слабкий (заборонений за симетрією)")
print("- π→π* перехід сильний (дозволений)")
```

Аналіз збуджень для H_2CO:

Перехід	Тип	λ (нм)	f	Характер
S_1	$n \to \pi^*$	355	0.001	Заборонений
S_2	$\pi \to \pi^*$	185	0.152	Дозволений
S_3	$n \to 3s$	172	0.021	Рідбергівський

Фізична інтерпретація:

- $n \to \pi^*$: неподілена пара O → антизв'язуюча МО C=O

- Низька сила осцилятора через симетрію
- $\pi \to \pi^*$: основна смуга поглинання в УФ
- Енергії залежать від функціоналу DFT

1.2.3. Вибір функціоналу для TDDFT

Різні функціонали дають різну точність для збуджень:

formaldehyde_functional_comparison.py

```python
# ============================================================
# formaldehyde_functional_comparison.py
# Порівняння різних функціоналів для TDDFT
# ============================================================

from pyscf import gto, dft, tddft

mol = gto.M(
    atom="""
    C  0.0000  0.0000  0.0000
    O  0.0000  0.0000  1.2050
    H  0.0000  0.9428 -0.5876
    H  0.0000 -0.9428 -0.5876
    """,
    basis="aug-cc-pvdz",
    unit="angstrom",
)

print("Порівняння функціоналів DFT для збуджень H2CO")
print("=" * 60)

# Список функціоналів для тестування
functionals = ["b3lyp", "pbe0", "cam-b3lyp", "wb97x-d"]

results = {}

for xc in functionals:
    print(f"\n{xc.upper()}:")
    print("-" * 40)

    # DFT розрахунок
    mf = dft.RKS(mol)
    mf.xc = xc
    mf.verbose = 0
    mf.kernel()

    # TDDFT
    td = tddft.TDDFT(mf)
    td.nstates = 3
    td.verbose = 0
    td.kernel()

    # Перший перехід (n→π*)
    energy_ev = td.e[0] * 27.2114
    wavelength = 1240 / energy_ev
    osc_str = td.oscillator_strength()[0]

    results[xc] = (energy_ev, wavelength, osc_str)
```

```
    print(f"  S₁: {energy_ev:.3f} eV ({wavelength:.1f} nm)")
    print(f"  f = {osc_str:.4f}")

print("\n" + "=" * 60)
print("Порівняння для n→π* переходу:")
print("-" * 60)
print(f"{'Функціонал':<15} {'λ (нм)':<12} {'Відхилення'}")
print("-" * 60)

exp_wavelength = 330  # Експериментальне значення (нм)

for xc, (e, wl, f) in results.items():
    diff = wl - exp_wavelength
    print(f"{xc.upper():<15} {wl:>10.1f}  {diff:+7.1f} нм")

print("-" * 60)
print(f"{'ЕКСПЕРИМЕНТ':<15} {exp_wavelength:>10.1f}  {'---'}")

print("\nВисновки:")
print("- Range-separated функціонали (CAM-B3LYP, ωB97X-D) найточніші")
print("- B3LYP завищує довжини хвиль")
print("- Для переносу заряду обов'язково використовувати range-separated")
```

Порівняння функціоналів (перехід $n \to \pi^*$):

Функціонал	λ (нм)	Похибка (нм)
B3LYP	355	+25
PBE0	342	+12
CAM-B3LYP	335	+5
ωB97X-D	332	+2
Експеримент	330	—

Рекомендації:

- Гібридні функціонали краще за чисті GGA
- Range-separated (CAM-B3LYP, ωB97X-D) найточніші
- Для переносу заряду обов'язково range-separated
- B3LYP часто завищує довжини хвиль

1.2.4. Візуалізація спектру

Для побудови спектру використовуємо гаусові або лоренцеві контури:

$$\varepsilon(\lambda) = \sum_n f_n \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(\lambda - \lambda_n)^2}{2\sigma^2}\right]$$

plot_uv_spectrum.py

```
# ============================================================
# plot_uv_spectrum.py
# Побудова УФ-спектру з уширенням
# ============================================================
```

```python
from pyscf import gto, dft, tddft
import numpy as np
import matplotlib.pyplot as plt

mol = gto.M(
    atom="""
    C  0.0000  0.0000  0.0000
    O  0.0000  0.0000  1.2050
    H  0.0000  0.9428 -0.5876
    H  0.0000 -0.9428 -0.5876
    """,
    basis="aug-cc-pvdz",
    unit="angstrom",
)

# TDDFT розрахунок
mf = dft.RKS(mol)
mf.xc = "cam-b3lyp"
mf.verbose = 0
mf.kernel()

td = tddft.TDDFT(mf)
td.nstates = 10
td.verbose = 0
td.kernel()

# Отримуємо енергії та сили осциляторів
energies = td.e * 27.2114  # Ha -> eV
wavelengths = 1240 / energies  # eV -> nm
osc_strengths = td.oscillator_strength()

print("Побудова УФ-спектру")
print("=" * 60)
print("\nЗбудження:")
for i, (wl, f) in enumerate(zip(wavelengths, osc_strengths)):
    if f > 0.001:  # Тільки значущі переходи
        print(f"S_{i+1}: λ={wl:.1f} nm, f={f:.4f}")

# Будуємо спектр з гаусовим уширенням
def gaussian(x, center, sigma):
    return np.exp(-(x - center)**2 / (2 * sigma**2)) / (sigma * np.sqrt(2*np.pi))

# Сітка довжин хвиль
wl_grid = np.linspace(150, 400, 1000)
spectrum = np.zeros_like(wl_grid)

# Параметр уширення
sigma = 10  # nm (для газової фази)

# Додаємо кожний перехід
for wl, f in zip(wavelengths, osc_strengths):
    if 150 < wl < 400:  # Тільки в УФ області
        spectrum += f * gaussian(wl_grid, wl, sigma)

# Малюємо
plt.figure(figsize=(10, 6))

# Штрих-спектр (окремі переходи)
plt.subplot(2, 1, 1)
```

```
for wl, f in zip(wavelengths, osc_strengths):
    if 150 < wl < 400 and f > 0.001:
        plt.stem([wl], [f], linefmt='b-', markerfmt='bo', basefmt=' ')
plt.xlim(150, 400)
plt.ylabel('Сила осцилятора')
plt.title('H₂CO: Штрих-спектр (TDDFT/CAM-B3LYP)')
plt.grid(alpha=0.3)

# Уширений спектр
plt.subplot(2, 1, 2)
plt.plot(wl_grid, spectrum, 'b-', linewidth=2)
plt.xlim(150, 400)
plt.xlabel('Довжина хвилі (нм)')
plt.ylabel('Інтенсивність (відн. од.)')
plt.title('Уширений спектр (σ=10 нм)')
plt.grid(alpha=0.3)

plt.tight_layout()
plt.savefig('h2co_uv_spectrum.pdf', dpi=300, bbox_inches='tight')
print("\nСпектр збережено у файл h2co_uv_spectrum.pdf")
```

Рис. 1.1. УФ-спектр формальдегіду, розрахований методом TDDFT/CAM-B3LYP/aug-cc-pVDZ.

Параметри уширення:

- Типове $\sigma = 0.3$–0.5 eV для конденсованої фази
- $\sigma = 0.1$–0.2 eV для газової фази
- Експериментальне уширення враховує розподіл за T

1.2.5. Аналіз характеру переходів

TDDFT надає інформацію про орбіталі, задіяні у переході:

analyze_transitions.py

```
# ============================================================
# analyze_transitions.py
# Детальний аналіз характеру електронних переходів
# ============================================================

from pyscf import gto, dft, tddft
import numpy as np

mol = gto.M(
    atom="""
    C  0.0000  0.0000  0.0000
    O  0.0000  0.0000  1.2050
    H  0.0000  0.9428 -0.5876
    H  0.0000 -0.9428 -0.5876
    """,
    basis="aug-cc-pvdz",
    unit="angstrom",
)
```

```
print("Детальний аналіз переходів H2CO")
print("=" * 60)

# DFT + TDDFT
mf = dft.RKS(mol)
mf.xc = "cam-b3lyp"
mf.verbose = 0
mf.kernel()

td = tddft.TDDFT(mf)
td.nstates = 5
td.verbose = 0
td.kernel()

# Аналізуємо кожен збуджений стан
for i in range(min(3, td.nstates)):  # Перші 3 стани
    energy_ev = td.e[i] * 27.2114
    wavelength = 1240 / energy_ev
    f = td.oscillator_strength()[i]

    print(f"\n{'='*60}")
    print(f"Збуджений стан {i+1}: {energy_ev:.3f} eV ({wavelength:.1f} nm)")
    print(f"Сила осцилятора f = {f:.4f}")
    print(f"{'='*60}")

    # Отримуємо амплітуди переходів X → Y
    # td.xy[i] містить (X, Y) амплітуди
    x_amp = td.xy[i][0]  # Амплітуди збудження

    # Індекси орбіталей
    nocc = mol.nelectron // 2  # Кількість зайнятих орбіталей

    print("\nОсновні внески (|амплітуда| > 0.1):")
    print(f"{'Перехід':<20} {'Амплітуда':<12} {'Внесок (%)'}")
    print("-" * 50)

    # Знаходимо значущі внески
    nvir = len(x_amp[0])  # Кількість віртуальних

    contributions = []
    for occ_i in range(nocc):
        for vir_i in range(nvir):
            amp = x_amp[occ_i, vir_i]
            contrib = amp**2 * 100  # У відсотках

            if abs(amp) > 0.1:
                homo_label = occ_i - nocc  # -1 для HOMO, -2 для HOMO-1, ...
                lumo_label = vir_i  # 0 для LUMO, 1 для LUMO+1, ...

                if homo_label == -1:
                    orb_from = "HOMO"
                elif homo_label < -1:
                    orb_from = f"HOMO{homo_label+1}"
                else:
                    orb_from = f"Occ{occ_i}"

                if lumo_label == 0:
                    orb_to = "LUMO"
                else:
```

```
                    orb_to = f"LUMO+{lumo_label}"

                transition = f"{orb_from} → {orb_to}"
                contributions.append((transition, amp, contrib))

    # Сортуємо за внеском
    contributions.sort(key=lambda x: x[2], reverse=True)

    for trans, amp, contrib in contributions[:5]:  # Топ-5
        print(f"{trans:<20} {amp:>10.4f}  {contrib:>10.1f}")

    # Інтерпретація
    if contributions:
        main_trans = contributions[0][0]
        print(f"\nДомінуючий перехід: {main_trans}")

        if "HOMO" in main_trans and "LUMO" in main_trans:
            print("Тип: одноелектронне збудження")
            if i == 0:
                print("Характер: n→π* (неподілена пара O → π* C=O)")
            elif i == 1:
                print("Характер: π→π* (зв'язуюча → антизв'язуюча)")

print("\n" + "="*60)
print("Примітка: Внески показують ймовірність кожного переходу")
print("Сума квадратів амплітуд дорівнює 1")
```

Приклад виводу для S_1 стану H_2CO:

```
Excited State 1: 3.492 eV (355 nm)  f=0.0012
    HOMO-1 -> LUMO     0.11 (1.2%)
    HOMO   -> LUMO     0.69 (47.6%)
    HOMO   -> LUMO+1   0.08 (0.6%)
```

Інтерпретація:

- Основний внесок: HOMO → LUMO (48%)
- HOMO — неподілена пара n(O)
- LUMO — антизв'язуюча π^*(C=O)
- Тип переходу: $n \to \pi^*$

Література

Основна література

1. *Jensen F.* Introduction to Computational Chemistry. — 3rd ed. — Wiley, 2017. — 661 p. — ISBN 1118825993.
2. *Levine I. N.* Quantum Chemistry. — 7th ed. — Pearson, 2014. — 714 p. — ISBN 978-0321803450.

Додаткові посилання

1. Basis Sets. — URL: https://gaussian.com/basissets/.
2. *Ho M.*, *Hernández-Perez J. M.* Evaluation of Gaussian Molecular Integrals. I Overlap Integrals // The Mathematica Journal. — 2012. — Vol. 14.
3. *Ho M.*, *Hernández-Perez J. M.* Evaluation of Gaussian Molecular Integrals. II. Kinetic-Energy Integrals // The Mathematica Journal. — 2013. — Vol. 15.
4. *Ho M.*, *Hernández-Perez J. M.* Evaluation of Gaussian Molecular Integrals. III. Nuclear-Electron Attraction Integrals // The Mathematica Journal. — 2014. — Vol. 16.
5. Simple Quantum Chemistry: Hartree-Fock in Python. — URL: https://nznano.blogspot.com/2018/03/simple-quantum-chemistry-hartree-fock.html.