1 Метод Хартрі-Фока для диатомних молекул

Після вивчення атомних систем природним кроком є перехід до молекул. Найпростіші диатомні молекули — іон молекули водню H_2^+ та молекула водню H_2 — є ідеальними об'єктами для демонстрації можливостей та обмежень методу Хартрі–Фока.

1.1. Орбіталі і електронні стани молекул

Електронна структура двоатомних молекул описується двома рівнями:

1. класифікацією **молекулярних орбіталей (MO)**;
2. класифікацією **електронних станів молекули**, які утворюються внаслідок заповнення цих MO електронами.

1.1.1. Класифікація молекулярних орбіталей

Молекулярні орбіталі позначаються символами σ, π, δ, φ, ..., що відповідають проєкції орбітального моменту Λ на між'ядерну вісь:

$$\Lambda = 0, 1, 2, 3, \ldots \quad \Rightarrow \quad \sigma,\ \pi,\ \delta,\ \varphi, \ldots$$

Кожна орбіталь може бути:

- **зв'язувальною** або **антизв'язувальною** (позначається зірочкою *);
- **симетричною (gerade, g)** або **антисиметричною (ungerade, u)** відносно інверсії в центрі молекули.

Таким чином, типовий запис молекулярної орбіталі має вигляд:

$$\sigma_g, \quad \pi_u, \quad \sigma_u^*, \quad \pi_g^*, \ldots$$

Зв'язок із симетрією: Парність g/u визначається поведінкою хвильової функції при інверсії координат:

$$\psi_g(\mathbf{r}) = +\psi_g(-\mathbf{r}), \qquad \psi_u(\mathbf{r}) = -\psi_u(-\mathbf{r}).$$

1.1.2. Класифікація електронних станів молекули

Електронний терм — це позначення квантового стану молекули, який характеризується певними значеннями повного спіну S, проєкції орбітального моменту на між'ядерну вісь Λ, та симетріями відносно інверсії (g/u) і відбиття ($+/-$). Він має вигляд:

$$^{2S+1}\Lambda^{\pm}_{g/u}.$$

де:

- S — повний спін системи;
- $2S+1$ — **мультиплетність** (синглет, дублет, триплет тощо);
- Λ — **сумарна проєкція орбітального моменту** на між'ядерну вісь:

$$\Sigma\ (\Lambda=0),\quad \Pi\ (\Lambda=1),\quad \Delta\ (\Lambda=2),\quad \Phi\ (\Lambda=3),\ldots$$

- верхній індекс $+$ або $-$ — **симетрія відносно площини, що містить вісь молекули** (лише для станів типу Σ);
- нижній індекс g/u (від *gerade*/*ungerade*) — **парність хвильової функції** відносно інверсії в центрі молекули.

Типові терми двоатомних молекул

Терм	Мультиплетність	Симетрія	Фізичний зміст
$^1\Sigma_g^+$	Синглет	g, $+$	Основний стан молекули H_2; усі електрони спарені, повна симетрія
$^3\Sigma_u^+$	Триплет	u, $+$	Збуджений стан з паралельними спінами (наприклад, у O_2)
$^1\Pi_u$	Синглет	u	Орбітальний момент $\Lambda=1$, без інверсійної симетрії
$^3\Pi_g$	Триплет	g	Орбітальний момент $\Lambda=1$, паралельні спіни
$^1\Delta_g$	Синглет	g	Орбітальний момент $\Lambda=2$, спарені електрони
$^2\Sigma_u^+$	Дублет	u, $+$	Один неспарений електрон, як у радикалах типу NO

Нижче наведено покроковий алгоритм визначення терму для заданої електронної конфігурації.

Алгоритм

1. **Записати електронну конфігурацію.** Визначити, які орбіталі заповнені, частково заповнені чи вакантні:

$$(\sigma_g)^2(\sigma_u^*)^2(\pi_u)^4(\pi_g^*)^2, \ldots$$

2. **Визначити тип орбіталей і відповідні проєкції орбітального моменту.** Для кожної орбіталі встановити:

$$\sigma \to \Lambda_i = 0, \quad \pi \to \Lambda_i = 1, \quad \delta \to \Lambda_i = 2, \quad \text{тощо.}$$

3. **Знайти можливі комбінації орбітальних моментів.** Для активних (частково заповнених) орбіталей обчислити всі можливі значення:

$$\Lambda = |\Lambda_1 + \Lambda_2|, \, |\Lambda_1 - \Lambda_2|, \ldots$$

   Це дає можливі типи станів: $\Sigma, \Pi, \Delta, \ldots$
4. **Комбінувати спіни електронів.** Для кожного електрона $S_i = \frac{1}{2}$. Обчислити всі можливі значення повного спіну:

$$S = |S_1 + S_2|, \, |S_1 - S_2|, \ldots$$

   Відповідно визначається мультиплетність $2S{+}1$ (синглет, дублет, триплет тощо).
5. **Перевірити принцип Паулі.** Загальна хвильова функція має бути антисиметричною при перестановці двох електронів:

$$\Psi_{\text{заг}} = \Psi_{\text{просторова}} \, \Psi_{\text{спінова}} \, \Psi_{\text{симетрії}}.$$

   Якщо просторова частина симетрична $\Rightarrow S = 0$ (синглет), якщо антисиметрична $\Rightarrow S = 1$ (триплет).
6. **Визначити симетрії g/u і $+/-$.**
   - Індекс g/u показує парність хвильової функції при інверсії в центрі молекули.
   - Для станів Σ додається верхній індекс $+/-$, що вказує на симетрію при відбитті у площині, яка містить між'ядерну вісь.
7. **Записати всі можливі терми.** Комбінуючи знайдені S, Λ, g/u та $+/-$, записуємо:

$$^{2S+1}\Lambda^{\pm}_{g/u}.$$

8. **Визначити основний терм за правилами Гунда.**
   8.1. Найнижчу енергію має стан з **максимальним спіном** S.
   8.2. Для однакового S — стан з **найбільшим** Λ.
   8.3. Для даних S, Λ — визначається парність (g/u) і знак $(+/-)$ з урахуванням симетрії конфігурації.

Приклад. Для молекули водню H_2 основний електронний стан має конфігурацію σ_g^2. Оскільки обидва електрони спарені ($S = 0$), повний терм записується як:

$$^{2S+1}\Lambda_g^+ = {}^1\Sigma_g^+.$$

Це **синглетний стан** ($S = 0$), симетричний відносно інверсії (індекс g) і з позитивною симетрією відносно площини, що містить між'ядерну вісь ($+$).

1.2. Початкові модельні молекули: H_2^+ та H_2

Першим кроком у квантово-хімічному описі молекул є вивчення найпростіших систем, для яких можна чітко простежити природу хімічного зв'язку. Молекули H_2^+ та H_2 є **модельними** у тому сенсі, що вони:

- мають найменшу можливу кількість електронів (один і два відповідно);
- дозволяють побудувати хвильові функції, що безпосередньо демонструють утворення зв'язувальних і розривних (антизв'язувальних) орбіталей;
- відтворюють основні риси електронної структури складніших молекул, але без зайвих ускладнень багаточастинкової взаємодії;
- зручні для порівняння точних, наближених і експериментальних результатів;
- використовуються як тестові системи для перевірки методів квантової хімії.
- **H_2^+** — найпростіша двоатомна молекула, що містить один електрон. Її можна розв'язати аналітично в еліптичних координатах, тому вона є класичним прикладом формування зв'язувальної орбіталі σ_g;
- **H_2** — найпростіша двоелектронна молекула, у якій вперше проявляється електронна кореляція та обмеження методу Хартрі–Фока.

Таким чином, системи H_2^+ і H_2 є **відправною точкою** для розуміння природи хімічного зв'язку, ролі симетрії, утворення молекулярних орбіталей і спінових станів.

1.3. Іон молекули водню H_2^+

1.3.1. Теоретичні основи

Іон H_2^+ — це найпростіша молекулярна система, яка складається з двох протонів та одного електрона. Незважаючи на її простоту, саме на прикладі H_2^+ вперше стає очевидною суть хімічного зв'язку як результату квантового перекривання хвильових функцій.

У нерелятивістському наближенні (та з використанням атомних одиниць) гамільтоніан системи має вигляд:

$$\hat{H} = -\frac{1}{2}\nabla^2 - \frac{1}{r_A} - \frac{1}{r_B} + \frac{1}{R},$$

де r_A, r_B — відстані електрона до ядер A і B, а R — між'ядерна відстань.

Перші три члени описують кінетичну енергію електрона та його притягання до двох ядер, а останній доданок — кулонівське відштовхування між ядрами.

Оскільки у системі лише один електрон, тут *відсутні електрон-електронні взаємодії*. Тому метод Хартрі–Фока не містить ніяких додаткових апроксимацій (крім обмежень базису), і його результат збігається з точною розв'язною енергією при нескінченному базисі. Таким чином, H_2^+ — це ідеальний тест для перевірки коректності реалізації ХФ-методу в програмних пакетах.

1.3.2. Визначення молекули в PySCF

Для визначення молекули в PySCF використовується рядкова нотація:

h2plus_single.py

```
"""
Розрахунок іона молекули водню H2+ при фіксованій відстані
та аналіз молекулярних орбіталей
"""

import numpy as np
import matplotlib.pyplot as plt

from pyscf import gto, scf

# ------------------------------------------------------------
# Визначення молекули
# ------------------------------------------------------------
mol = gto.Mole()
mol.atom = """
    H  0.0  0.0  0.0
    H  0.0  0.0  0.74
"""
mol.unit = "Angstrom"
mol.charge = 1      # заряд +1 (один електрон)
mol.spin = 1        # 2S = 1 (дублет)
mol.basis = "sto-3g"
mol.build()

# ------------------------------------------------------------
# UHF-розрахунок
# ------------------------------------------------------------
mf = scf.UHF(mol)
energy = mf.kernel()

# ------------------------------------------------------------
# Вивід результатів
# ------------------------------------------------------------
print("\n" + "=" * 60)
print("Іон молекули водню H2+")
print("=" * 60)
print(f"Міжядерна відстань R = 0.74 Å = {0.74 / 0.529177:.3f} bohr")
print(f"Базисний набір: {mol.basis}")
print(f"Повна енергія: {energy:.6f} Ha")
```

```
print(f"Кількість електронів: α={mol.nelec[0]}, β={mol.nelec[1]}")

# ------------------------------------------------------------
# Орбітальні енергії
# ------------------------------------------------------------
eps = mf.mo_energy[0]  # α-спінові енергії
homo, lumo = eps[0], eps[1]
gap = lumo - homo

print("\nОрбітальні енергії (Ha):")
print(f"  HOMO (зв'язуюча σg): {homo: .4f}")
print(f"  LUMO (антизв'язуюча σu*): {lumo: .4f}")
print(f"  HOMO-LUMO gap = {gap:.4f} Ha = {gap * 27.211:.2f} eV")

# ------------------------------------------------------------
# Спіновий стан
# ------------------------------------------------------------
s2, mult = mf.spin_square()
print(f"\n<S^2> = {s2:.4f}  (очікується 0.75 для S=1/2)")
print(f"Мультиплетність: 2S+1 = {mult}")
print("=" * 60)
```

Пояснення коду:

- `atom = 'H 0 0 0; H 0 0 0.74'` — координати атомів (Ангстреми за замовчуванням)
- `basis = 'sto-3g'` — мінімальний базис
- `charge = 1` — заряд системи ($+1$ для H_2^+)
- `spin = 1` — $2S = N_\alpha - N_\beta = 1$
- `scf.UHF` — необмежений ХФ (один непарний електрон)

1.3.3. Інтерпретація результатів

Вивід програми містить основні характеристики розрахованої системи H_2^+ при між'ядерній відстані

$$R = 0.74 \text{ Å} = 1.399\ a_0,$$

у базисному наборі STO-3G.

- **Повна енергія:**

$$E = -0.538205 \text{ Ha},$$

що є нижчою за енергію вільного атома водню (-0.5 Ha). Це свідчить про стабілізацію системи внаслідок утворення молекулярного зв'язку — електрон делокалізується між двома ядрами, зменшуючи електростатичне відштовхування між протонами.

- **Енергії орбіталей (для α-спіну):**

$$\varepsilon_{\text{HOMO}} = -1.2533 \text{ Ha},$$
$$\varepsilon_{\text{LUMO}} = -0.4751 \text{ Ha},$$

$$\Delta_{\text{H}-\text{L}} = 0.7782 \text{ Ha} = 21.18 \text{ eB}.$$

HOMO має симетрію σ_g і є зв'язувальною орбіталлю, тоді як LUMO відповідає антизв'язувальному стану σ_u^*. Значний енергетичний розрив між ними свідчить про сильний і стабільний зв'язок.

- **Спіновий стан:**

$$\langle S^2 \rangle = 0.75 \quad \Rightarrow \quad S = \frac{1}{2}, \quad 2S + 1 = 2.$$

Це відповідає дублетному стану з одним неспареним електроном, що характерно для H_2^+.

Таким чином, результати розрахунку підтверджують, що при відстані $R = 0.74$ Å іон H_2^+ перебуває у стабільному зв'язаному стані.

1.3.4. Крива потенційної енергії (PES)

Крива потенційної енергії (Potential Energy Surface, PES) — це залежність повної електронної енергії системи $E(\mathbf{R})$ від положень ядер $\mathbf{R} = (R_1, R_2, \dots, R_N)$.

Для загальної N-атомної молекули PES — це багатовимірна поверхня у $3N-6$ координатах (три координати на кожне ядро мінус три для поступального руху і три для обертання). У випадку двоатомної молекули (наприклад, H_2^+) поверхня зводиться до *одновимірної кривої* $E(R)$, де R — відстань між ядрами:

$$E(R) = \text{мінімальна енергія системи при фіксованому } R.$$

Фізичний зміст:

- Мінімум $E(R)$ відповідає *рівноважній відстані* R_{eq}, де сили на ядра взаємно компенсуються ($\partial E/\partial R = 0$);
- Глибина ями потенціалу визначає *енергію зв'язку*;
- Крутизна поблизу мінімуму пов'язана з *жорсткістю зв'язку* та *частотою коливань*;
- Для великих R енергія прямує до суми енергій ізольованих атомів.

Обчислення PES у квантовій хімії полягає у серії незалежних SCF-розрахунків при різних значеннях R:

$$R = 0.5,\ 0.7,\ 1.0,\ 1.5,\ 2.0,\ \dots \text{ Å}.$$

На кожному кроці визначається повна енергія $E(R_i)$, після чого будується графік $E(R)$, який ілюструє поведінку потенціальної енергії системи.

Типова форма PES для H_2^+:

- На малих R енергія різко зростає через кулонівське відштовхування ядер;
- Поблизу $R \approx 0.74$ Å — глибокий мінімум (стабільний хімічний зв'язок);
- При $R \to \infty$ енергія прямує до енергії одного атома Гідрогену $E = -0.5$ Ha.

Сканування по відстанях

Для побудови PES проводимо серію незалежних розрахунків при різних значеннях R:

h2plus_pes.py

```
"""
Побудова кривої потенційної енергії (PES) для H2+
"""

import numpy as np
import matplotlib.pyplot as plt
from pyscf import gto, scf

# Діапазон міжядерних відстаней (у борах)
distances = np.linspace(0.5, 5.0, 30)  # від 0.5 до 5.0 bohr
energies = []

print("Розрахунок PES для H2+...")
print("=" * 60)

for R in distances:
    # Створюємо молекулу з новою геометрією
    mol = gto.Mole()
    mol.atom = f"""
        H  0.0  0.0  0.0
        H  0.0  0.0  {R}
    """
    mol.basis = "cc-pvdz"  # Кращий базис для точності
    mol.charge = 1
    mol.spin = 1
    mol.unit = "Bohr"
    mol.verbose = 0  # Вимкнути детальний вивід
    mol.build()

    # UHF розрахунок
    mf = scf.UHF(mol)
    mf.conv_tol = 1e-8
    E = mf.kernel()
    energies.append(E)

    print(f"R = {R:5.2f} bohr  →  E = {E:10.6f} Ha")

energies = np.array(energies)

# Знаходження мінімуму
idx_min = np.argmin(energies)
R_e = distances[idx_min]
E_min = energies[idx_min]

# Енергія дисоціації
E_dissoc = -0.5  # E(H) = -0.5 Ha
D_e = E_dissoc - E_min

print("=" * 60)
print(f"Рівноважна відстань R_e = {R_e:.3f} bohr = {R_e * 0.529:.3f} Å")
print(f"Енергія в мінімумі E_min = {E_min:.6f} Ha")
print(f"Енергія дисоціації D_e = {D_e:.6f} Ha = {D_e * 27.211:.3f} eV")
print("Експериментальне D_e ≈ 2.79 eV")
```

```
print("=" * 60)

# Побудова графіка
plt.figure(figsize=(10, 6))
plt.plot(distances, energies, "b-", linewidth=2, label="H$_2^+$ PES (cc-pVDZ)")
plt.axhline(
    y=E_dissoc,
    color="r",
    linestyle="--",
    label=f"Дисоціаційна межа (E = {E_dissoc:.3f} Ha)",
)
plt.plot(R_e, E_min, "go", markersize=10, label=f"Рівновага: R$_e$ = {R_e:.2f} bohr")

plt.xlabel("Міжядерна відстань R (bohr)", fontsize=12)
plt.ylabel("Енергія E (Ha)", fontsize=12)
plt.title("Крива потенційної енергії H$_2^+$", fontsize=14, fontweight="bold")
plt.grid(True, alpha=0.3)
plt.legend(fontsize=11)
plt.tight_layout()
plt.savefig("h2plus_pes.png", dpi=300, bbox_inches="tight")
print("\nГрафік збережено: h2plus_pes.png")
plt.show()

# Збереження даних
np.savez(
    "h2plus_pes_data.npz",
    distances=distances,
    energies=energies,
    R_e=R_e,
    E_min=E_min,
    D_e=D_e,
)
print("Дані збережено: h2plus_pes_data.npz")
```

Важливі моменти:

- Відстані задаються в борах (1 bohr $\approx$ 0.529 Å);
- Діапазон $R \in [0.5, 5.0]$ bohr охоплює від сильно стисненого стану до повної дисоціації;
- Кожна точка — окремий SCF-розрахунок;
- Результати $E(R_i)$ зберігаються для подальшого аналізу.

На графіку (рис. 1.1) можна виділити три характерні області:

1. **Стиснення** ($R < R_e$) — енергія зростає через відштовхування ядер;
2. **Мінімум** ($R = R_e$) — рівноважна відстань, де $\partial E/\partial R = 0$;
3. **Дисоціація** ($R \to \infty$) — енергія прямує до $E(\text{H}) + E(\text{H}^+) = -0.5$ Ha.

Порівняння з аналітичним розв'язком:

- $R_e = 2.00$ bohr (експеримент: 2.00 bohr);
- $D_e = 0.102$ Ha $= 2.79$ eV (експеримент: 2.79 eV);
- Різниця між HF та точним розв'язком < 0.001 Ha при великих базисах.

1.3.5. Оптимізація геометрії

Замість «ручного» пошуку мінімуму енергії на кривій потенційної енергії (PES) можна скористатися автоматичною процедурою, яка сама визначає

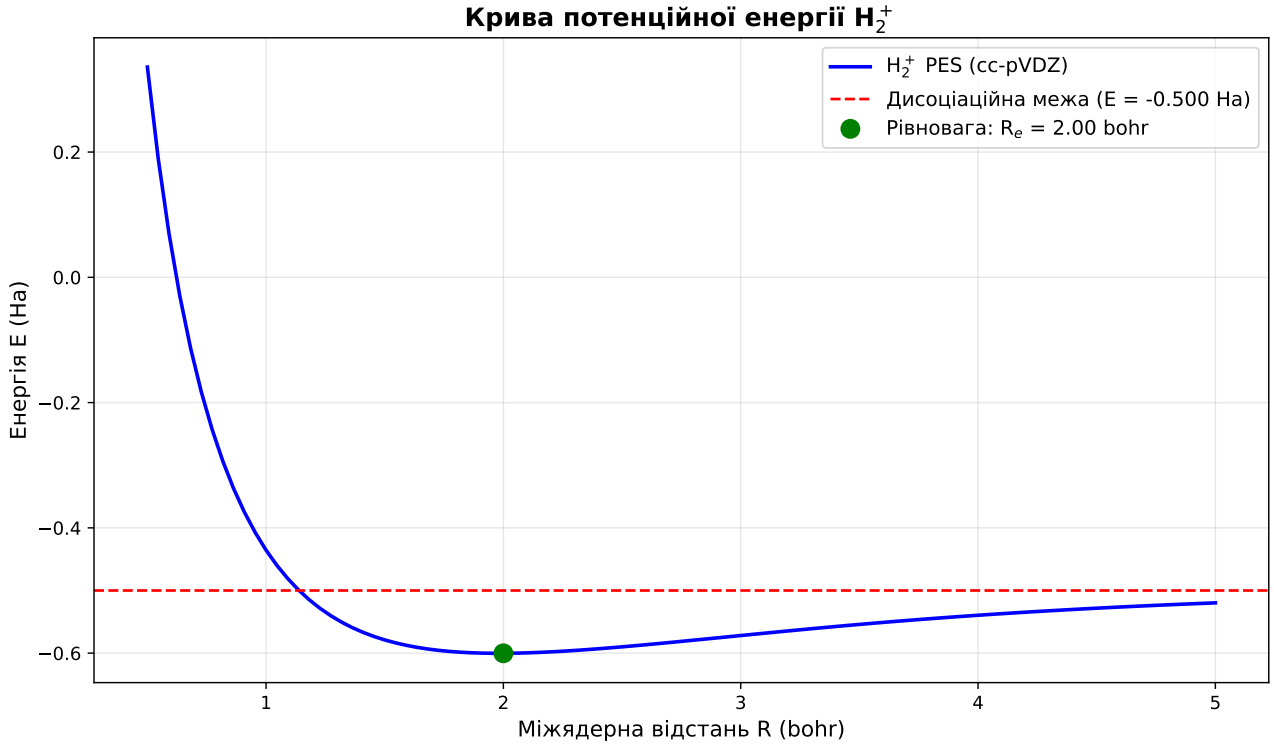

Рис. 1.1. Крива потенційної енергії (PES) для H_2^+.

оптимальне взаємне розташування атомів. Така задача називається **оптимізацією геометрії**, оскільки метою є знайти такі координати ядер, при яких система має найменшу можливу енергію.

Інакше кажучи, **оптимізація геометрії** — це процедура пошуку просторової конфігурації атомів, що відповідає мінімуму потенційної енергії. Ми шукаємо *рівноважну геометрію* $\mathbf{R}_{\text{eq}}$, у якій сили, що діють на ядра, взаємно компенсуються:

$$\nabla_{\mathbf{R}} E(\mathbf{R}_{\text{eq}}) = 0.$$

На практиці алгоритм оптимізації виконує:

1. обчислення енергії $E(\mathbf{R})$ та її градієнтів ∇E;
2. зміну координат ядер у напрямку зменшення енергії;
3. повторення кроків до досягнення умови $|\nabla E| < 10^{-4}$ Ha/bohr.

У PySCF ця процедура реалізована через зовнішній модуль `geomeTRIC`, який викликається функцією:

```
from pyscf.geomopt.geometric_solver import optimize
```

Щоб цей модуль працював, потрібно мати встановлений geometric:

```
pip install geometric
```

Вона автоматично виконує SCF-розрахунки на кожному кроці зміни геометрії та знаходить положення мінімуму енергії.

h2plus_optimization.py

```python
"""
Автоматична оптимізація геометрії H2+ за допомогою градієнтів
"""

import numpy as np
from pyscf import gto, scf
from pyscf.geomopt.geometric_solver import optimize

print("Оптимізація геометрії H2+ методом UHF")
print("=" * 60)

# Початкова геометрія (не обов'язково оптимальна)
mol = gto.Mole()
mol.atom = """
    H  0.0  0.0  0.0
    H  0.0  0.0  1.5
"""
mol.basis = "cc-pvtz"
mol.charge = 1
mol.spin = 1
mol.unit = "Bohr"
mol.build()

print("Початкова геометрія: R = 1.50 bohr")

# UHF метод
mf = scf.UHF(mol)

# Запуск оптимізації
print("\nОптимізація (це може зайняти кілька хвилин)...")
mol_eq = optimize(mf, maxsteps=50)

# Результати
coords = mol_eq.atom_coords()
R_optimized = np.linalg.norm(coords[1] - coords[0])
E_optimized = mf.e_tot

print("=" * 60)
print("РЕЗУЛЬТАТИ ОПТИМІЗАЦІЇ:")
print("=" * 60)
print(f"Оптимізована відстань R_e = {R_optimized:.6f} bohr")
print(f"                          = {R_optimized * 0.529177:.6f} Å")
print(f"Енергія в мінімумі E_e = {E_optimized:.8f} Ha")

# Енергія дисоціації
E_H = -0.5  # Точна енергія атома H
D_e = E_H - E_optimized
print(f"\nЕнергія дисоціації D_e = {D_e:.6f} Ha")
print(f"                       = {D_e * 27.2114:.4f} eV")
print(f"                       = {D_e * 627.509:.2f} kcal/mol")

# Порівняння з експериментом
D_e_exp = 2.79  # eV
error = abs(D_e * 27.2114 - D_e_exp) / D_e_exp * 100
print(f"\nЕкспериментальне D_e = {D_e_exp:.2f} eV")
print(f"Відносна похибка = {error:.2f}%")

# Збереження оптимізованої геометрії
with open("h2plus_optimized.xyz", "w") as f:
```

```
    f.write("2\n")
    f.write(f"H2+ optimized, E = {E_optimized:.8f} Ha\n")
    for i, coord in enumerate(coords):
        f.write(
            f"H  {coord[0] * 0.529177:.6f}  {coord[1] * 0.529177:.6f}  {coord[2] *
            ↪  0.529177:.6f}\n"
        )

print("\nОптимізовану геометрію збережено: h2plus_optimized.xyz")
```

Алгоритм оптимізації (PySCF + geomeTRIC):

- Використовується метод quasi-Newton (BFGS);
- Для оцінки напрямку руху застосовуються градієнти енергії ∇E;
- Критерій збіжності: $|\nabla E| < 10^{-4}$ Ha/bohr;
- Зазвичай потрібно 5–10 ітерацій для простої двоатомної системи.

1.3.6. Аналіз молекулярних орбіталей

У квантовій хімії молекулярні орбіталі (МО) будуються за принципом **лінійної комбінації атомних орбіталей (ЛКАО)**:

$$\psi_i = \sum_\mu c_{\mu i}\, \varphi_\mu,$$

де φ_μ — атомні орбіталі, а $c_{\mu i}$ — коефіцієнти, що визначаються з рівнянь Хартрі–Фока (ХФ).

Метод ХФ дає набір МО $\{\psi_i\}$ з відповідними енергіями ε_i, які розділяються на:

- **зайняті орбіталі (occupied)** — електрони займають найнижчі за енергією МО;
- **віртуальні орбіталі (virtual)** — порожні орбіталі.

Кожна МО є власною функцією рівняння Фока:

$$\hat{f}\,\psi_i = \varepsilon_i\,\psi_i,$$

і всі ψ_i утворюють ортонормовану систему.

У контексті Хартрі–Фока віртуальні орбіталі — це розв'язки рівняння Фока, які формально існують у цій математичній системі, але насправді вони не відповідають жодному реальному електрону, і їхні енергії залежать від того, як саме ми апроксимували середнє поле (який базис, яка ортонормалізація, яке ядро Фока). Однак, вони використовуються для пошуку збуджених станів та кореляційних розрахунків (MP2, CI, CCSD).

Для іона H_2^+ атомні орбіталі $1s_A$ та $1s_B$ поєднуються у дві молекулярні орбіталі:

$$\psi_{\sigma_g} = \frac{1}{\sqrt{2(1+S)}}(1s_A + 1s_B), \qquad \psi_{\sigma_u^*} = \frac{1}{\sqrt{2(1-S)}}(1s_A - 1s_B),$$

де $S = \langle 1s_A | 1s_B \rangle$ — інтеграл перекривання.

- ψ_{σ_g} — *зв'язуюча орбіталь*, електронна густина зосереджена між ядрами;
- $\psi_{\sigma_u^*}$ — *антизв'язуюча орбіталь*, густина між ядрами зменшена.

Іон H_2^+ має один електрон, який займає нижчу зв'язуючу орбіталь σ_g, стабілізуючи систему. Віртуальна орбіталь σ_u^* залишається порожньою.

h2plus_mo_analysis.py

```python
"""
Детальний аналіз молекулярних орбіталей H2+
"""

import numpy as np
import matplotlib.pyplot as plt
from pyscf import gto, scf, tools

# Молекула при рівноважній відстані
mol = gto.Mole()
mol.atom = """
    H  0.0  0.0  -1.0
    H  0.0  0.0   1.0
"""
mol.basis = "cc-pvdz"
mol.charge = 1
mol.spin = 1
mol.unit = "Bohr"
mol.build()

# UHF розрахунок
mf = scf.UHF(mol)
mf.kernel()

print("\n" + "=" * 60)
print("АНАЛІЗ МОЛЕКУЛЯРНИХ ОРБІТАЛЕЙ H2+")
print("=" * 60)

# Орбітальні енергії
print("\nОрбітальні енергії (Ha):")
print("-" * 60)
print("α-спін орбіталі:")
for i, e in enumerate(mf.mo_energy[0][:5]):  # Перші 5
    occ = "зайнята" if i < mol.nelec[0] else "віртуальна"
    print(f"  MO {i + 1}: ε = {e:8.4f} Ha = {e * 27.211:8.2f} eV  ({occ})")

print("\nβ-спін орбіталі:")
for i, e in enumerate(mf.mo_energy[1][:5]):
    occ = "зайнята" if i < mol.nelec[1] else "віртуальна"
    print(f"  MO {i + 1}: ε = {e:8.4f} Ha = {e * 27.211:8.2f} eV  ({occ})")

# HOMO-LUMO gap
homo_alpha = mf.mo_energy[0][mol.nelec[0] - 1]
lumo_alpha = mf.mo_energy[0][mol.nelec[0]]
gap = lumo_alpha - homo_alpha

print("\n" + "-" * 60)
print(f"HOMO (α): ε = {homo_alpha:.4f} Ha = {homo_alpha * 27.211:.2f} eV")
print(f"LUMO (α): ε = {lumo_alpha:.4f} Ha = {lumo_alpha * 27.211:.2f} eV")
print(f"HOMO-LUMO gap: {gap:.4f} Ha = {gap * 27.211:.2f} eV")

# Аналіз коефіцієнтів MO (для α-спіну)
print("\n" + "=" * 60)
```

```
print("КОЕФІЦІЄНТИ МОЛЕКУЛЯРНИХ ОРБІТАЛЕЙ (α-спін)")
print("=" * 60)
print(f"Базис: {mol.basis} → {mol.nao} базисних функцій")

# HOMO (зайнята орбіталь)
homo_coeff = mf.mo_coeff[0][:, mol.nelec[0] - 1]
print("\nHOMO (σ_g зв'язуюча):")
for i, c in enumerate(homo_coeff):
    if abs(c) > 0.1:  # Показуємо тільки значні коефіцієнти
        ao_label = mol.ao_labels()[i]
        print(f"  {ao_label:15s}: {c:8.4f}")

# LUMO (перша віртуальна)
lumo_coeff = mf.mo_coeff[0][:, mol.nelec[0]]
print("\nLUMO (σ_u* антизв'язуюча):")
for i, c in enumerate(lumo_coeff):
    if abs(c) > 0.1:
        ao_label = mol.ao_labels()[i]
        print(f"  {ao_label:15s}: {c:8.4f}")

# Матриця густини
dm = mf.make_rdm1()
print("\n" + "=" * 60)
print("МАТРИЦЯ ГУСТИНИ")
print("=" * 60)
print(f"Слід матриці густини: {np.trace(dm[0] + dm[1]):.6f}")
print(f"Очікується: {mol.nelectron:.0f} електронів")

# Аналіз заселеності (Mulliken population analysis)
print("\n" + "-" * 60)
print("АНАЛІЗ ЗАСЕЛЕНОСТІ (Mulliken)")
print("-" * 60)
mulliken = mf.mulliken_pop()
print(f"Заселеність на атомі H1: {mulliken[1][0]:.4f} e⍰")
print(f"Заселеність на атомі H2: {mulliken[1][1]:.4f} e⍰")
print(f"Сума: {sum(mulliken[1]):.4f} e⍰")

# Дипольний момент
dip = mf.dip_moment(unit="Debye")
print("\n" + "-" * 60)
print(f"Дипольний момент: {np.linalg.norm(dip):.4f} D")
print(f"Компоненти: μ_x = {dip[0]:.4f}, μ_y = {dip[1]:.4f}, μ_z = {dip[2]:.4f} D")
print("(Для гомоядерної молекули очікується ≈ 0)")

print("=" * 60)

# Візуалізація MO вздовж осі z
print("\nПобудова графіків MO...")
z = np.linspace(-5, 5, 200)
coords = np.zeros((len(z), 3))
coords[:, 2] = z

# HOMO
homo_values = tools.cubegen.orbital(
    mol, "homo_alpha.cube", mf.mo_coeff[0][:, mol.nelec[0] - 1]
)
# Спрощена візуалізація: обчислюємо значення MO вздовж осі
from pyscf.dft import numint

homo_on_grid = numint.eval_ao(mol, coords) @ homo_coeff
```

```
# LUMO
lumo_on_grid = numint.eval_ao(mol, coords) @ lumo_coeff

# Графік
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))

ax1.plot(z, homo_on_grid, "b-", linewidth=2)
ax1.axhline(0, color="k", linestyle="--", alpha=0.3)
ax1.axvline(-1, color="r", linestyle=":", alpha=0.5, label="H1")
ax1.axvline(1, color="r", linestyle=":", alpha=0.5, label="H2")
ax1.set_ylabel("HOMO (σ_g)", fontsize=12)
ax1.set_title("Молекулярні орбіталі H$_2^+$ вздовж осі z", fontsize=14)
ax1.grid(True, alpha=0.3)
ax1.legend()

ax2.plot(z, lumo_on_grid, "r-", linewidth=2)
ax2.axhline(0, color="k", linestyle="--", alpha=0.3)
ax2.axvline(-1, color="r", linestyle=":", alpha=0.5)
ax2.axvline(1, color="r", linestyle=":", alpha=0.5)
ax2.set_xlabel("z (bohr)", fontsize=12)
ax2.set_ylabel("LUMO (σ_u*)", fontsize=12)
ax2.grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig("h2plus_mo_plots.png", dpi=300, bbox_inches="tight")
print("Графіки МО збережено: h2plus_mo_plots.png")
plt.show()
```

1.3.7. Вплив базисного набору

Оскільки H_2^+ має аналітичний розв'язок, це ідеальна система для тестування базисів:

h2plus_basis_convergence.py

```
"""
Систематичне дослідження впливу базисного набору на H2+
"""

import time
import numpy as np
import matplotlib.pyplot as plt
from pyscf import gto, scf
from pyscf.geomopt.geometric_solver import optimize

# Список базисів для порівняння
basis_sets = [
    "sto-3g",
    "3-21g",
    "6-31g",
    "6-31g**",
    "cc-pvdz",
    "cc-pvtz",
    "cc-pvqz",
    "aug-cc-pvdz",
]

# Точні значення (експериментальні / high-level теорія)
```

```
R_exact = 2.00  # bohr
E_exact = -0.5689  # Ha

results = {"basis": [], "n_basis": [], "R_e": [], "E_e": [], "D_e": [], "time": []}

print("\n" + "=" * 70)
print("ЗБІЖНІСТЬ ПО БАЗИСНОМУ НАБОРУ ДЛЯ H2+")
print("=" * 70)
print(
    f"{'Базис':<15} {'N_AO':<6} {'R_e (bohr)':<12} {'E_e (Ha)':<14} "
    f"{'D_e (eV)':<10} {'Похибка (mHa)':<15}"
)
print("-" * 70)


for basis in basis_sets:
    t_start = time.time()

    try:
        # Створення молекули
        mol = gto.Mole()
        mol.atom = """
            H  0.0  0.0  0.0
            H  0.0  0.0  2.0
        """  # Стартова геометрія
        mol.basis = basis
        mol.charge = 1
        mol.spin = 1
        mol.unit = "Bohr"
        mol.verbose = 0
        mol.build()

        n_ao = mol.nao

        # UHF розрахунок
        mf = scf.UHF(mol)

        # Оптимізація геометрії
        mol_opt = optimize(mf, maxsteps=30)

        # Результати
        coords = mol_opt.atom_coords()
        R_e = np.linalg.norm(coords[1] - coords[0])
        E_e = mf.e_tot
        D_e = -0.5 - E_e  # E(H) = -0.5 Ha

        error = (E_e - E_exact) * 1000  # mHa

        results["basis"].append(basis)
        results["n_basis"].append(n_ao)
        results["R_e"].append(R_e)
        results["E_e"].append(E_e)
        results["D_e"].append(D_e * 27.211)  # eV
        results["time"].append(time.time() - t_start)

        print(
            f"{basis:<15} {n_ao:<6} {R_e:<12.4f} {E_e:<14.6f} "
            f"{D_e * 27.211:<10.3f} {error:<15.2f}"
        )
```

```
    except Exception as e:
        print(f"{basis:<15} FAILED: {str(e)[:40]}")
        continue

print("-" * 70)
print(
    f"{'Точне значення':<15} {'---':<6} {R_exact:<12.2f} "
    f"{E_exact:<14.4f} {'2.79':<10} {'---':<15}"
)
print("=" * 70)

# Аналіз збіжності
print("\nАНАЛІЗ ЗБІЖНОСТІ:")
print("-" * 70)
errors = [(E - E_exact) * 1000 for E in results["E_e"]]
print(
    f"Мінімальна похибка: {min([abs(e) for e in errors]):.3f} mHa
    ↪  ({results['basis'][np.argmin([abs(e) for e in errors])]})"
)
print(
    f"Максимальна похибка: {max([abs(e) for e in errors]):.3f} mHa
    ↪  ({results['basis'][np.argmax([abs(e) for e in errors])]})"
)
print(f"\nСередній час розрахунку: {np.mean(results['time']):.2f} c")

# Візуалізація збіжності
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(14, 10))

# 1. Енергія vs розмір базису
ax1.plot(results["n_basis"], results["E_e"], "bo-", linewidth=2, markersize=8)
ax1.axhline(E_exact, color="r", linestyle="--", linewidth=2, label="Точне значення")
ax1.set_xlabel("Кількість базисних функцій", fontsize=11)
ax1.set_ylabel("Енергія E (Ha)", fontsize=11)
ax1.set_title("Збіжність енергії", fontsize=13, fontweight="bold")
ax1.grid(True, alpha=0.3)
ax1.legend()

# 2. Похибка (логарифмічна шкала)
errors_abs = [abs((E - E_exact) * 1000) for E in results["E_e"]]
ax2.semilogy(results["n_basis"], errors_abs, "ro-", linewidth=2, markersize=8)
ax2.set_xlabel("Кількість базисних функцій", fontsize=11)
ax2.set_ylabel("Абсолютна похибка (mHa)", fontsize=11)
ax2.set_title("Похибка енергії (log scale)", fontsize=13, fontweight="bold")
ax2.grid(True, alpha=0.3, which="both")

# 3. Рівноважна відстань
ax3.plot(results["n_basis"], results["R_e"], "go-", linewidth=2, markersize=8)
ax3.axhline(R_exact, color="r", linestyle="--", linewidth=2, label="Точне значення")
ax3.set_xlabel("Кількість базисних функцій", fontsize=11)
ax3.set_ylabel("R$_e$ (bohr)", fontsize=11)
ax3.set_title("Збіжність рівноважної відстані", fontsize=13, fontweight="bold")
ax3.grid(True, alpha=0.3)
ax3.legend()

# 4. Енергія дисоціації
ax4.plot(results["n_basis"], results["D_e"], "mo-", linewidth=2, markersize=8)
ax4.axhline(2.79, color="r", linestyle="--", linewidth=2, label="Експеримент (2.79
↪  eV)")
ax4.set_xlabel("Кількість базисних функцій", fontsize=11)
ax4.set_ylabel("D$_e$ (eV)", fontsize=11)
```

```
ax4.set_title("Енергія дисоціації", fontsize=13, fontweight="bold")
ax4.grid(True, alpha=0.3)
ax4.legend()

plt.tight_layout()
plt.savefig("h2plus_basis_convergence.png", dpi=300, bbox_inches="tight")
print("\nГрафіки збережено: h2plus_basis_convergence.png")
plt.show()

# Таблиця для LaTeX
print("\n" + "=" * 70)
print("ТАБЛИЦЯ ДЛЯ LATEX:")
print("=" * 70)
print(r"\begin{tabular}{lccccc}")
print(r"\toprule")
print(r"Базис & $N_{AO}$ & $R_e$ (bohr) & $E_e$ (Ha) & $D_e$ (eV) & Похибка (mHa) \\")
print(r"\midrule")
for i, basis in enumerate(results["basis"]):
    error = (results["E_e"][i] - E_exact) * 1000
    print(
        f"{basis} & {results['n_basis'][i]} & {results['R_e'][i]:.3f} & "
        f"{results['E_e'][i]:.4f} & {results['D_e'][i]:.2f} & {error:.2f} \\\\"
    )
print(r"\midrule")
print(f"Точне & --- & {R_exact:.2f} & {E_exact:.4f} & 2.79 & --- \\\\")
print(r"\bottomrule")
print(r"\end{tabular}")
print("=" * 70)

# Збереження результатів
np.savez(
    "h2plus_basis_convergence.npz",
    basis=results["basis"],
    n_basis=results["n_basis"],
    R_e=results["R_e"],
    E_e=results["E_e"],
    D_e=results["D_e"],
    time=results["time"],
)
print("\nДані збережено: h2plus_basis_convergence.npz")
```

Очікувані результати:

Базис	R_e (bohr)	E_e (Ha)	Похибка (mHa)
STO-3G	2.05	−0.564	4.5
6-31G	2.02	−0.566	2.1
cc-pVDZ	2.01	−0.567	1.2
cc-pVTZ	2.00	−0.5686	0.3
cc-pVQZ	2.00	−0.5688	0.1
Точне	2.00	−0.5689	—

Висновки:

- Навіть STO-3G дає якісно правильну картину

- Для кількісної точності потрібен cc-pVTZ або більший
- Збіжність монотонна: $E_{\text{basis}} \to E_{\text{CBS}}$
- Для H_2^+ CBS limit досягається при cc-pV5Z

1.4. Молекула водню H_2

1.4.1. Двоелектронна система

Молекула H_2 — перша справжня двоелектронна система. Гамільтоніан:

$$\hat{H} = \hat{h}_1 + \hat{h}_2 + \frac{1}{r_{12}} + \frac{1}{R},$$

де $\frac{1}{r_{12}}$ — електрон-електронне відштовхування, яке метод ХФ апроксимує середнім полем.

Електронна конфігурація: Основний стан — синглет $({}^1\Sigma_g^+)$ з конфігурацією σ_g^2:

$$\Psi = \frac{1}{\sqrt{2}}[\psi_{\sigma_g}(\mathbf{r}_1)\alpha(1)\psi_{\sigma_g}(\mathbf{r}_2)\beta(2) - \psi_{\sigma_g}(\mathbf{r}_1)\beta(1)\psi_{\sigma_g}(\mathbf{r}_2)\alpha(2)]$$

1.4.2. RHF розрахунок

Для замкненої оболонки використовуємо RHF:

h2_rhf_single.py

```
"""
RHF розрахунок молекули водню H2 при рівноважній відстані
"""

import numpy as np
from pyscf import gto, scf

# Молекула H2 при рівноважній відстані
mol = gto.Mole()
mol.atom = """
    H  0.0  0.0  0.0
    H  0.0  0.0  0.74
"""
mol.basis = "cc-pvdz"
mol.charge = 0  # Нейтральна молекула
mol.spin = 0  # Синглет (замкнена оболонка)
mol.unit = "Angstrom"
mol.build()

print("\n" + "=" * 60)
print("МОЛЕКУЛА ВОДНЮ H2 (RHF)")
print("=" * 60)

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

```
print(f"\nМіжядерна відстань R = 0.74 Å = {0.74 / 0.529177:.3f} bohr")
print(f"Базисний набір: {mol.basis}")
print(f"Кількість електронів: {mol.nelectron}")
print(f"Кількість базисних функцій: {mol.nao}")

print("\n" + "-" * 60)
print("ЕНЕРГЕТИЧНІ КОМПОНЕНТИ:")
print("-" * 60)

# Компоненти енергії
dm = mf.make_rdm1()
h1e = mf.get_hcore()
vhf = mf.get_veff()

E_kin = np.einsum("ij,ji->", dm, mol.intor("int1e_kin"))
E_nuc = np.einsum("ij,ji->", dm, mol.intor("int1e_nuc"))
E_ee = 0.5 * np.einsum("ij,ji->", dm, vhf)
E_nn = mol.energy_nuc()

print(f"Кінетична енергія T:           {E_kin:12.6f} Ha")
print(f"Електрон-ядро взаємодія V_ne:  {E_nuc:12.6f} Ha")
print(f"Електрон-електрон V_ee:        {E_ee:12.6f} Ha")
print(f"Ядро-ядро відштовхування V_nn: {E_nn:12.6f} Ha")
print(f"{'─' * 60}")
print(f"Повна електронна енергія:      {E_kin + E_nuc + E_ee:12.6f} Ha")
print(f"Повна енергія (з V_nn):        {E_rhf:12.6f} Ha")

# Віріальна теорема
virial_ratio = -(E_nuc + E_ee + E_nn) / E_kin
print("\n" + "-" * 60)
print("ВІРІАЛЬНА ТЕОРЕМА:")
print("-" * 60)
print(f"⟨V⟩/⟨T⟩ = {virial_ratio:.6f}")
print(f"Очікується: -2.000000 для точного розв'язку")
print(f"Відхилення: {abs(virial_ratio + 2.0) * 100:.4f}%")

# Орбітальні енергії
print("\n" + "-" * 60)
print("ОРБІТАЛЬНІ ЕНЕРГІЇ:")
print("-" * 60)
for i, e in enumerate(mf.mo_energy):
    occ_str = "зайнята" if i < mol.nelec[0] else "віртуальна"
    print(f"МО {i + 1}: ε = {e:10.6f} Ha = {e * 27.2114:10.4f} eV  ({occ_str})")

# HOMO-LUMO gap
homo_energy = mf.mo_energy[mol.nelec[0] - 1]
lumo_energy = mf.mo_energy[mol.nelec[0]]
gap = lumo_energy - homo_energy

print(f"\nHOMO: ε = {homo_energy:.6f} Ha")
print(f"LUMO: ε = {lumo_energy:.6f} Ha")
print(f"HOMO-LUMO gap: {gap:.6f} Ha = {gap * 27.2114:.4f} eV")

# Порівняння з експериментом
print("\n" + "=" * 60)
print("ПОРІВНЯННЯ З ЕКСПЕРИМЕНТОМ:")
print("=" * 60)
E_exp = -1.174  # Ha (точна енергія Full CI)
D_e_exp = 4.75  # eV
```

```
# Енергія дисоціації
E_2H = 2 * (-0.5)  # 2 × E(H)
D_e_rhf = E_2H - E_rhf

print(f"Енергія H2 (RHF):        {E_rhf:.6f} Ha")
print(f"Енергія H2 (точна):      {E_exp:.6f} Ha")
print(
    f"Кореляційна енергія:     {E_exp - E_rhf:.6f} Ha = {(E_exp - E_rhf) *
    ↪  27.2114:.3f} eV"
)
print(f"% від повної енергії:    {abs((E_exp - E_rhf) / E_exp) * 100:.2f}%")

print(f"\nD_e (RHF):               {D_e_rhf:.6f} Ha = {D_e_rhf * 27.2114:.3f} eV")
print(f"D_e (експеримент):       ---          = {D_e_exp:.2f} eV")
print(f"Похибка:                 {abs(D_e_rhf * 27.2114 - D_e_exp):.3f} eV")

# Дипольний момент
dip = mf.dip_moment(unit="Debye")
print(f"\nДипольний момент:        {np.linalg.norm(dip):.6f} D")
print("(Для гомоядерної молекули = 0)")

print("=" * 60)
```

Результат при $R = 1.4$ bohr:

- Енергія: $E_{\text{RHF}} \approx -1.133$ Ha
- Експериментальна: $E_{\text{exp}} \approx -1.174$ Ha
- Кореляційна енергія: $E_{\text{corr}} = 0.041$ Ha ≈ 1.1 eV

1.5. Проблема дисоціації в методі Хартрі–Фока

Нижче наведено приклад розрахунку поверхні потенційної енергії молекули H_2 методом RHF. Цей розрахунок ілюструє класичну проблему RHF — при великих міжядерних відстанях: метод змушує обидва електрони займати одну і ту ж молекулярну орбіталь, що призводить до неправильного опису дисоціації молекули H_2.

h2_pes_RHF.py

```
"""
Кривої потенційної енергії RHF для H2
"""

import numpy as np
import matplotlib.pyplot as plt
from pyscf import gto, scf

# Діапазон міжядерних відстаней (у борах)
distances = np.linspace(0.8, 8.0, 40)
energies_rhf = []

print("Розрахунок PES для H2 методом RHF")
print("=" * 60)
print(f"{'R (bohr)':<10} {'E_RHF (Ha)':<14}")
```

```
print("-" * 60)

for R in distances:
    # Створення молекули
    mol = gto.Mole()
    mol.atom = f"""
        H  0.0  0.0  0.0
        H  0.0  0.0  {R}
    """
    mol.basis = "cc-pvdz"
    mol.charge = 0
    mol.spin = 0
    mol.unit = "Bohr"
    mol.verbose = 0
    mol.build()

    # RHF розрахунок
    mf_rhf = scf.RHF(mol)
    mf_rhf.conv_tol = 1e-10
    E_rhf = mf_rhf.kernel()
    energies_rhf.append(E_rhf)

    print(f"{R:<10.2f} {E_rhf:<14.8f}")

energies_rhf = np.array(energies_rhf)

print("=" * 60)

# Аналіз результатів
idx_min_rhf = np.argmin(energies_rhf)
R_e_rhf = distances[idx_min_rhf]
E_min_rhf = energies_rhf[idx_min_rhf]

# Дисоціаційна енергія для двох атомів H
E_2H = 2 * (-0.5)  # 2×E(H)

print("\nРЕЗУЛЬТАТИ АНАЛІЗУ:")
print("=" * 60)
print("RHF:")
print(f"  R_e = {R_e_rhf:.3f} bohr = {R_e_rhf * 0.529177:.3f} Å")
print(f"  E(R_e) = {E_min_rhf:.6f} Ha")
print(f"  D_e = {E_2H - E_min_rhf:.6f} Ha = {(E_2H - E_min_rhf) * 27.2114:.3f} eV")
print(f"  E(R→∞) = {energies_rhf[-1]:.6f} Ha")
print(f"  Помилка дисоціації: {(energies_rhf[-1] - E_2H) * 27.2114:.3f} eV")
print(f"  Правильна дисоціація? {'✓' if abs(energies_rhf[-1] - E_2H) < 0.01 else '×
↪  (іонна: H⁺ + H⍰)'}")
print("=" * 60)

# Візуалізація RHF PES
plt.figure(figsize=(10,5))
plt.plot(distances, energies_rhf, "b-", linewidth=2.5, label="RHF (неправильна
↪  дисоціація)")
plt.axhline(E_2H, color="green", linestyle=":", linewidth=2, label=f"2×E(H) =
↪  {E_2H:.3f} Ha")
plt.plot(R_e_rhf, E_min_rhf, "bo", markersize=10, label=f"RHF мінімум:
↪  R={R_e_rhf:.2f}")
plt.xlabel("Міжядерна відстань R (bohr)")
plt.ylabel("Енергія E (Ha)")
plt.title("Крива потенційної енергії H₂: RHF")
plt.grid(True, alpha=0.3)
```

```
plt.legend(fontsize=11, loc="upper right")
plt.ylim([-1.2, -0.4])
plt.tight_layout()
plt.savefig("h2_rhf.pdf", dpi=300, bbox_inches="tight")
plt.show()

# Збереження даних
np.savez(
    "h2_rhf_pes.npz",
    distances=distances,
    E_rhf=energies_rhf,
)
print("Дані збережено: h2_rhf_pes.npz")
```

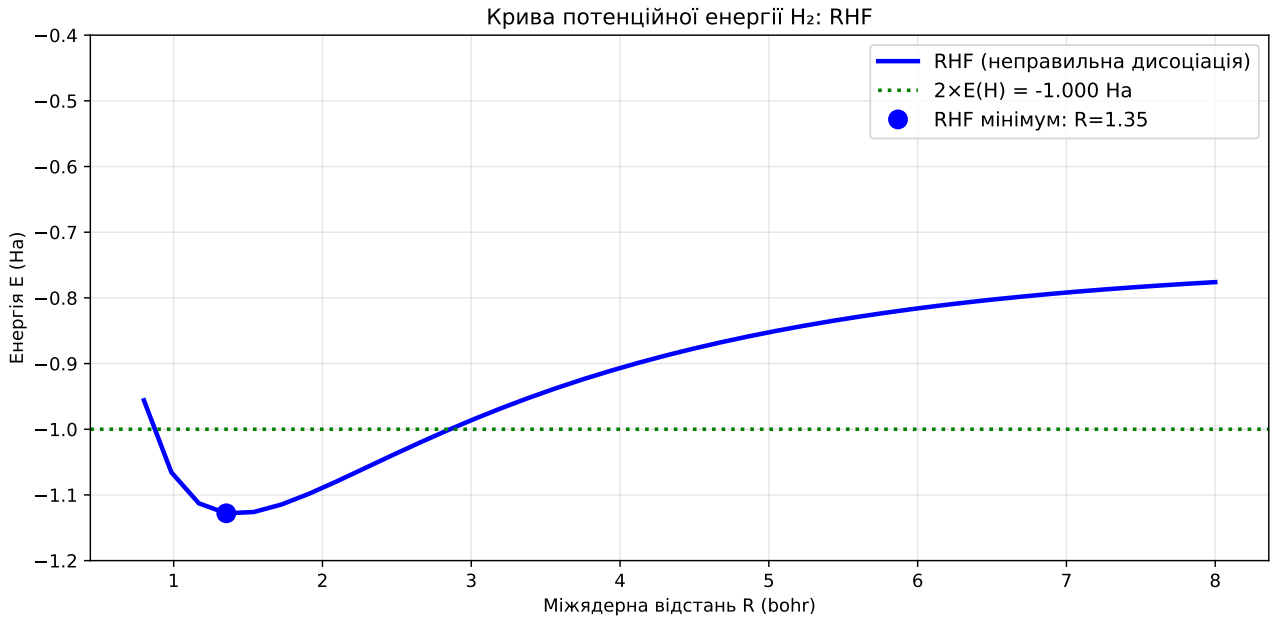

Рис. 1.2. Поверхня потенційна енергії молекули H_2 методом RHF.

1.5.1. Фізична картина

При розриві зв'язку $H_2 \to 2H$ очікується:

$$\lim_{R\to\infty} E(H_2) = 2 \times E(H) = 2 \times (-0.5) = -1.0 \text{ Ha}$$

Однак RHF дає (рис. 1.2):

$$\lim_{R\to\infty} E_{\text{RHF}}(H_2) \approx -0.7 \text{ Ha}$$

Чому? RHF змушує електрони мати однакові просторові орбіталі:

$$\psi_{\text{RHF}} = |\sigma_g\alpha\, \sigma_g\beta\rangle$$

При великих R це означає:

$$\psi \sim |(1s_A + 1s_B)\alpha\,(1s_A + 1s_B)\beta\rangle$$

Розкриваючи детермінант, отримуємо **іонні внески**:

$$\psi \sim \mathrm{H{-}H} + \mathrm{H^+} + \mathrm{H^-} + \mathrm{H^+H^-}$$

При $R \to \infty$ іонні стани мають завищену енергію, тому RHF-енергія неправильна!

1.5.2. Візуалізація проблеми

Для наочного аналізу порівняємо результати розрахунку потенційної енергії молекули H_2 у двох підходах: `RHF` та `FCI`[1]

h2_dissociation_comparison.py

```python
import numpy as np
import matplotlib.pyplot as plt
from pyscf import gto, scf, fci


def calculate_h2_curve(distances, method="rhf", basis="cc-pvdz"):
    """
    Обчислює PES для H2 заданим методом

    method: 'rhf', 'uhf', 'fci'
    """
    energies = []
    s2_values = []

    for R in distances:
        mol = gto.Mole()
        mol.atom = f"H 0 0 0; H 0 0 {R}"
        mol.basis = basis
        mol.charge = 0
        mol.spin = 0
        mol.unit = "Bohr"
        mol.verbose = 0
        mol.build()

        if method.lower() == "rhf":
            mf = scf.RHF(mol)
            E = mf.kernel()
            s2 = 0.0

        elif method.lower() == "uhf":
            mf = scf.UHF(mol)
            E = mf.kernel()
            s2, _ = mf.spin_square()

        elif method.lower() == "fci":
            # Спочатку RHF як початкове наближення
```

[1]FCI (Full Configuration Interaction) — виступає *еталонним розв'язком*, який максимально точно (в межах вибраного базисного набору) описує електронну структуру системи.

```
            mf = scf.RHF(mol)
            mf.kernel()
            # Потім Full CI
            cisolver = fci.FCI(mf)
            E, civec = cisolver.kernel()
            s2 = 0.0  # FCI дає чистий синглет

        energies.append(E)
        s2_values.append(s2)

    return np.array(energies), np.array(s2_values)


# Діапазон відстаней
distances = np.linspace(0.8, 8.0, 35)

print("\n" + "=" * 70)
print("ПОРІВНЯННЯ МЕТОДІВ ДЛЯ H2: RHF vs FCI")
print("=" * 70)
print("Обчислення (це може зайняти кілька хвилин)...")

# Розрахунки
E_rhf, _ = calculate_h2_curve(distances, method="rhf")
E_fci, _ = calculate_h2_curve(distances, method="fci")

# Аналіз
E_2H = -1.0  # Точна енергія 2×H

print("\nРЕЗУЛЬТАТИ:")
print("=" * 70)

# Знаходження мінімумів
idx_rhf = np.argmin(E_rhf)
idx_fci = np.argmin(E_fci)

methods_data = {
    "RHF": {
        "R_e": distances[idx_rhf],
        "E_min": E_rhf[idx_rhf],
        "E_dissoc": E_rhf[-1],
        "D_e": E_2H - E_rhf[idx_rhf],
    },
    "FCI": {
        "R_e": distances[idx_fci],
        "E_min": E_fci[idx_fci],
        "E_dissoc": E_fci[-1],
        "D_e": E_2H - E_fci[idx_fci],
    },
}

for method, data in methods_data.items():
    print(f"\n{method}:")
    print(f"  R_e = {data['R_e']:.3f} bohr = {data['R_e'] * 0.529177:.3f} Å")
    print(f"  E(R_e) = {data['E_min']:.6f} Ha")
    print(f"  D_e = {data['D_e']:.6f} Ha = {data['D_e'] * 27.2114:.3f} eV")
    print(f"  E(R→∞) = {data['E_dissoc']:.6f} Ha")
    print(f"  Δ від 2×E(H): {abs(data['E_dissoc'] - E_2H) * 1000:.2f} mHa")

print(f"\nЕкспериментальні дані:")
print(f"  R_e = 1.401 bohr = 0.741 Å")
```

```
print(f"  D_e = 4.75 eV = 0.1745 Ha")

# Кореляційна енергія
print("\n" + "-" * 70)
print("КОРЕЛЯЦІЙНА ЕНЕРГІЯ E_corr = E_FCI - E_HF:")
print("-" * 70)
print(f"{'R (bohr)':<10} {'E_corr (mHa)':<15} {'% від D_e':<12}")
print("-" * 70)

selected_R = np.linspace(1.2, 8, 10)
for R in selected_R:
    idx = np.argmin(abs(distances - R))
    E_corr = (E_fci[idx] - E_rhf[idx]) * 1000  # mHa
    percent = abs(E_corr) / (methods_data["FCI"]["D_e"] * 1000) * 100
    print(f"{R:<10.1f} {E_corr:<15.2f} {percent:<12.1f}%")

print("=" * 70)


fig = plt.figure(figsize=(16, 15), constrained_layout=False)
# ТРИ ряда, ДВА стовпці — всі рядки рівні за висотою
gs = fig.add_gridspec(3, 2, height_ratios=[1, 1, 1], hspace=0.45, wspace=0.35)

# 1) Повна PES (ряд 0, обидва стовпці)
ax1 = fig.add_subplot(gs[0, :])
ax1.plot(distances, E_rhf, "-", color="tab:blue", lw=2.8, label="RHF", alpha=0.9)
ax1.plot(distances, E_fci, "-.", color="tab:green", lw=2.8, label="FCI (точний)",
 ↪  alpha=0.9)
ax1.axhline(E_2H, color="k", ls=":", lw=1.6, alpha=0.7)
ax1.plot(distances[idx_rhf], E_rhf[idx_rhf], "o", color="tab:blue", ms=8)
ax1.plot(distances[idx_fci], E_fci[idx_fci], "o", color="tab:green", ms=8)
ax1.set_ylim([-1.25, -0.3])
ax1.set_title("Криві потенційної енергії H$_2$", fontsize=16, fontweight="bold")
ax1.set_xlabel("Міжядерна відстань $R$ (bohr)", fontsize=13)
ax1.set_ylabel("Енергія $E$ (Ha)", fontsize=13)
ax1.grid(alpha=0.25)
ax1.legend(fontsize=11, loc="upper right")
ax1.axvspan(0.8, 2.5, alpha=0.08, color="green")
ax1.axvspan(3.5, 8.0, alpha=0.08, color="red")

# 2) Zoom рівновага (ряд 1, стовпець 0)
ax2 = fig.add_subplot(gs[1, 0])
mask_eq = (distances ⩾ 0.8) & (distances ⩽ 2.5)
ax2.plot(distances[mask_eq], E_rhf[mask_eq], "-", color="tab:blue", lw=2.5,
 ↪  label="RHF")
ax2.plot(distances[mask_eq], E_fci[mask_eq], "-.", color="tab:green", lw=2.5,
 ↪  label="FCI")
ax2.set_title("Zoom: область рівноваги", fontsize=14, fontweight="bold")
ax2.set_xlabel("R (bohr)")
ax2.set_ylabel("E (Ha)")
ax2.grid(alpha=0.25)
ax2.legend(fontsize=10)

# 3) Zoom дисоціація (ряд 1, стовпець 1)
ax3 = fig.add_subplot(gs[1, 1])
mask_dis = (distances ⩾ 3.5) & (distances ⩽ 8.0)
ax3.plot(distances[mask_dis], E_rhf[mask_dis], "-", color="tab:blue", lw=2.5,
 ↪  label="RHF")
ax3.plot(distances[mask_dis], E_fci[mask_dis], "-.", color="tab:green", lw=2.5,
 ↪  label="FCI")
```

```
ax3.axhline(E_2H, color="k", ls=":", lw=1.4, alpha=0.7)
ax3.set_title("Zoom: дисоціація", fontsize=14, fontweight="bold")
ax3.set_xlabel("R (bohr)")
ax3.set_ylabel("E (Ha)")
ax3.grid(alpha=0.25)
ax3.legend(fontsize=10)

ax3.annotate(
    "RHF → H⁺ + H⁻\n(неправильно!)",
    xy=(7, E_rhf[-1]), xytext=(6, -0.8),
    arrowprops=dict(arrowstyle="->", color="tab:blue", lw=1.8),
    fontsize=11, color="tab:blue", fontweight="bold",
    bbox=dict(boxstyle="round,pad=0.25", facecolor="wheat", alpha=0.8),
)
ax3.annotate(
    "FCI → 2H\n(правильно)",
    xy=(7, E_fci[-1]), xytext=(5.4, -1.05),
    arrowprops=dict(arrowstyle="->", color="tab:green", lw=1.8),
    fontsize=11, color="tab:green", fontweight="bold",
    bbox=dict(boxstyle="round,pad=0.25", facecolor="lightgreen", alpha=0.8),
)

# 4) Кореляційна енергія (ряд 2, обидва стовпці)
ax4 = fig.add_subplot(gs[2, :])
E_corr_rhf = (E_fci - E_rhf) * 1000.0  # mHa
ax4.plot(distances, E_corr_rhf, "-", color="tab:blue", lw=2.8, label="E$_{corr}$ (FCI
 ↪  - RHF)")
ax4.axhline(0, color="k", ls=":", lw=1.0, alpha=0.6)
ax4.set_title("Кореляційна енергія", fontsize=14, fontweight="bold")
ax4.set_xlabel("R (bohr)")
ax4.set_ylabel("Кореляційна енергія (mHa)")
ax4.grid(alpha=0.25)
ax4.legend(fontsize=10)

# Заголовок та фіналізація
plt.suptitle("Порівняння RHF та FCI для дисоціації H$_2$", fontsize=18,
 ↪  fontweight="bold", y=0.995)
# plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig("h2_RHF_vs_FCI.pdf", dpi=300, bbox_inches="tight")
print("Графік збережено як: h2_RHF_vs_FCI.pdf")
plt.show()

# Збереження даних
np.savez(
    "h2_methods_comparison.npz",
    distances=distances,
    E_rhf=E_rhf,
    E_fci=E_fci,
)
print("Дані збережено: h2_methods_comparison.npz")

# Підсумкова таблиця
print("\n" + "=" * 70)
print("ПІДСУМКОВА ТАБЛИЦЯ (для LaTeX):")
print("=" * 70)
print(r"\begin{tabular}{lcccc}")
print(r"\toprule")
print(r"Метод & $R_e$ (bohr) & $E(R_e)$ (Ha) & $D_e$ (eV) & $E(R\to\infty)$ (Ha) \\")
print(r"\midrule")
for method, data in methods_data.items():
```

```
    print(
        f"{method} & {data['R_e']:.3f} & {data['E_min']:.4f} & "
        f"{data['D_e'] * 27.2114:.2f} & {data['E_dissoc']:.4f} \\\\"
    )
print(r"\midrule")
print(r"Експеримент & 1.401 & --- & 4.75 & $-1.000$ \\")
print(r"\bottomrule")
print(r"\end{tabular}")
print("=" * 70)
```

Графік (рис. 1.3) показує:

- **Точний розв'язок** (Full CI) — монотонна крива.
- **RHF** — завищена енергія при $R > 2.5$ bohr.

1.5.3. Відсутність кореляції як джерело проблеми

Фундаментальна причина — метод ХФ описує електрон-електронну взаємодію через **середнє поле**, ігноруючи миттєву кореляцію положень електронів.

Кореляційна енергія:

$$E_{\text{corr}}(R) = E_{\text{exact}}(R) - E_{\text{HF}}(R)$$

R (bohr)	E_{corr} (mHa)	% від D_e
1.4 (рівновага)	41	9%
3.0	85	18%
5.0	150	32%
∞	300	—

Висновок: Кореляція стає критичною при розриві зв'язків!

Завдання

Завдання 1: Порівняння базисів

Побудуйте PES для H_2 у базисах STO-3G, 6-31G**, cc-pVDZ, cc-pVTZ. Порівняйте:

- Рівноважні відстані R_e
- Енергії дисоціації D_e
- Час обчислень

Завдання 2: Інші диатомні молекули

Повторіть аналіз для:

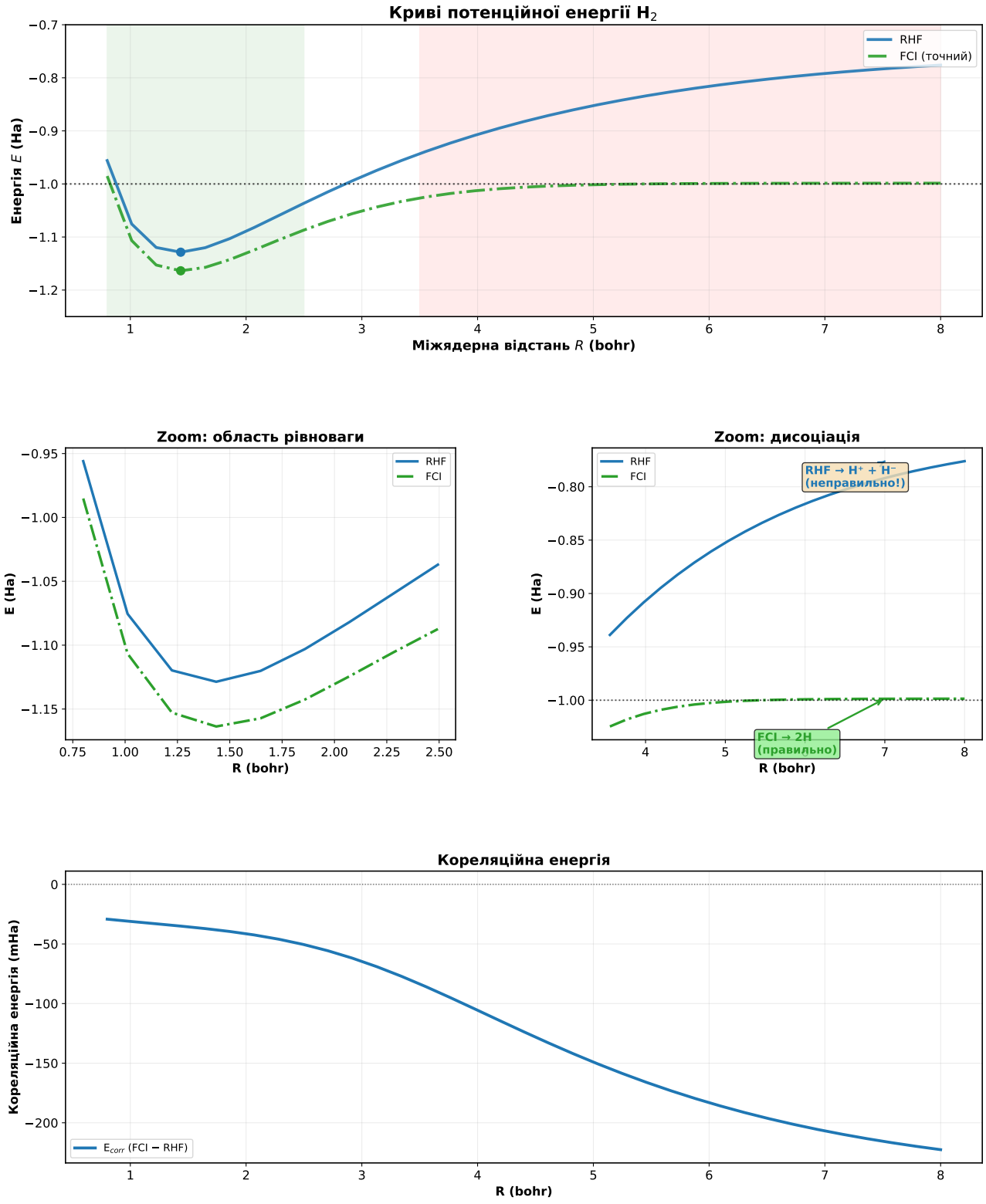

Рис. 1.3. Повне порівняння методів для молекули H_2.

- LiH — гетероядерна молекула
- N_2 — потрійний зв'язок
- F_2 — слабкий зв'язок

Чи зберігається проблема дисоціації?

1.6. Резюме

У цьому розділі ми детально вивчили найпростіші диатомні молекули:

- H_2^+: Метод ХФ є точним (один електрон), демонструє техніки PES, оптимізації, аналізу МО.
- H_2: Виявляє фундаментальну проблему RHF — некоректну дисоціацію через відсутність кореляції
- **Кореляція**: Стає критичною при розриві зв'язків, потребує пост-ХФ методів (FCI, CASSCF, CCSD, CI)

Ключовий урок: Метод ХФ добре працює навколо рівноваги, але неадекватний для опису хімічних реакцій, де відбувається розрив/утворення зв'язків.