

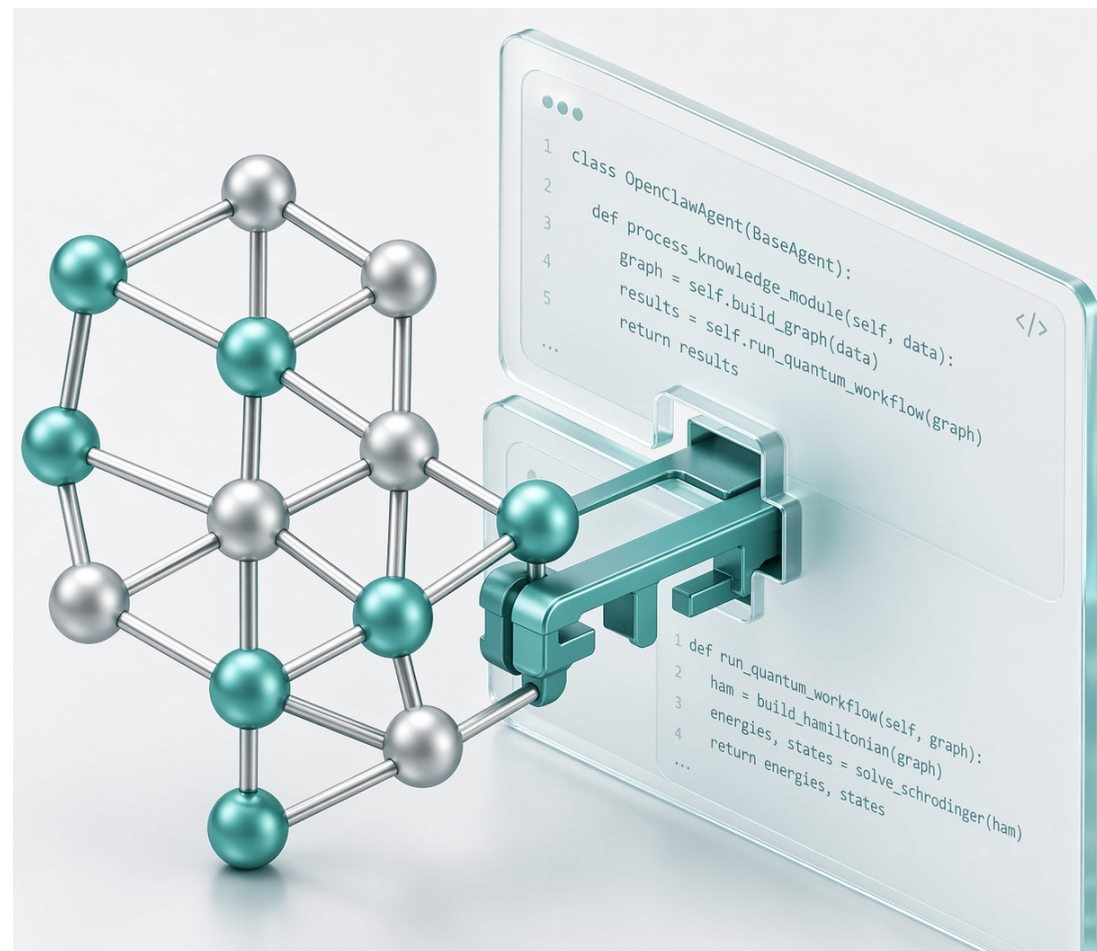


中国科学院大学 前沿交叉科学学院  
University of Chinese Academy of Sciences School of Advanced Interdisciplinary Sciences

# Full Differentiable TDDFT Code for Neural Network XC Functional and Automatic Quantum Chemistry with OpenClaw

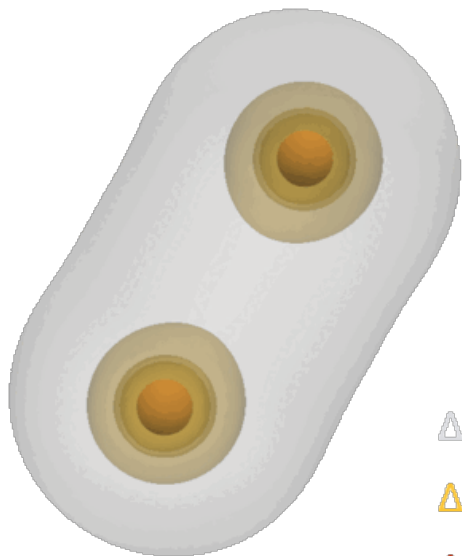
Yuan Jiao

School of Advanced Interdisciplinary Sciences, SAIS  
University of Chinese Academy of Sciences



- **Full Differentiable TDDFT Code for Neural Network XC Functional**
  - **DL-TDDFT – A Test for Accelerating TDDFT**
  - **Quantum Chemistry with OpenClaw, Hermes – Automatic Materials Calculation**
- Skills and Agents Team for Researching**

Molecule = Na<sub>2</sub>  
 $\Delta\rho(\mathbf{r}) = |\rho_{CCSD}(\mathbf{r}) - \rho_{\theta,DFT}(\mathbf{r})|$

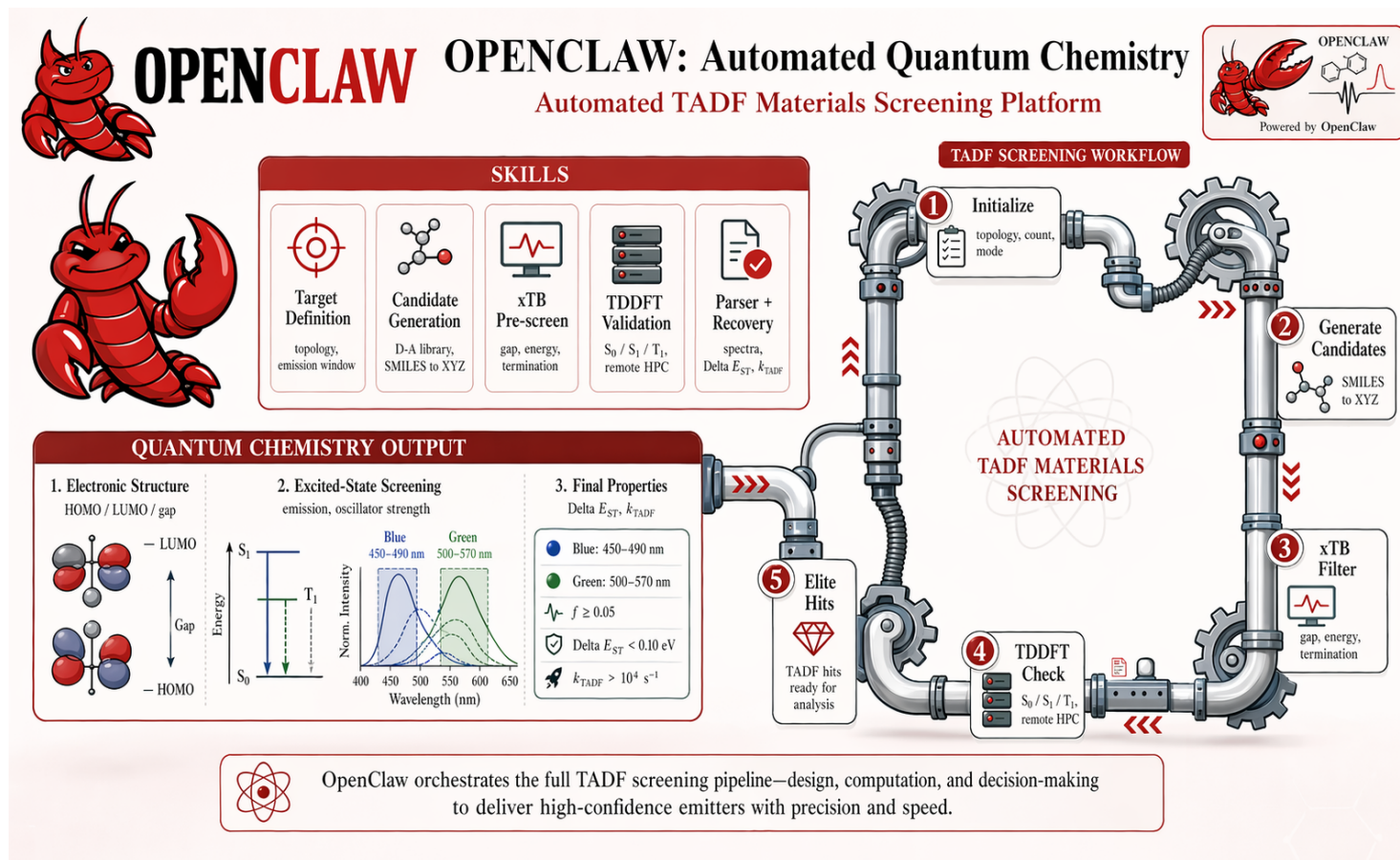


$$\Delta\rho(\mathbf{r}) = 5E-5$$

$$\Delta\rho(\mathbf{r}) = 3E-2$$

$$\Delta\rho(\mathbf{r}) = 6E-2$$

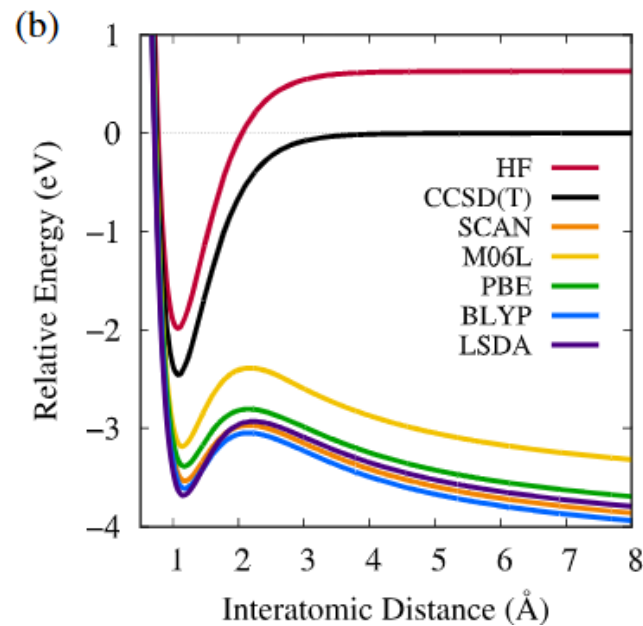
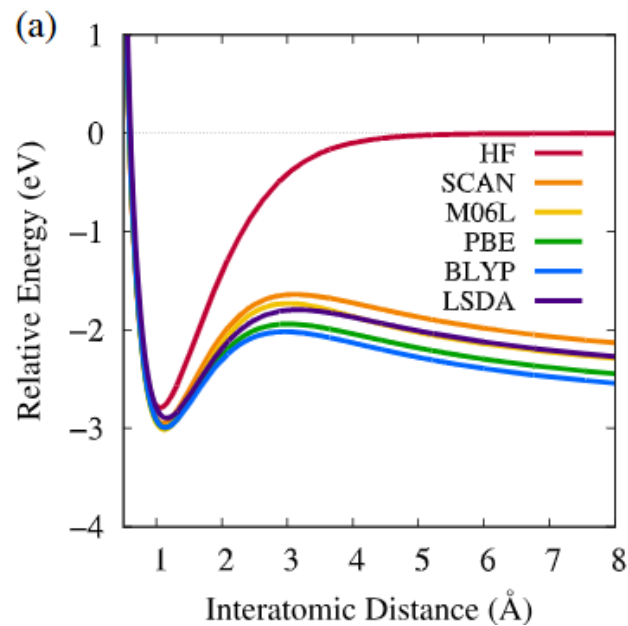
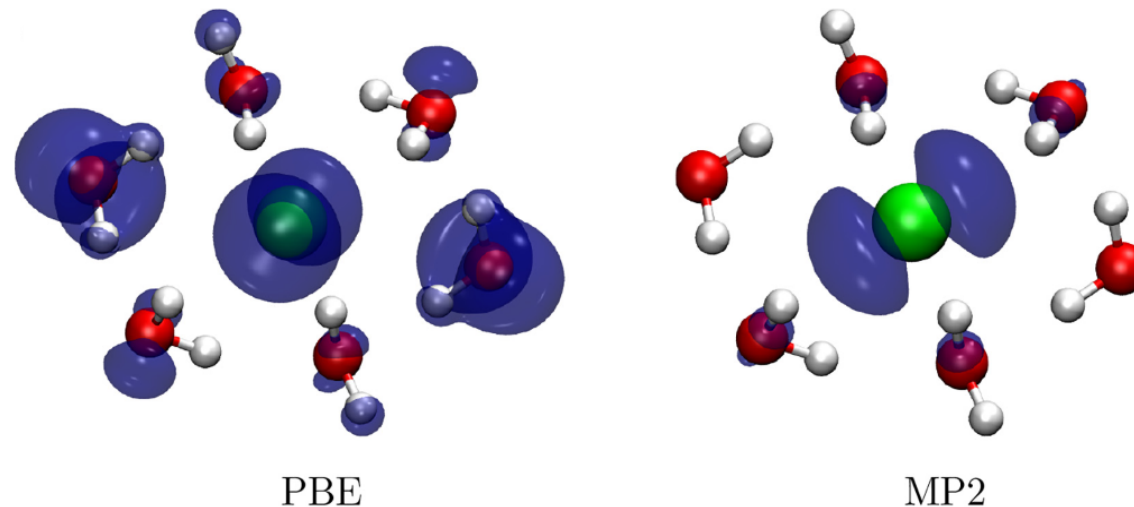
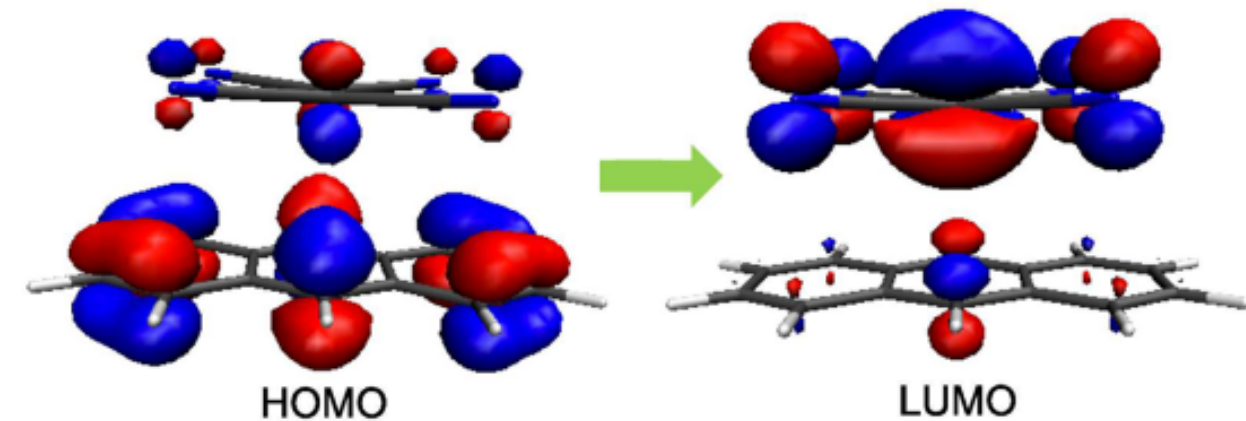
Epoch = 1  
 Mean  $\Delta\rho(r) = 1.926E-04$



# ■ Background – SIE and TDDFT

*J. Chem. Theory Comput.* 2019, 15, 8, 4305–4311

*WIREs Comput Mol Sci.* 2023;13(2):e1631



**N-electron system**

$$v_{eff}(\mathbf{r}, \mathbf{R}) = v_{ext}(\mathbf{R}) + \int d\mathbf{r}' \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} + v_{xc}(\mathbf{r})$$

**One-electron system ( $|\mathbf{r}| \rightarrow \infty$ )**

$$v_{eff}(\mathbf{r}, \mathbf{R}) = v_{ext}(\mathbf{R}) + \frac{1}{|\mathbf{r}|} + v_{xc}(\mathbf{r})$$

# ■ Background – SIE and TDDFT

$$v_{ext}(\mathbf{x}, \{\mathbf{R}\}) \Leftrightarrow \rho(\mathbf{x}) \Leftrightarrow \{\psi_i(\mathbf{x})\}$$

The Time-Dependent of XC Functional Vanished !

$$\left[ -\frac{1}{2} \nabla^2 + v_{ext}(\mathbf{x}, \{\mathbf{R}\}) + v_H(\mathbf{x}) + v_{xc}(\mathbf{x}) \right] \psi_i(\mathbf{x}) = \varepsilon_i \psi_i(\mathbf{x})$$

Runge-Gross Theorem

$$v_{ext}(\mathbf{x}, \{\mathbf{R}\}, t) + \mathbf{C}(t) \Leftrightarrow \rho(\mathbf{x}, t) \Leftrightarrow \{\psi_i(\mathbf{x}, t) e^{-i\alpha(t)}\}$$

$$\left[ -\frac{1}{2} \nabla^2 + v_{ext}(\mathbf{x}, \{\mathbf{R}\}, t) + v_H(\mathbf{x}, t) + v_{xc}(\mathbf{x}, t) \right] \psi_i(\mathbf{x}, t) = i \frac{\partial}{\partial t} \psi_i(\mathbf{x}, t)$$

Linear-Response – **Adiabatic Approximation**

$$F_{aa}^{(0)} x_{ai} - x_{ai} F_{ii}^{(0)} + \sum_{bj} \left( \frac{\partial F_{ai}}{\partial P_{bj}} x_{bj} + \frac{\partial F_{ai}}{\partial P_{jb}} y_{bj} \right) P_{ii}^{(0)} = \omega x_{ia}$$

$$F_{aa}^{(0)} y_{ai} - y_{ai} F_{ii}^{(0)} - \sum_{bj} P_{ii}^{(0)} \left( \frac{\partial F_{ia}}{\partial P_{bj}} x_{bj} + \frac{\partial F_{ia}}{\partial P_{jb}} y_{bj} \right) = \omega x_{ia}$$

$$F_{aa}^{(0)} = \int d\mathbf{r} \psi_a^*(\mathbf{x}, t) \left\{ -\frac{1}{2} \nabla^2 + v_{ext}(\mathbf{x}, \{\mathbf{R}\}, t) + v_H(\mathbf{x}, t) + v_{xc}(\mathbf{x}, t) \right\} \psi_a(\mathbf{x}, t)$$

$$F_{ia} = \int d\mathbf{r} \psi_i^*(\mathbf{x}, t) \left\{ -\frac{1}{2} \nabla^2 + v_{ext}(\mathbf{x}, \{\mathbf{R}\}, t) + v_H(\mathbf{x}, t) + v_{xc}(\mathbf{x}, t) \right\} \psi_a(\mathbf{x}, t) + g_{ai}(\omega) + \Delta F_{ia}^{(0)}$$

Casida-TDDFT

$$\begin{pmatrix} A & B \\ B^* & A^* \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} = \omega \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}$$

$$\psi_{ex} = \sum_{ai} x_{ai} \hat{a}_a^+ \hat{a}_i \psi_{gr}$$

$$\psi_{di} = \sum_{ai} y_{ai} \hat{a}_a^+ \hat{a}_i \psi_{gr}$$

TDA-TDDFT

$$AX = \omega X$$

$$\psi_{ex} = \sum_{ai} x_{ai} \hat{a}_a^+ \hat{a}_i \psi_{gr}$$

# ■ Background – SIE and TDDFT

## Casida-TDDFT

$$\begin{pmatrix} A & B \\ B^* & A^* \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} = \omega \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}$$

$$\psi_{ex} = \sum_{ai} x_{ai} \hat{a}_a^+ \hat{a}_i \psi_{gr}$$

$$\psi_{di} = \sum_{ai} y_{ai} \hat{a}_a^+ \hat{a}_i \psi_{gr}$$



## TDA-TDDFT

$$AX = \omega X$$

$$\psi_{ex} = \sum_{ai} x_{ai} \hat{a}_a^+ \hat{a}_i \psi_{gr}$$

## A, B Matrix and Response Kernel

$$A = \delta_{ij} \delta_{ab} (\epsilon_a - \epsilon_i) +$$

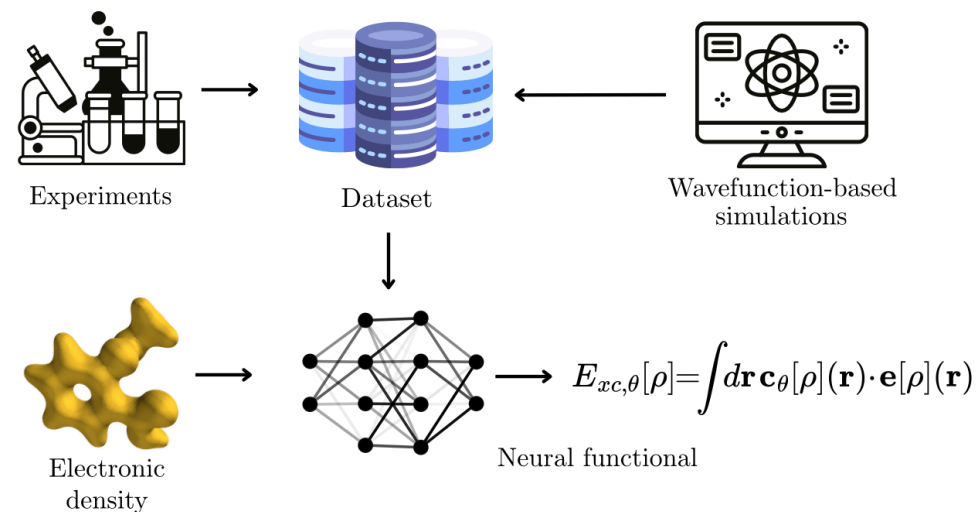
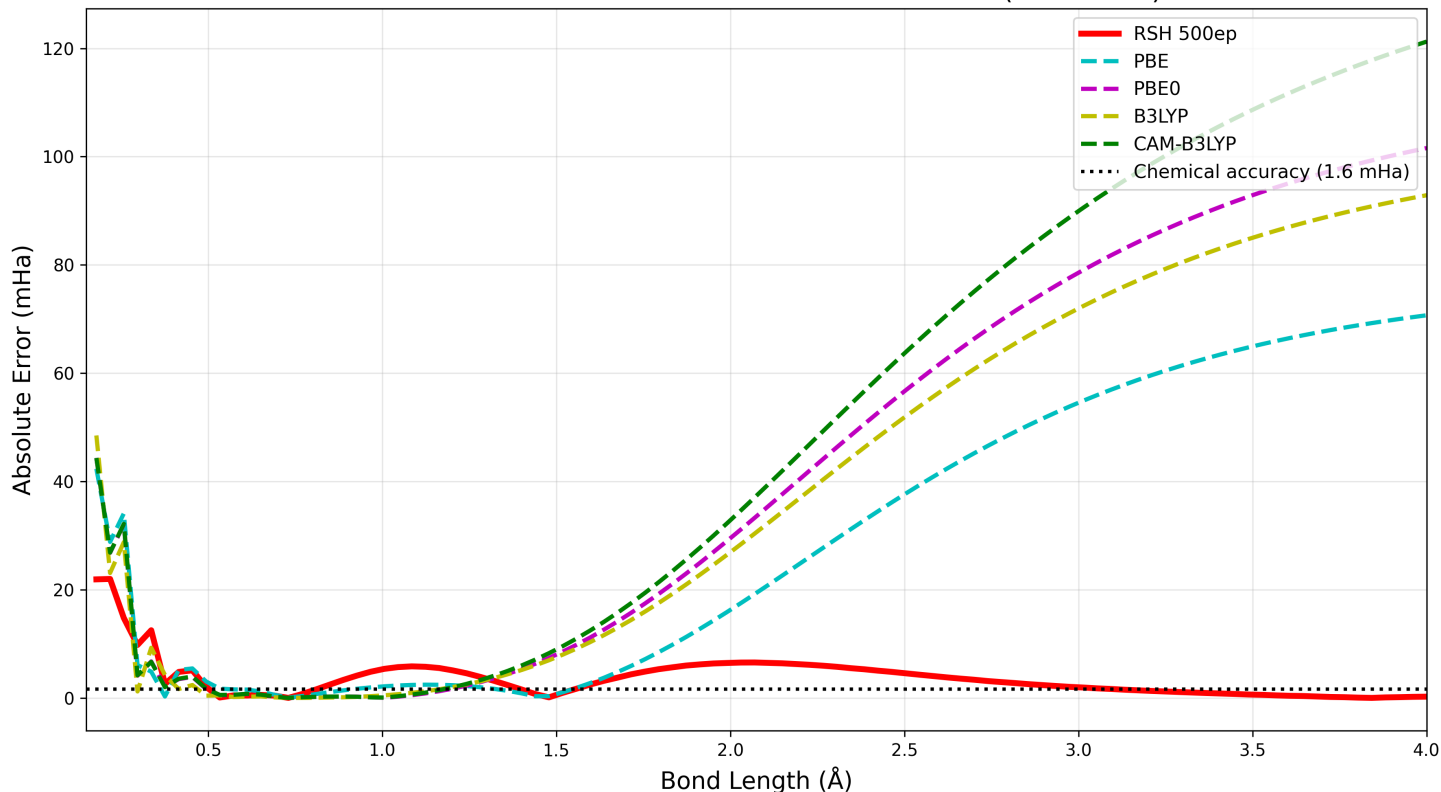
$$\int d\mathbf{r} \int d\mathbf{r}' \phi_i(\mathbf{r}) \phi_a^*(\mathbf{r}) \left\{ \frac{1}{|\mathbf{r} - \mathbf{r}'|} + f_{xc} \right\} \phi_b^*(\mathbf{r}') \phi_j(\mathbf{r}')$$

$$B = \int d\mathbf{r} \int d\mathbf{r}' \mathbf{r}' \phi_i(\mathbf{r}) \phi_a^*(\mathbf{r}) \left\{ \frac{1}{|\mathbf{r} - \mathbf{r}'|} + f_{xc} \right\} \phi_b^*(\mathbf{r}') \phi_j(\mathbf{r}')$$

$$f_{xc}[\rho] = \frac{\delta^2 E_{xc}[\rho]}{\delta \rho(\mathbf{r}) \delta \rho(\mathbf{r}')}$$

# ■ TD-GradDFT – GradDFT for Ground State

H2 Dissociation Curve - Functional Errors vs FCI ( $r \geq 0.15 \text{ \AA}$ )



J. Chem. Phys. 160, 062501 (2024)

## Coefficient Functions

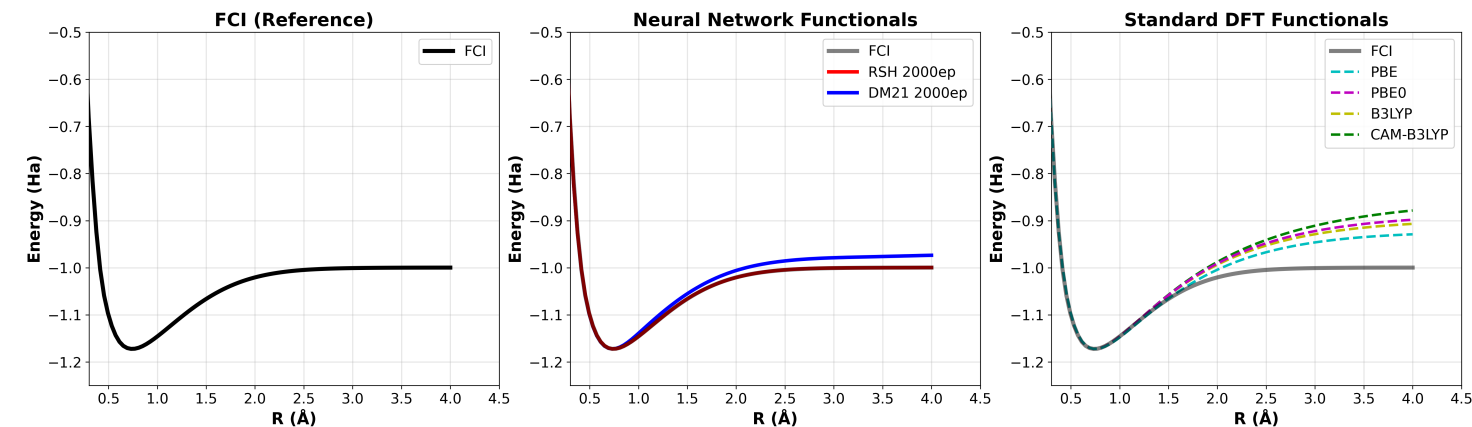
$$c_\theta[\rho](\mathbf{r}) = (c_{1,\theta}[\rho](\mathbf{r}), c_{2,\theta}[\rho](\mathbf{r}) \dots c_{k,\theta}[\rho](\mathbf{r}))$$

## Energy Densities

$$e[\rho](\mathbf{r}) = (e_1[\rho](\mathbf{r}), e_2[\rho](\mathbf{r}) \dots e_k[\rho](\mathbf{r}))$$

## Loss Function

$$L = \frac{1}{N^2} \sum_i (E_{pre} - E_{true})^2 + \frac{1}{N^2} \sum_i (\rho_{pre} - \rho_{true})^2$$



# ■ TD-GradDFT – Response Kernel with Automatic Differentiation

- Total XC Energy

$$E_{xc}^{\theta}[\rho(\mathbf{r})] = \int d\mathbf{r} e_{xc}^{\theta}[\rho(\mathbf{r})]$$

- XC Energy on grid  $e_{xc}^{\theta}[\rho(\mathbf{r})]$

- XC Potential on Grid

$$v_{xc}^{\theta}[\rho(\mathbf{r})] = \frac{\partial e_{xc}^{\theta}[\rho(\mathbf{r})]}{\partial \rho(\mathbf{r})}$$
$$f_{xc}^{\theta}[\rho(\mathbf{r})] = \frac{\partial^2 e_{xc}^{\theta}[\rho(\mathbf{r})]}{\partial \rho^2(\mathbf{r})}$$

- Calculated by jax

$$e_{xc}^{\theta}[\rho(\mathbf{r})] \xrightarrow{\text{jax.grad()/jax.hessian()}} v_{xc}^{\theta}[\rho(\mathbf{r})] \xrightarrow{\text{jax.grad()/jax.hessian()}} f_{xc}^{\theta}[\rho(\mathbf{r})]$$

```
```python
def point_energy(variables: Array) -> Array:
    rho_point = jnp.maximum(variables[0],
density_floor_value)
    if kind_norm == "LDA":
        grad_point = jnp.zeros((3,),
dtype=variables.dtype)
        tau_point = jnp.asarray(0.0,
dtype=variables.dtype)
    elif kind_norm == "GGA":
        grad_point = variables[1:4]
        tau_point = jnp.asarray(0.0,
dtype=variables.dtype)
    elif kind_norm == "MGGA":
        grad_point = variables[1:4]
        tau_point = jnp.maximum(variables[4], 0.0)
    ...
    return eval_xc_energy_density(spec_norm, features)

return jax.jit(jax.vmap(jax.hessian(point_energy)))
```
```

# ■ TD-GradDFT – Response Kernel with Automatic Differentiation

- **LDA**

$$E_{xc}[\rho(\mathbf{r})] = \int d\mathbf{r} e_{xc}[\rho(\mathbf{r})]$$

$$f_{xc}[\rho(\mathbf{r})] = \frac{\delta^2 e_{xc}}{\delta \rho}$$

- **GGA**

$$E_{xc} = \int d\mathbf{r} e_{xc}[\rho(\mathbf{r}), \nabla \rho(\mathbf{r})]$$

$$\delta E_{xc} = \int d\mathbf{r} \delta e_{xc}[\rho(\mathbf{r}), \nabla \rho(\mathbf{r})] = \int d\mathbf{r} \frac{\partial e_{xc}}{\partial \rho} \delta \rho + \frac{\partial \nabla e_{xc}}{\partial \nabla \rho} \delta \nabla \rho$$

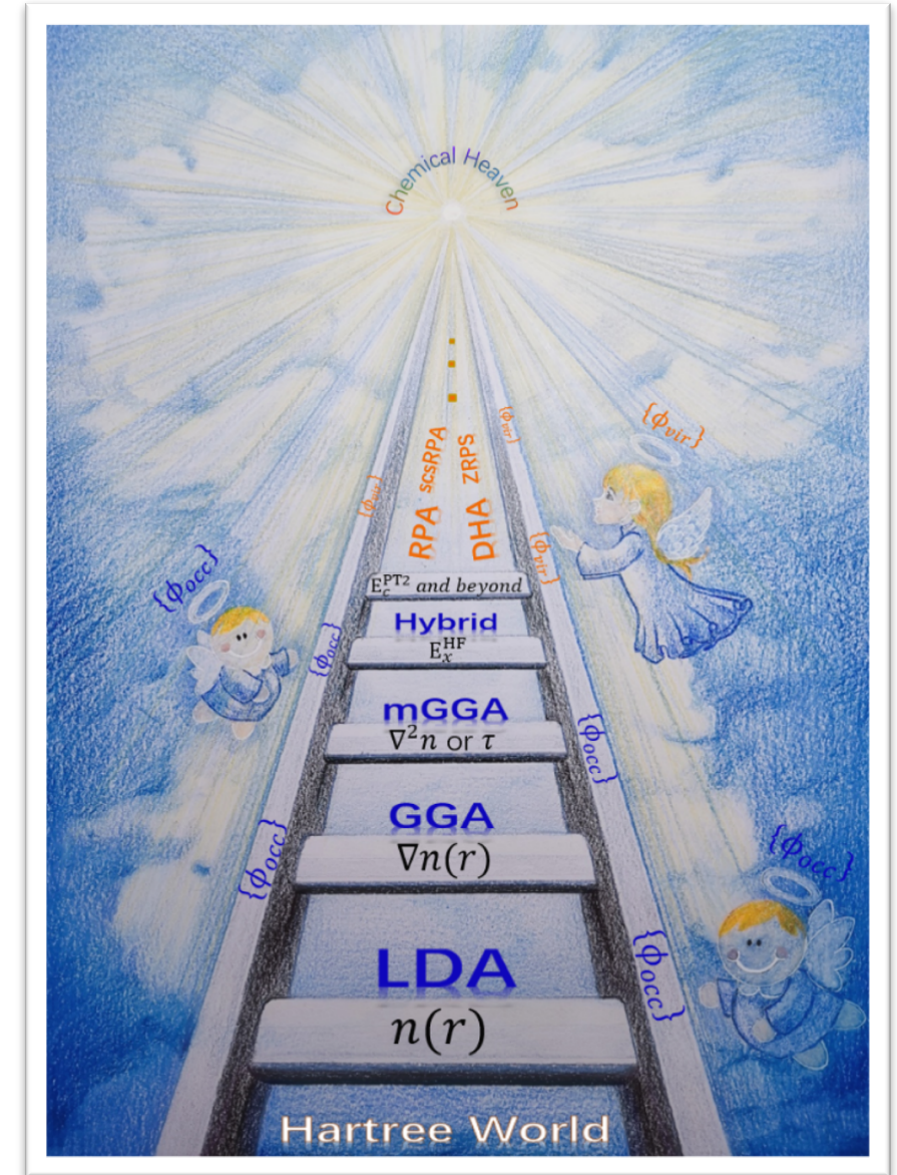
$$\delta^2 E_{xc} = \int d\mathbf{r} \sum_{\mu\nu} \frac{\partial^2 e_{xc}}{\partial u_\nu \partial u_\mu} \delta u_\nu \delta u_\mu$$

- **Meta-GGA**

$$E_{xc} = \int d\mathbf{r} e_{xc}[\rho(\mathbf{r}), \nabla \rho(\mathbf{r}), \tau]$$

$$f_{xc} = \frac{\delta^2 e_{xc}}{\partial u_\nu \partial u_\mu} \quad \tau = -\frac{1}{2} \sum_i |\nabla \psi|^2$$

$$u_\mu = [\rho(\mathbf{r}), \nabla \rho(\mathbf{r}), \tau]$$



# ■ TD-GradDFT – Response Kernel with Automatic Differentiation

- HF Exchange Kernel

$$E_x^{HF} = \int d\mathbf{r} c_x^{hf}[\rho(\mathbf{r})] e_x^{hf}[\phi(\mathbf{r})]$$

$$e_x^{hf}[\phi] = -\frac{1}{2} \sum_{pq} \phi_p^*(\mathbf{r}) \phi_q(\mathbf{r}) \int d\mathbf{r}' \frac{\phi_i^*(\mathbf{r}') \phi_j(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}$$

In A matrix of Casida equation

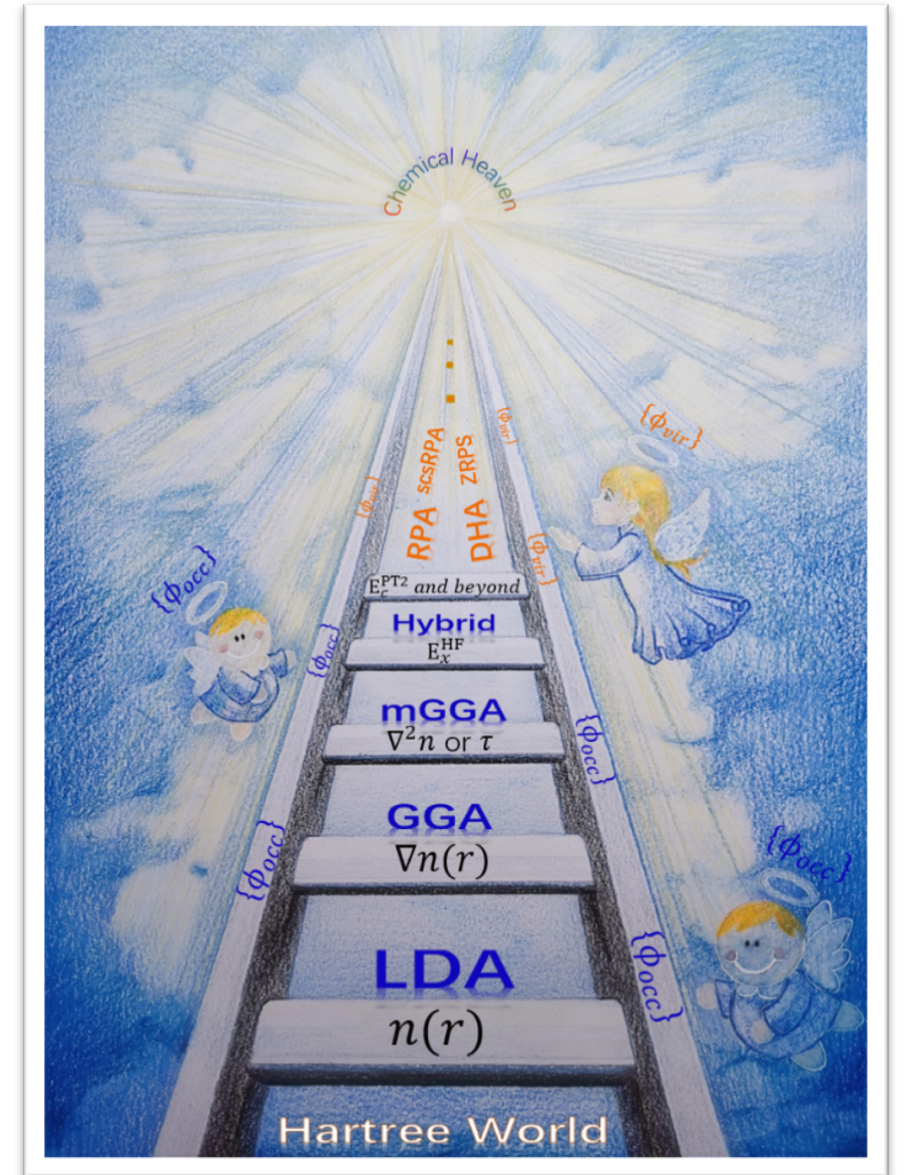
$$K_{ai,bj} = -\alpha_{eff}(ij|ab) \quad \text{with} \quad \alpha_{eff} = \frac{\int c_x^{hf}[\rho(\mathbf{r})] \rho(\mathbf{r}) d\mathbf{r}}{\int \rho(\mathbf{r}) d\mathbf{r}}$$

- PT2 kernel

$$E_c^{PT2} = \sum_{ia,jb} \frac{(ai|jb)[2(ai|jb) - (ib|ja)]}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b} \rightarrow \int d\mathbf{r} c_c^{pt2}[\rho] e_c^{pt2}[\phi(\mathbf{r})]$$

$$e_c^{PT2} = \sum_{ia,jb} \rho_{ai} V_{jb}(\mathbf{r}) \frac{[2(ai|jb) - (ib|ja)]}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b}$$

$$f_{xc}^{pt2} = \frac{\partial^2 c_c^{pt2}[\rho]}{\partial u_\nu \partial u_\mu} e_c^{pt2}[\phi(\mathbf{r})] + \frac{\partial^2 e_c^{pt2}[\phi(\mathbf{r})]}{\partial u_\nu \partial u_\mu} c_c^{pt2}[\rho]$$



# ■ TD-GradDFT – Response Kernal with Automatic Differentiation

The Excited States of Benzene Molecule with PBE0/STO-3G in PySCF and TD-GradDFT

| state | Enery/ev(pyscf) | Enery/ev(ours) | Error/ev | OSC(pyscf) | OSC(ours) | Error    |
|-------|-----------------|----------------|----------|------------|-----------|----------|
| 1     | 6.56            | 6.56           | 3.21E-04 | 4.55E-11   | 6.83E-11  | 2.29E-11 |
| 2     | 7.97            | 7.98           | 8.22E-04 | 6.97E-10   | 4.32E-09  | 3.62E-09 |
| 3     | 9.08            | 9.08           | 2.42E-04 | 6.64E-01   | 6.64E-01  | 8.88E-05 |
| 4     | 9.08            | 9.08           | 5.34E-04 | 6.64E-01   | 6.64E-01  | 1.50E-04 |
| 5     | 9.19            | 9.19           | 2.29E-04 | 2.55E-24   | 2.59E-05  | 2.59E-05 |
| 6     | 9.30            | 9.30           | 1.08E-04 | 1.09E-20   | 2.31E-07  | 2.31E-07 |

# ■ TD-GradDFT – Training Modes in Neural Network XC Functional

- **Fixed-Density Mode**

$$D_\theta = D_{ref}$$

$$L = L(\theta, D) \Big|_{D=D_{ref}}$$

$$\nabla_\theta L = \partial_\theta L(\theta, D) \Big|_{D=D_{ref}} \text{ with } \partial_\theta D = 0$$

- **Explicit SCF Differential Mode**

$$L = L(\theta, D_\theta) \text{ with } D_\theta = T_{SCF}^\theta(D_0)$$

$$\nabla_\theta L = \partial_\theta L(\theta, D_\theta) + \partial_{D_\theta} L(\theta, D_\theta) \cdot \partial_\theta D_\theta$$

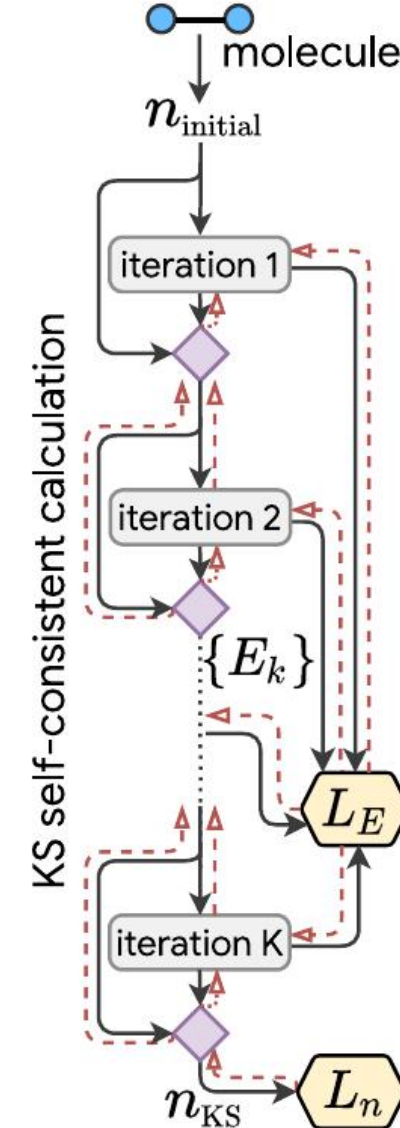
$$\text{with } \partial_\theta D_\theta = \sum_{K-1} [\prod_{s=t+1}^{K-1} \partial_D T_{SCF}^\theta(D^s)] \partial_\theta T_{SCF}^\theta(D^t)$$

- **Implicit SCF Differential Mode**

$$L = L(\theta, D_\theta) \text{ with } D_\theta = T_{SCF}^K(D_0)$$

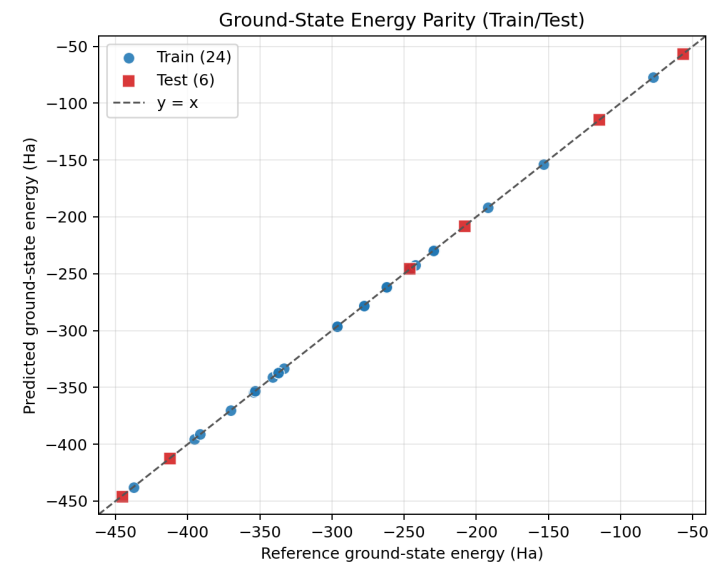
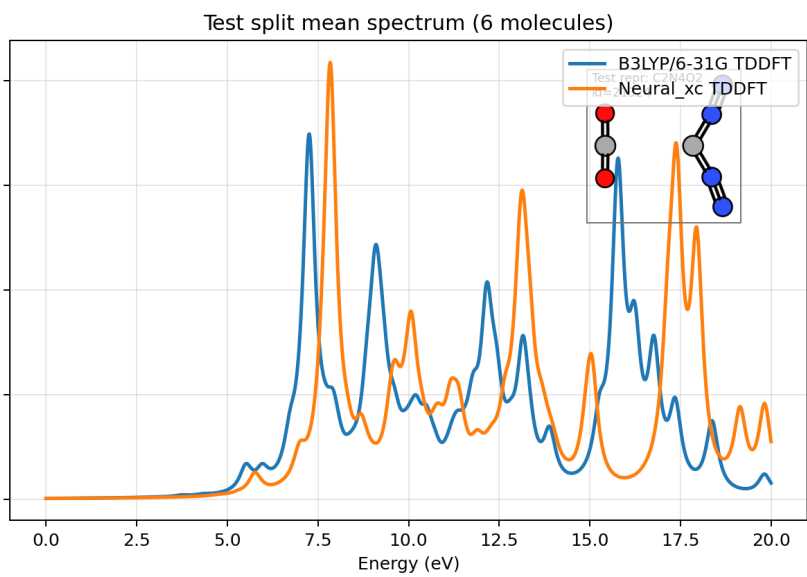
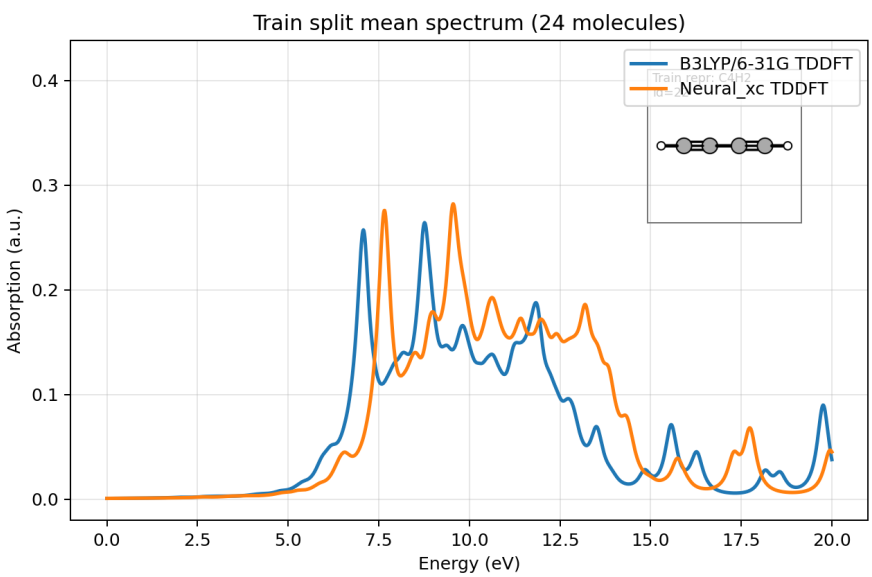
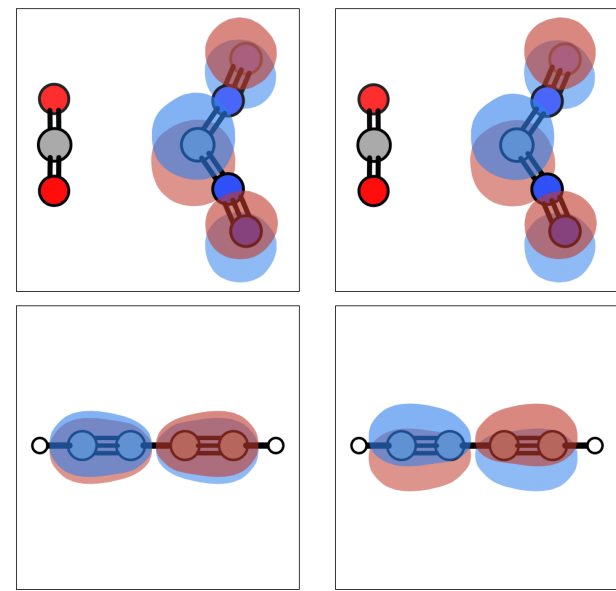
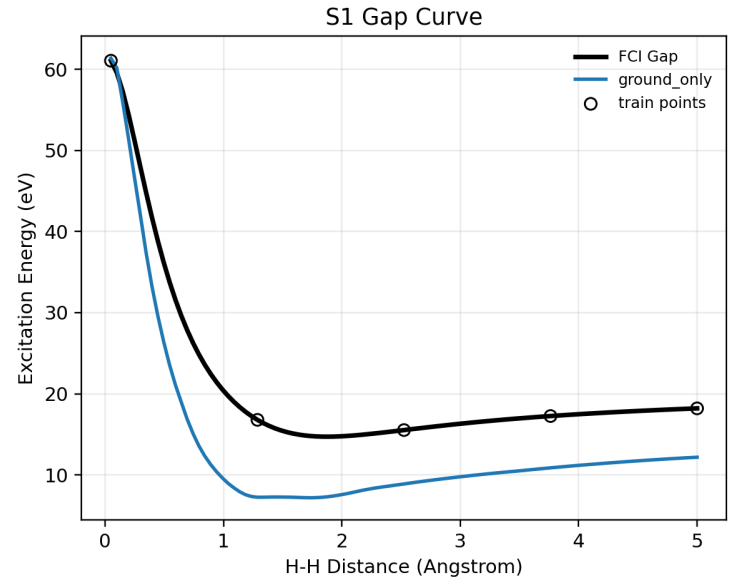
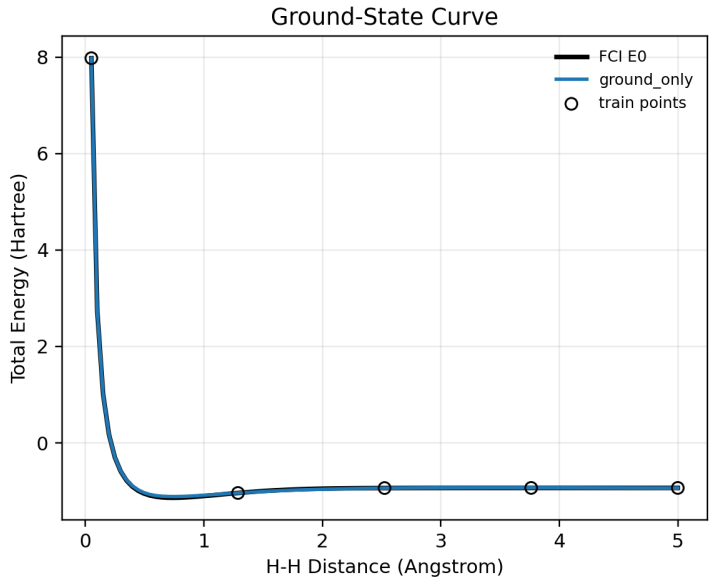
$$R(D_\theta, \theta) = F_\theta(D_\theta)D_\theta S - SD_\theta F_\theta(D_\theta) = 0$$

$$\nabla_\theta L = \partial_\theta L(\theta, D_\theta) + \partial_{\rho_\theta} L(\theta, D_\theta) \frac{\partial_\theta R}{\partial_{D_\theta} R}$$



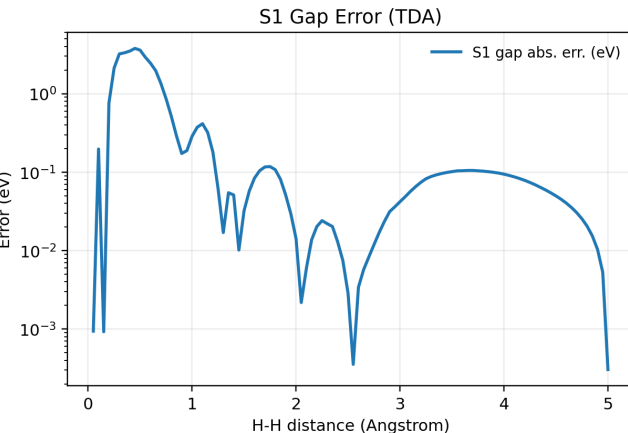
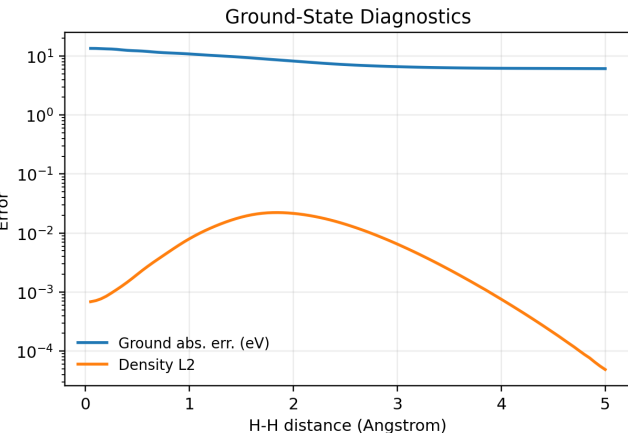
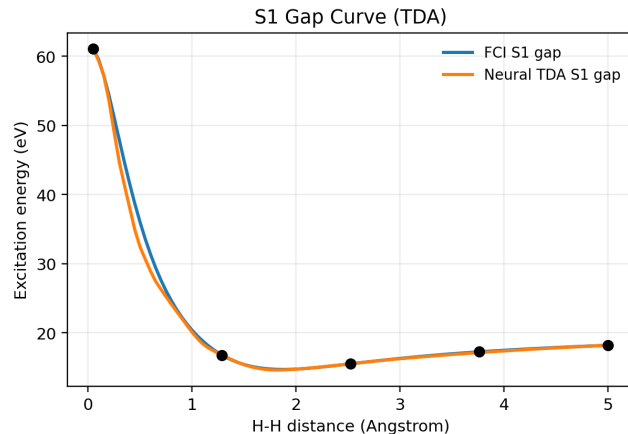
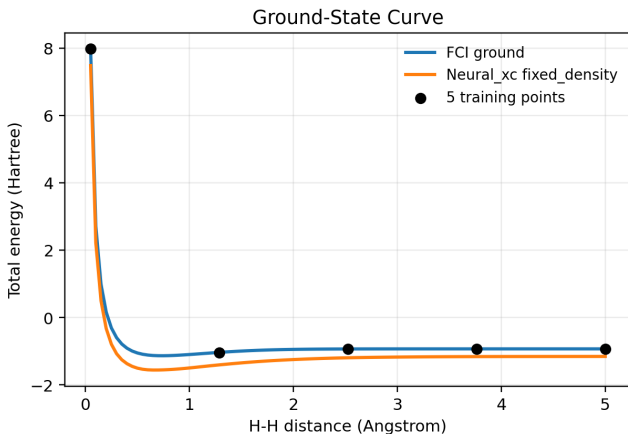
# TD-GradDFT – Training Test on QM9 with Ground State

### Dense H2 Dissociation Inference from Trained Neural\_xc Checkpoints

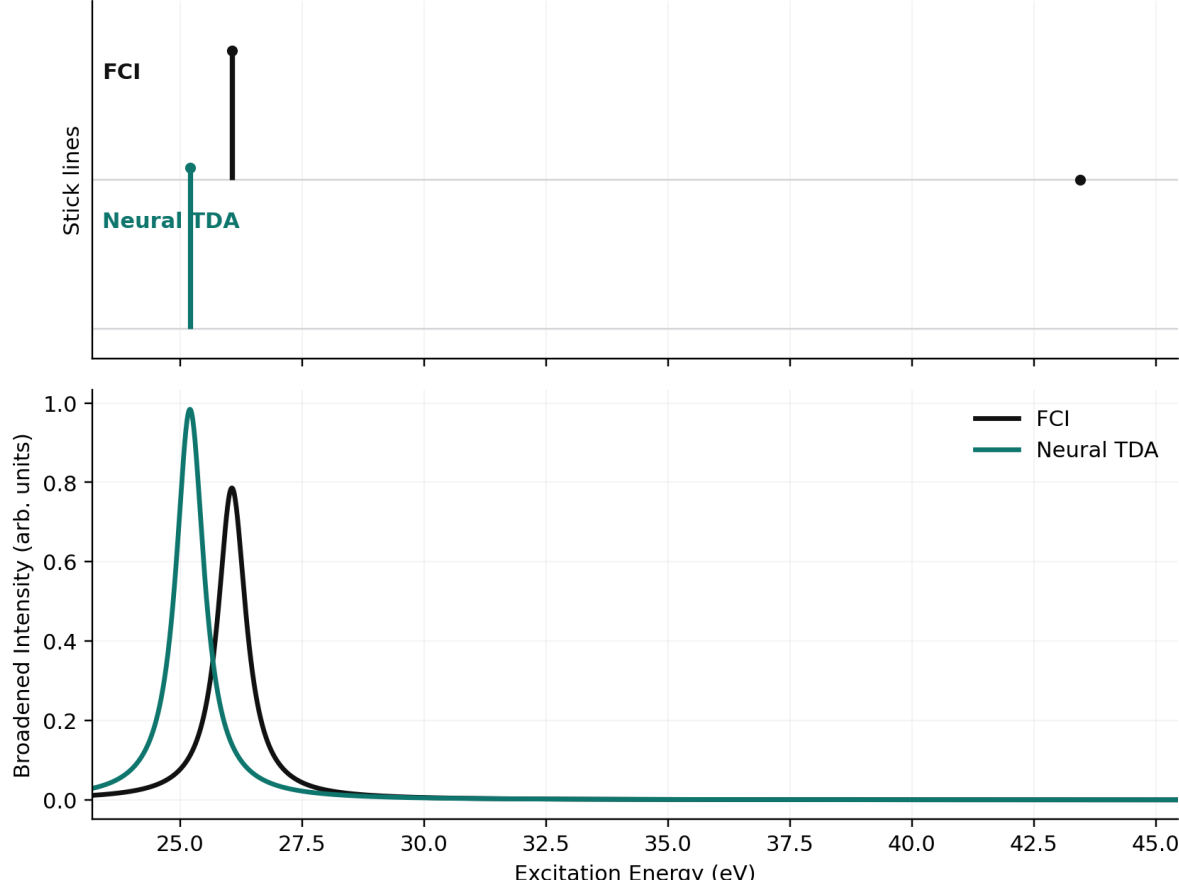


# TD-GradDFT – Training Test on H<sub>2</sub> Dissociation Curve

H2 S1-only TDA training vs FCI | b3lyp/sto-3g

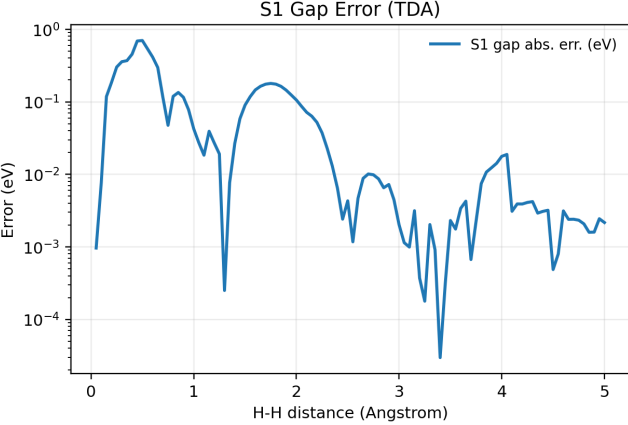
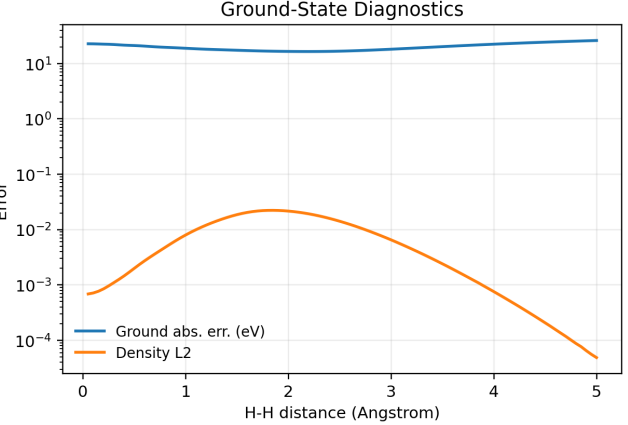
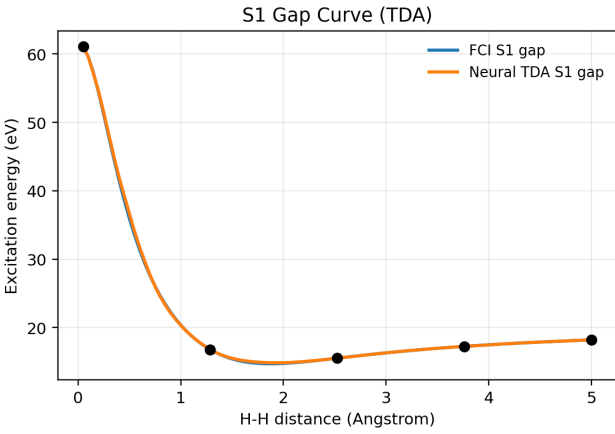
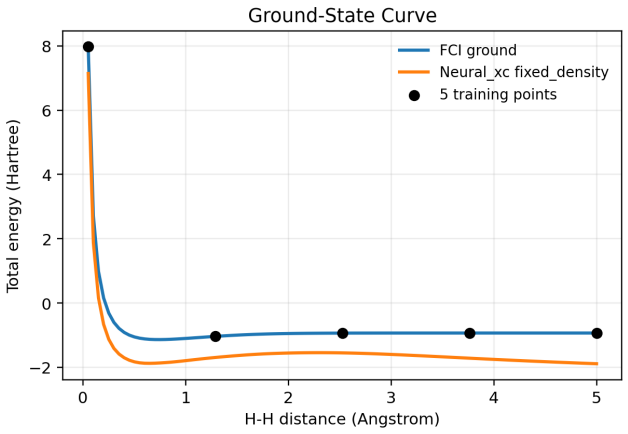


H2 equilibrium stick spectrum at R = 0.75 A

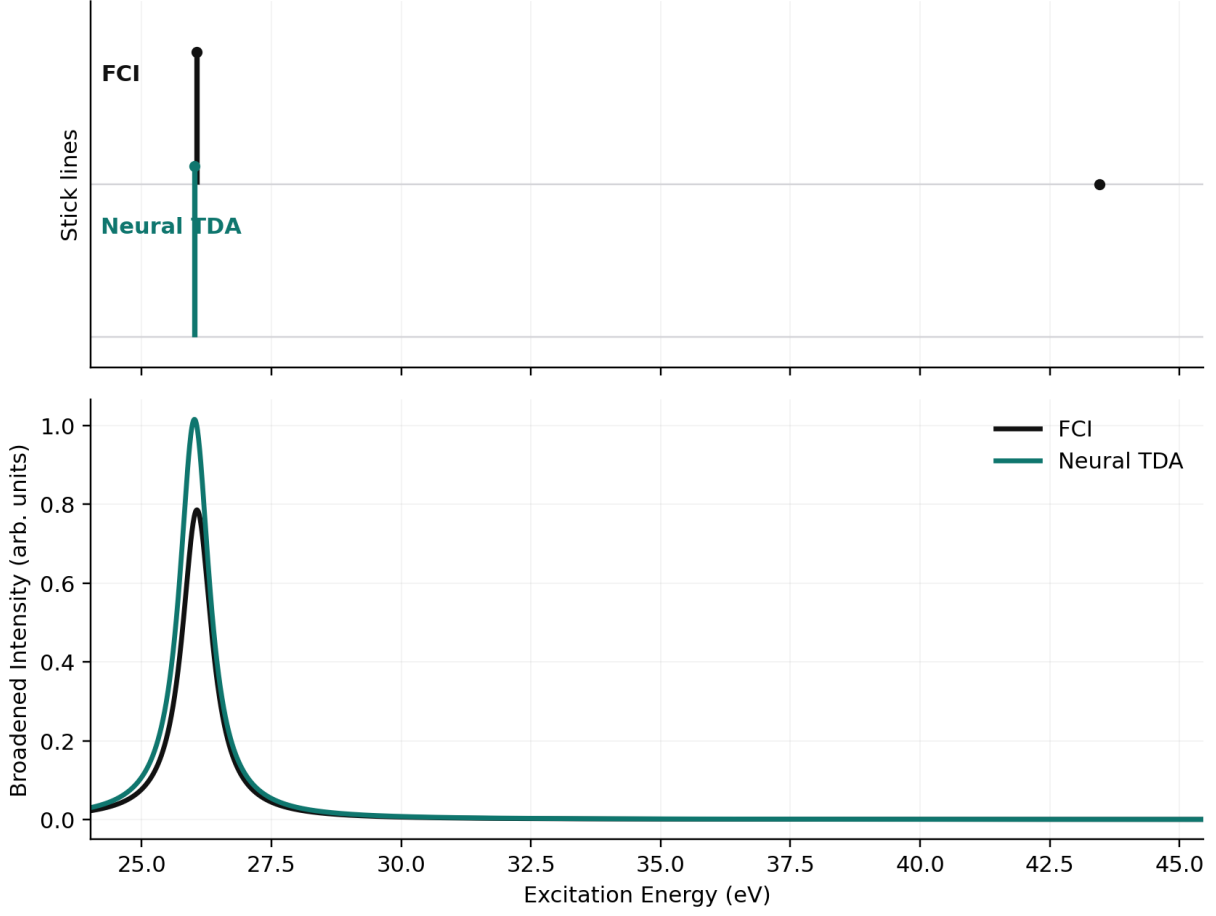


# TD-GradDFT – Improvement for Electrons Double Exciting with MP2 Hybrid

H2 S1-only TDA training vs FCI | b3lyp/sto-3g



H2 equilibrium stick spectrum at R = 0.75 A



# ■ TD-GradDFT – Failure of Full Absorption Spectrum Calculation

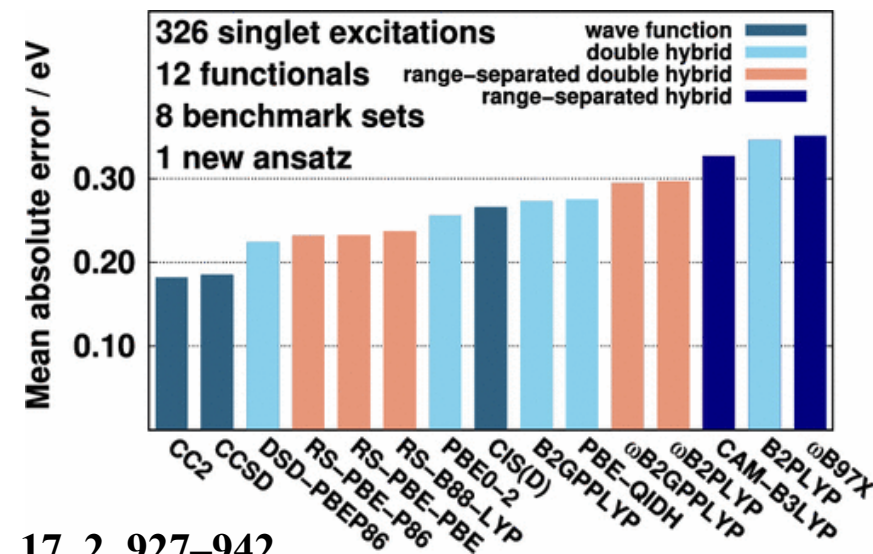
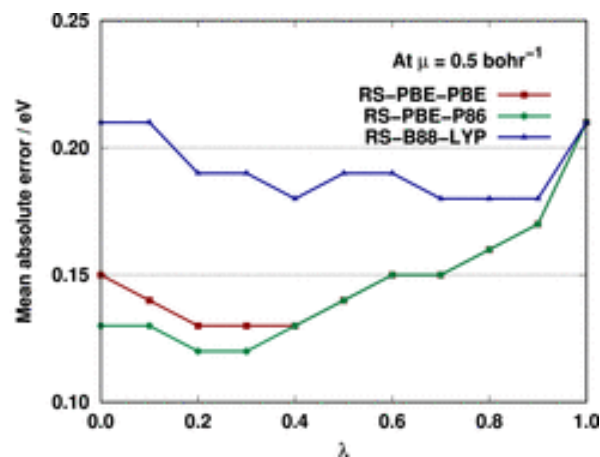
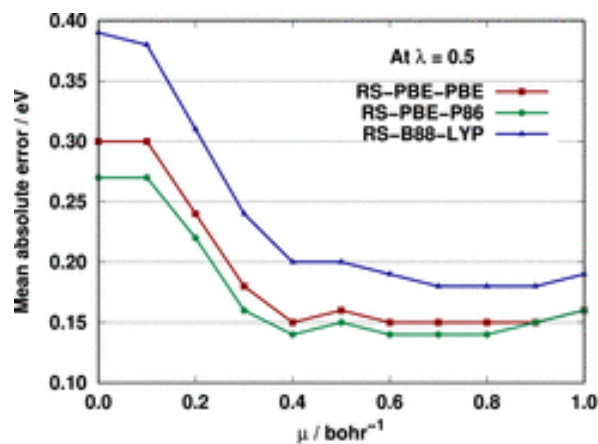
- TDDFT is not a rigorous theorem for excited state wave function

$$\psi_{ex}^{CIS} = \phi_0 + \sum_{ai} x_{ai} \phi_a^i$$

$$\psi_{ex}^{FCI} = \phi_0 + \sum_{ai} c_{ai} \phi_a^i + \sum_{aibj} c_{aibj} \phi_{ab}^{ij} + \sum_{aibjck} c_{aibjck} \phi_{abc}^{ijk} + \dots$$

- No Ground Truth has a good performance both on ground state and excited state for complex electron configuration

**But !**

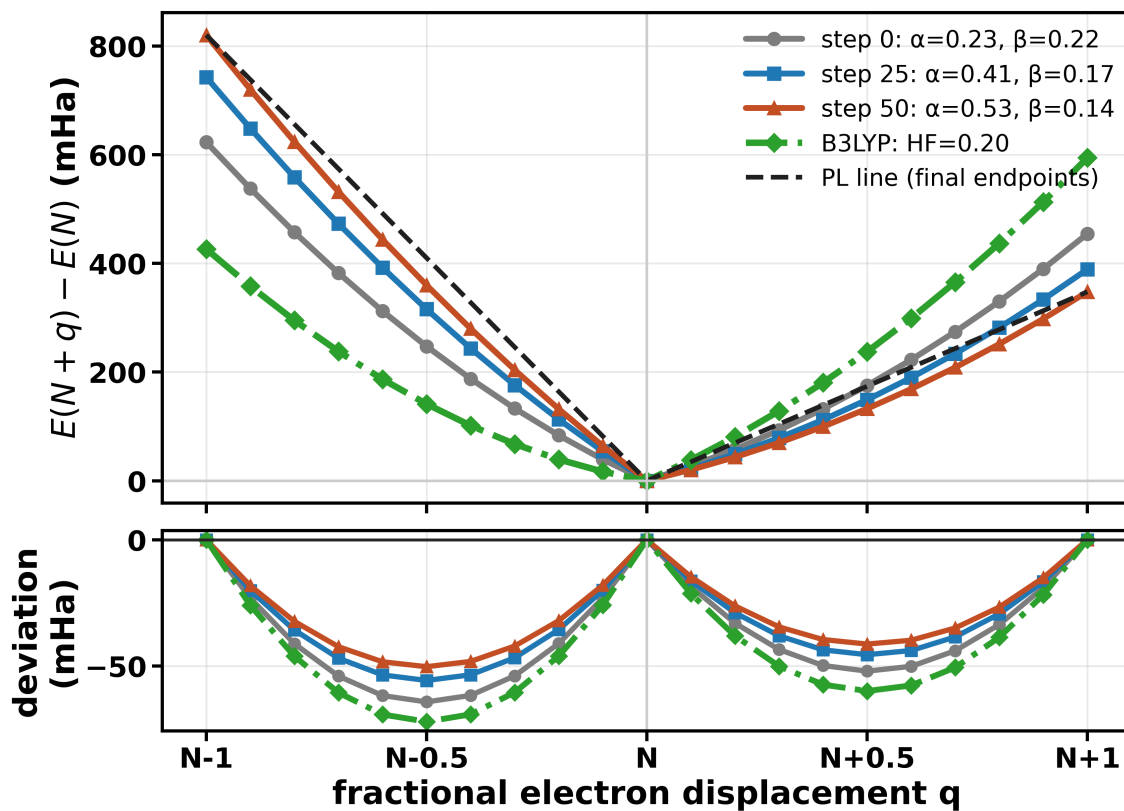


# ■ TD-GradDFT – Functional and Outlook

## ● Outlook

- Traditional RSH XC Functional optimization
- Rigorous MP2 Response Kernel
- Benchmarking three differentiable mode
- More efficient code !

Fractional Occupation Curves: nn-RSH vs B3LYP



---

## ● Functional

- Differential restricted SCF code with three modes and differential LR-TDDFT code
- HF and PT2 local energy density hybrid
- Traditional RSH XC Functional optimization only depending on Janak Theorem (Under Testing)
- Fractional charge evaluation and differential unrestricted SCF code (Under Testing)

# DL-TDDFT – Accelerating TDDFT with CIS-like Graph Neural Network

## Davidson Method Based

- **Subspace**

$$V_m = [v_1, v_2, \dots, v_m] \quad W_m = AV_m$$

- **Residuals**

$$H_m = V_m^T AV_m = V_m^T W_m$$

$$H_m y_k = \theta_k y_k, x_k = V_m y_k$$

$$r_k = Ax_k - \theta_k x_k$$

- **Expansion**

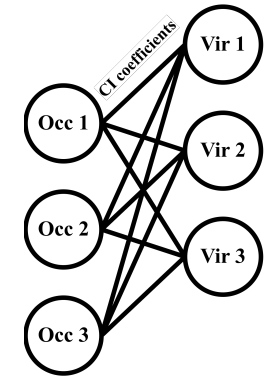
$$t_k \approx (\text{diag}(A) - \theta_k I)^{-1} r_k$$

$$V_{(m+1)} = \text{orth}\left([V_m, t_1, t_2, \dots]\right)$$

## Graph Neural Network Based

$$n_p = [C_p || \epsilon_p]$$

**GAT-V2**  $\longrightarrow$



- **Node**

$$o_i = f_{occ}(n_i) \quad v_a = f_{vir}(n_a)$$

- **Edge**

$$e_{ai} = f_{vir}(\eta_{ai}) \quad \eta_{ai} = [r_{ai}, \Delta\epsilon_{ai}, d_{ai}]$$

- **Interaction and Transition**

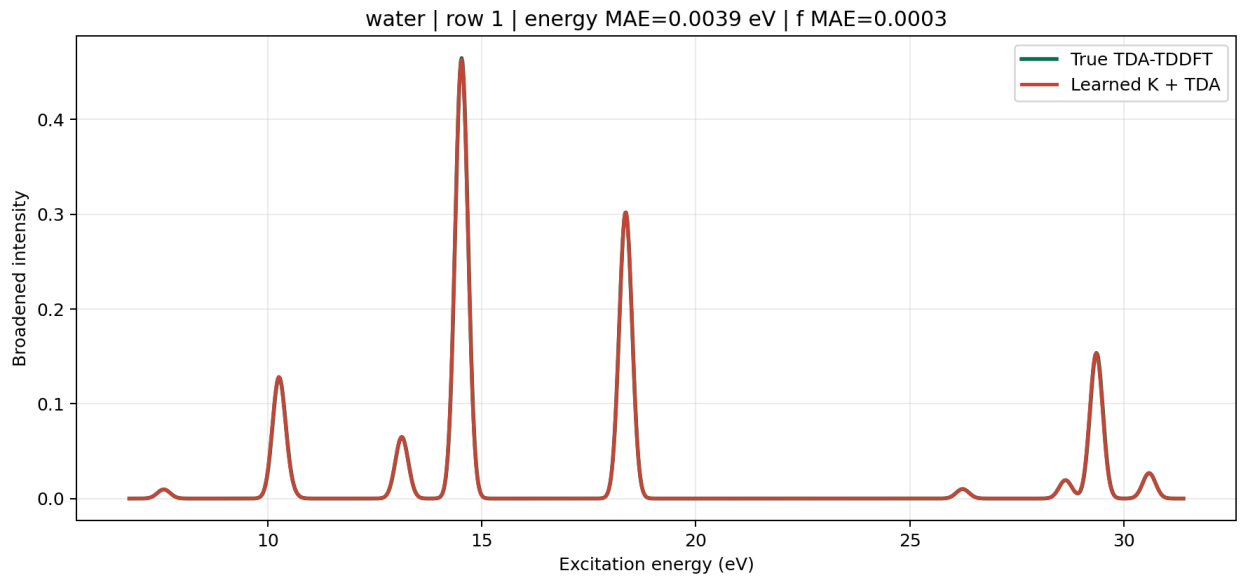
$$z_{ai} = f_{int}([o_i^l \otimes v_a^l, o_i^l - v_a^l, e_{ai}^l])$$

- **Pair-Pair Decoder**

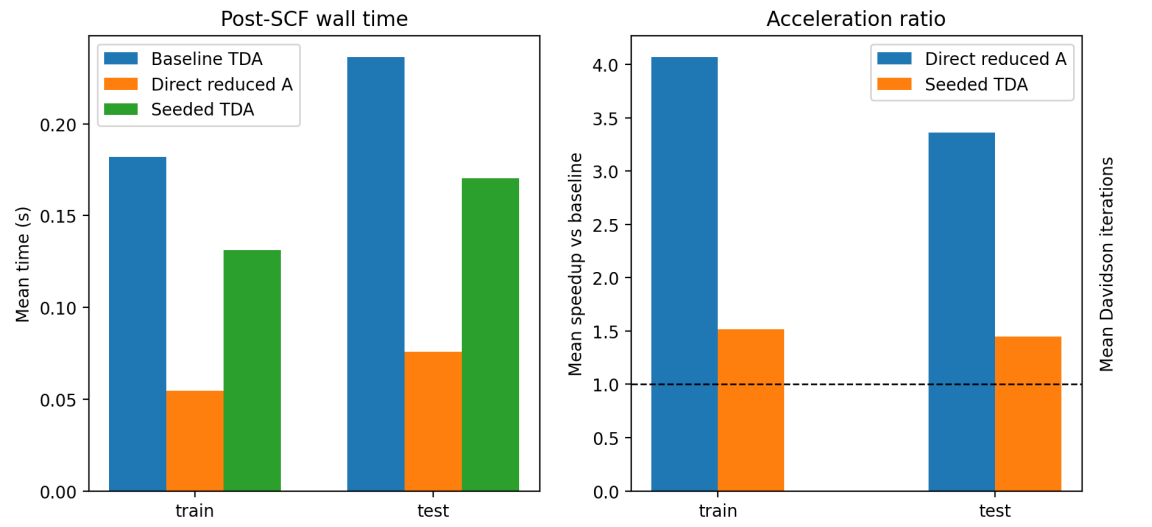
$$D_{ai,bj} = f([z_{ai} \otimes z_{bj}, z_{ai} - z_{bj}, \phi_{ai,bj}])$$

# DL-TDDFT – A Test for Accelerating TDDFT

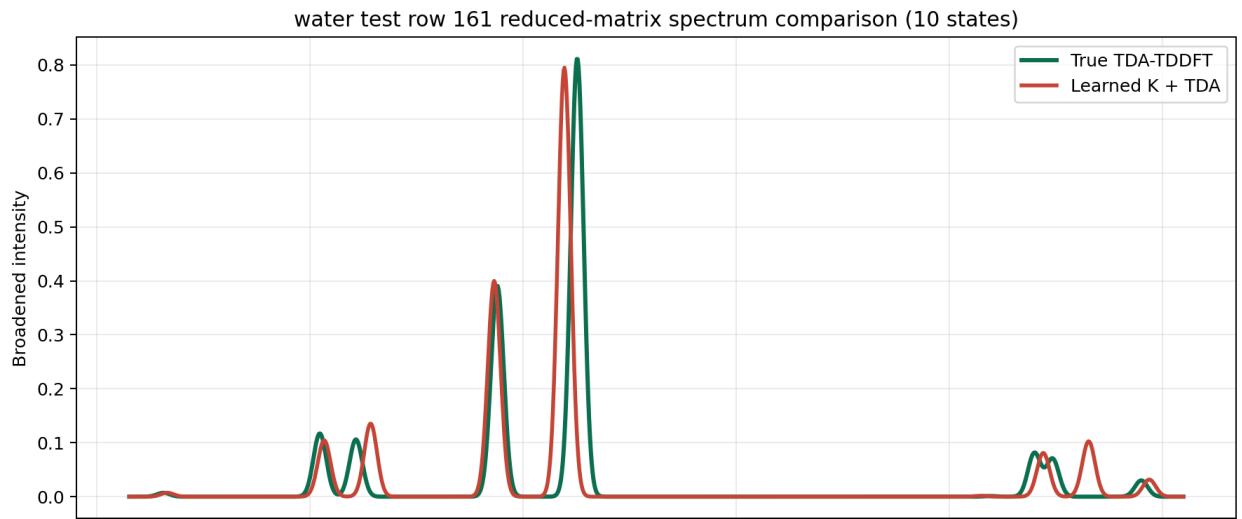
## Water in Full Excitation Space - training set



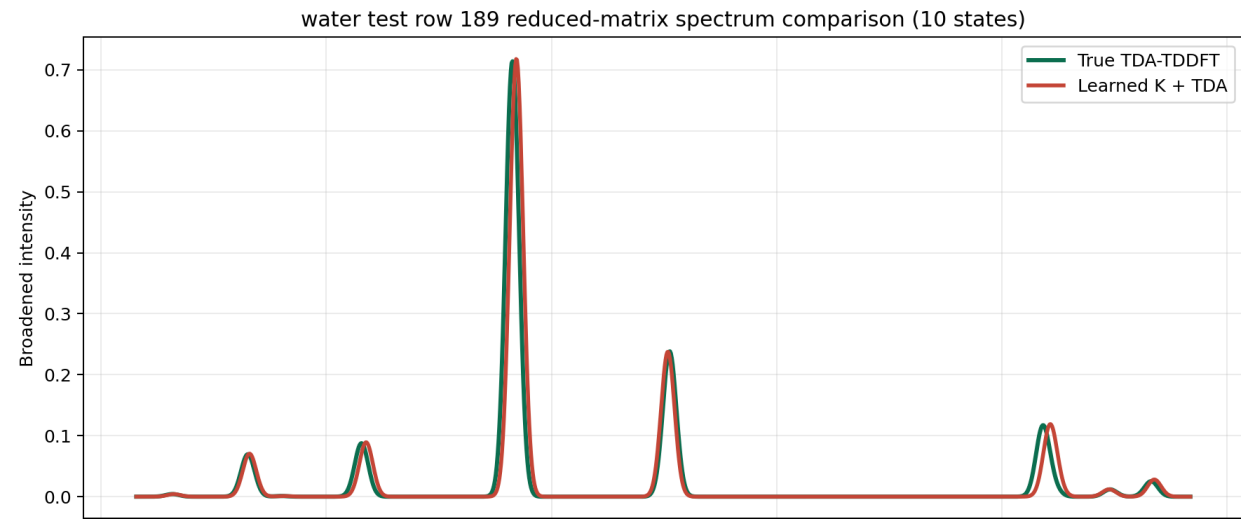
## Speed up



## Water in Full Excitation Space - test set

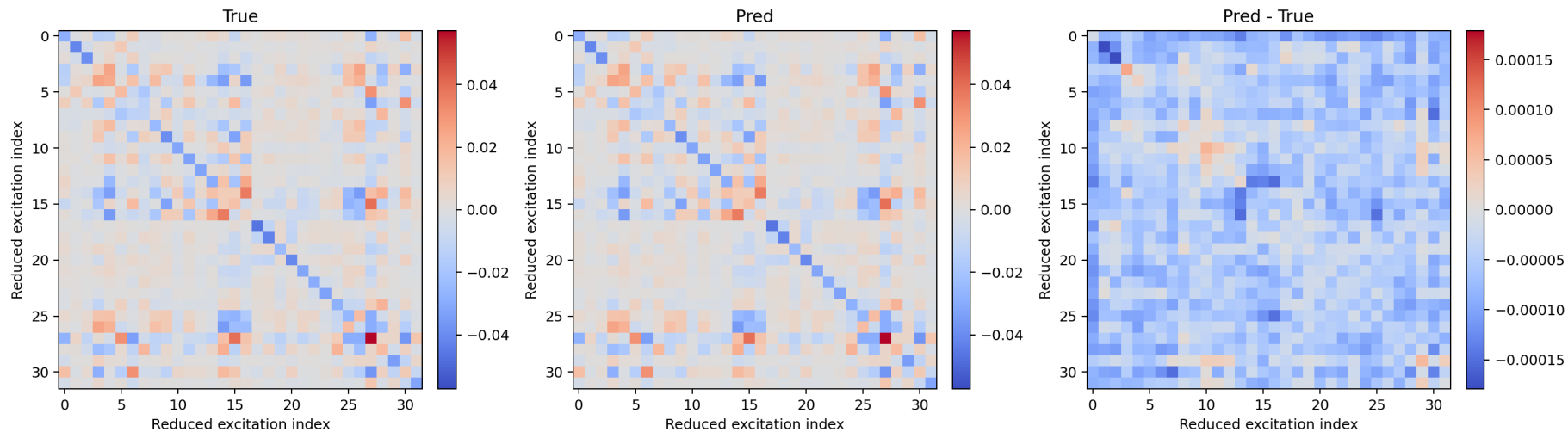


## Water in Full Excitation Space - test set



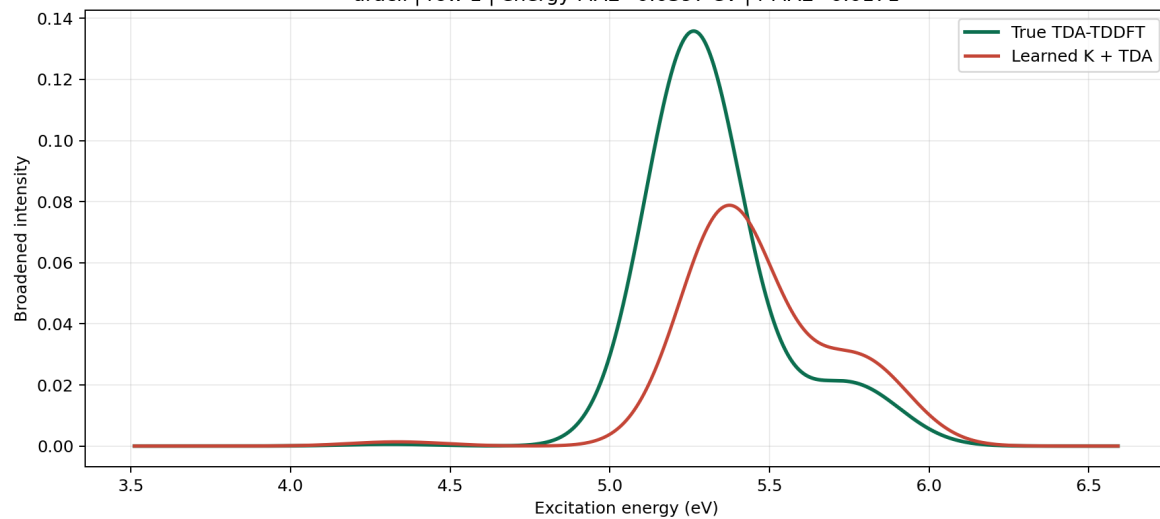
# DL-TDDFT – A Test for Accelerating TDDFT

uracil train row 1 | reduced K | MAE=0.000053, RMSE=0.000061



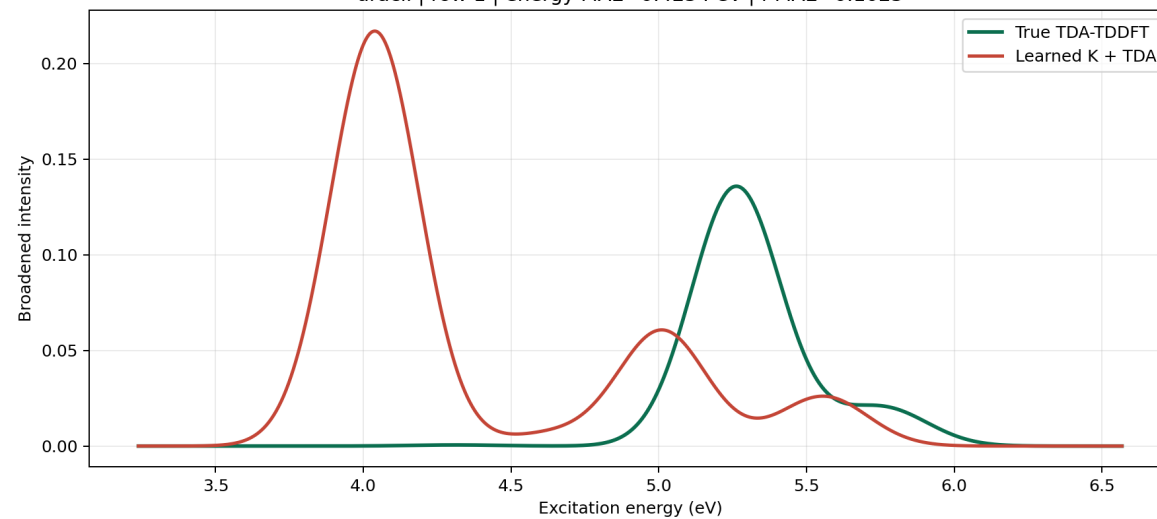
## Uracil in Excitation Subspace( $32 \times 32$ )

uracil | row 1 | energy MAE=0.0397 eV | f MAE=0.0171



## Uracil in Excitation Subspace( $64 \times 64$ )

uracil | row 1 | energy MAE=0.4254 eV | f MAE=0.1025



# ■ Background – Inspiration From Claude Code

## Water Molecule: Setup and Calculations

### Computational Setup:

- **Molecule:** H<sub>2</sub>O (water molecule)
- **Method:** TDDFT with various functionals
- **Basis Set:** 6-31G(d) and 6-31+G(d,p)
- **Functionals Tested:**
  - PBE (GGA functional)
  - B3LYP (Hybrid functional)
  - CAM-B3LYP (Long-range corrected hybrid)

### Vibe Coding Approach:

- Used natural language to describe requirements
- AI assistant generated Python code for TDDFT calculations
- Automated spectrum plotting and analysis
- Interactive refinement of visualization

## Comparison of Different Functionals

### Key Observations:

- PBE underestimates excitation energies
- B3LYP provides better agreement with experiment
- CAM-B3LYP improves for charge-transfer states
- Basis set effects are significant

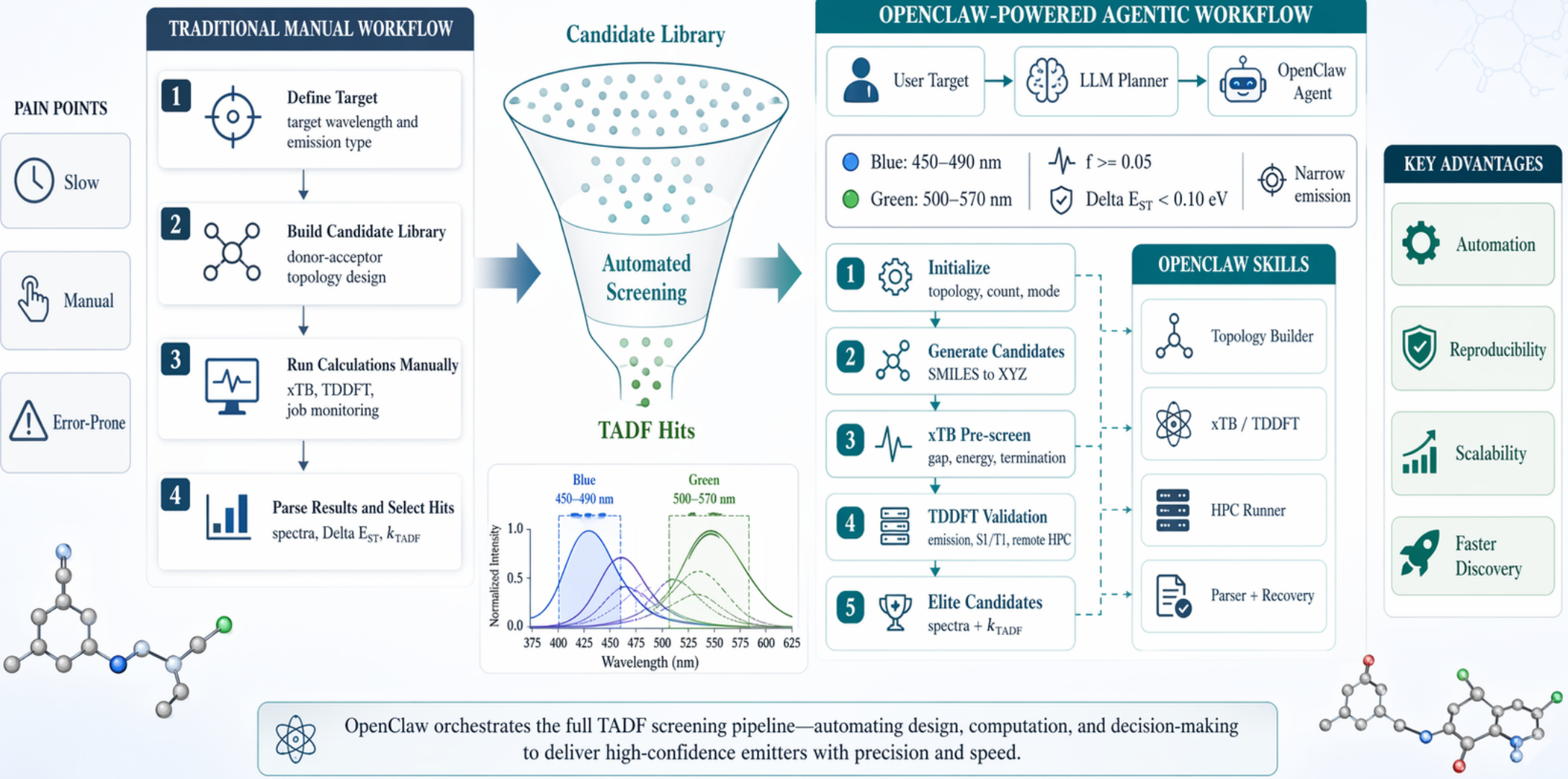
| Functional | Peak (eV) | Intensity |
|------------|-----------|-----------|
| PBE        | 7.2       | 0.85      |
| B3LYP      | 7.8       | 0.92      |
| CAM-B3LYP  | 8.1       | 0.88      |
| Exp.       | 7.9       | -         |

Table 1: Comparison of excitation energies

# Background – High Throughput Materials Design

[github.com/silico-quantum/tadf-screening](https://github.com/silico-quantum/tadf-screening)

## TRADITIONAL TADF SCREENING VS. OPENCLAW-POWERED AGENTIC WORKFLOW



# ■ Openclaw and Hermes – Basic Architecture

```
# Silico 🐙 - AI Research Partner
```

```
## Architecture
```

```
Layer	Components
**Identity**	`SOUL.md` · `IDENTITY.md` · `USER.md`
**Memory**	`MEMORY.md` · `memory/*.md`
**Workflow**	`HEARTBEAT.md` · `AGENTS.md` · task plans
**Skills**	OpenClaw tools · chemistry workflows · automation modules
```

```
## Runtime
```

- **\*\*Interface\*\***: Feishu DM
- **\*\*Execution\*\***: Local macOS environment + optional remote HPC
- **\*\*Models\*\***: Configurable LLM backend
- **\*\*Automation\*\***: Scheduled research and workflow jobs

```
## Key Capabilities
```

- Quantum chemistry workflows (DFT, TDDFT, xTB screening)
- Feishu docs / drive / wiki / messaging automation
- Web research, code execution, and repo maintenance
- Long-running task orchestration and progress tracking

```
## Self-Evolution (OpenSpace)
```

```
Every task teaches · Every failure improves
```



```
# Hermes △ - AI Computational Chemistry Partner
```

```
## Architecture
```

```
Layer	Components
**Identity**	Hermes Agent · USER PROFILE · MEMORY
**Memory**	persistent memory · session history · reusable skills
**Tasks**	todo · cron jobs · delegated subagents
**Skills**	PySCF · xTB · Browser · Feishu · GitHub · visualization
```

```
## Runtime
```

- **\*\*Host\*\***: Local Hermes environment + Feishu DM
- **\*\*Local Compute\*\***: PySCF, Python, matplotlib, file/workspace automation
- **\*\*External Access\*\***: web search, browser automation, messaging tools
- **\*\*Model\*\***: OpenAI GPT-5.4 (current session)

```
## Key Capabilities
```

- Computational chemistry workflows (PySCF, xTB, spectra, PES, MD analysis)
- Scientific scripting, plotting, and reproducible project pipelines
- Web lookup, browser verification, and literature cross-checking
- Sub-agent orchestration, file management, and result delivery

```
## Working Style
```

- Direct execution first, then concise reporting
- Prefer reproducible scripts over manual operations
- Scientific correctness over cosmetic imitation
- Fast iteration with explicit documentation of approximations

```
## Self-Evolution
```

```
Every task teaches · Every correction sharpens the workflow
```

<https://github.com/HKUDS/OpenSpace>

# ■ Openclaw – A Standard Skills Group For Quantum Chemistry Calculation

## Core Skills

### 1. Structure & Sampling

- [Molecular Sampler](#): Extract and sample molecular structures from cluster/ONIOM files. Union-Find molecule identification + distance-sorted neighbor sampling.
- [RDKit Chemistry](#): 3D conformer generation (ETKDG), molecular descriptors (LogP, TPSA), and Gasteiger charge analysis.
- [xTB Cluster MD](#): Semi-empirical MD (GFN-FF/GFN2-xTB) for organic molecular clusters with automated trajectory animation.

### 2. Electronic Structure Computing

- [PySCF](#): Python-based quantum chemistry. Supports Ground state (HF, KS-DFT), Excited states (LR-TDDFT, TDA), and Post-HF methods.
- [MOMAP](#): helper for molecular photophysics and charge transport. Predict radiative/non-radiative (IC/ISC) rate constants and quantum yields.

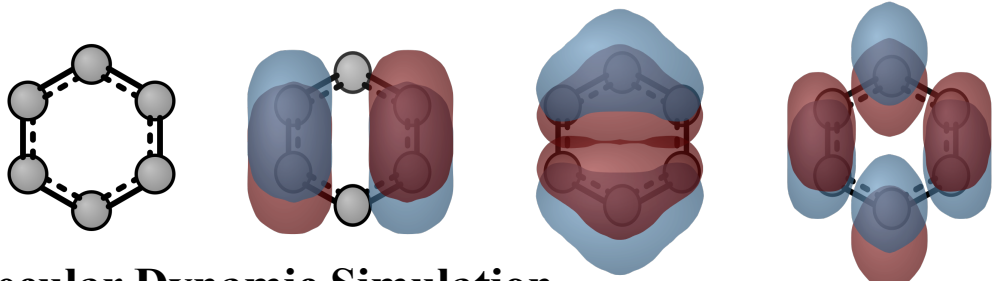
### 3. Analysis & Visualization

- [Multiwfn](#): Advanced wave function analysis. Population analysis (Hirshfeld, ADCH, CM5), bond orders, and spectroscopy (UV-Vis, IR, Raman).
- [xyzrender](#): Command-line driven, publication-quality molecular graphics. Supports transparent backgrounds, bond orders, and orbital rendering.
- [Orbital Analysis](#): Streamlined workflow for MO composition, energy level diagrams, and isosurface generation.

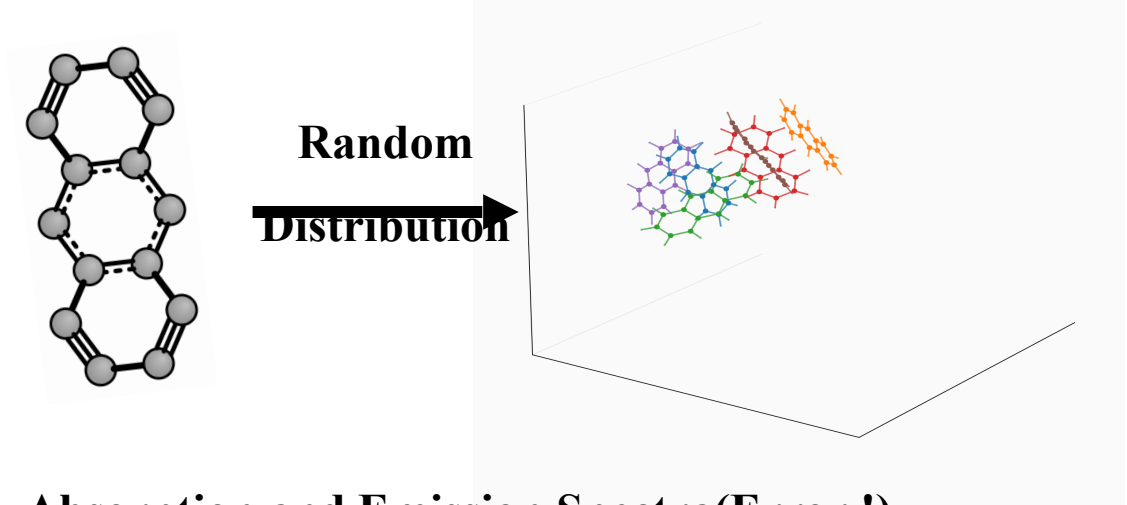
```
pyscf/  
├── SKILL.md  
├── VERSION_UPDATE.md  
├── references/  
│   ├── theory/  
│   │   ├── pyscf-advanced.md  
│   │   ├── pyscf-api-reference.md  
│   │   └── pyscf-jax-integration.md  
│   └── practice/  
│       ├── 2d-potential-energy-surface.md  
│       ├── emission-spectrum-guide.md  
│       └── emission-spectrum-workflow.md  
├── scripts/  
│   └── dft_calculation.py  
└── tools/  
    ├── analysis.py  
    ├── cascf.py  
    ├── ccsd.py  
    ├── dft.py  
    ├── geometry.py  
    ├── mp2.py  
    ├── pes.py  
    ├── scf.py  
    ├── spectrum.py  
    └── tddft.py
```

# ■ Openclaw and Hermes – A Standard Skills Group For Quantum Chemistry Calculation

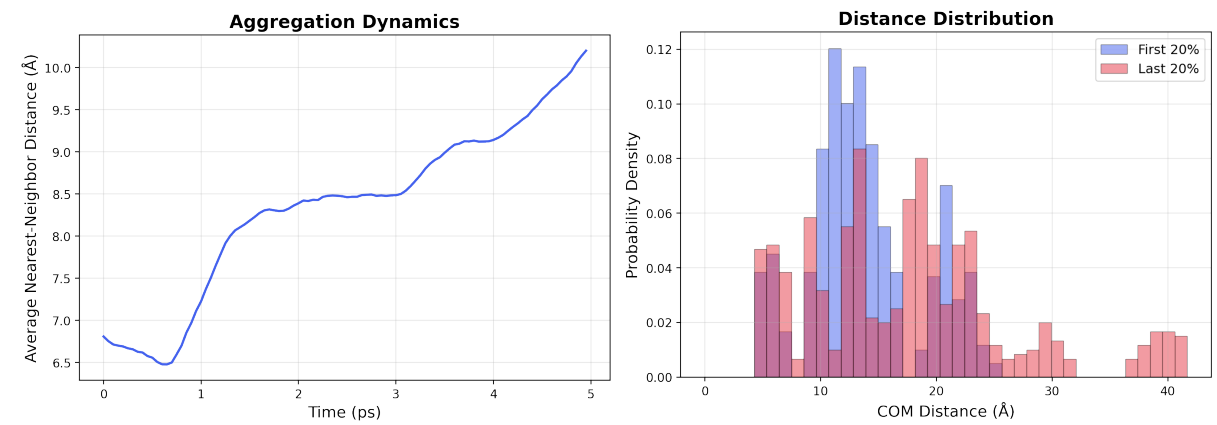
## Basic Ground State Calculation



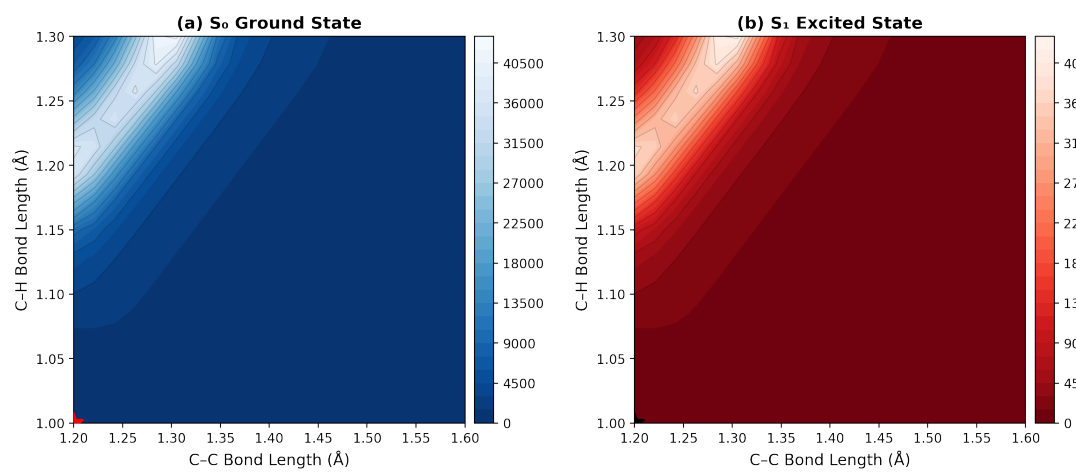
## Molecular Dynamic Simulation Visualization



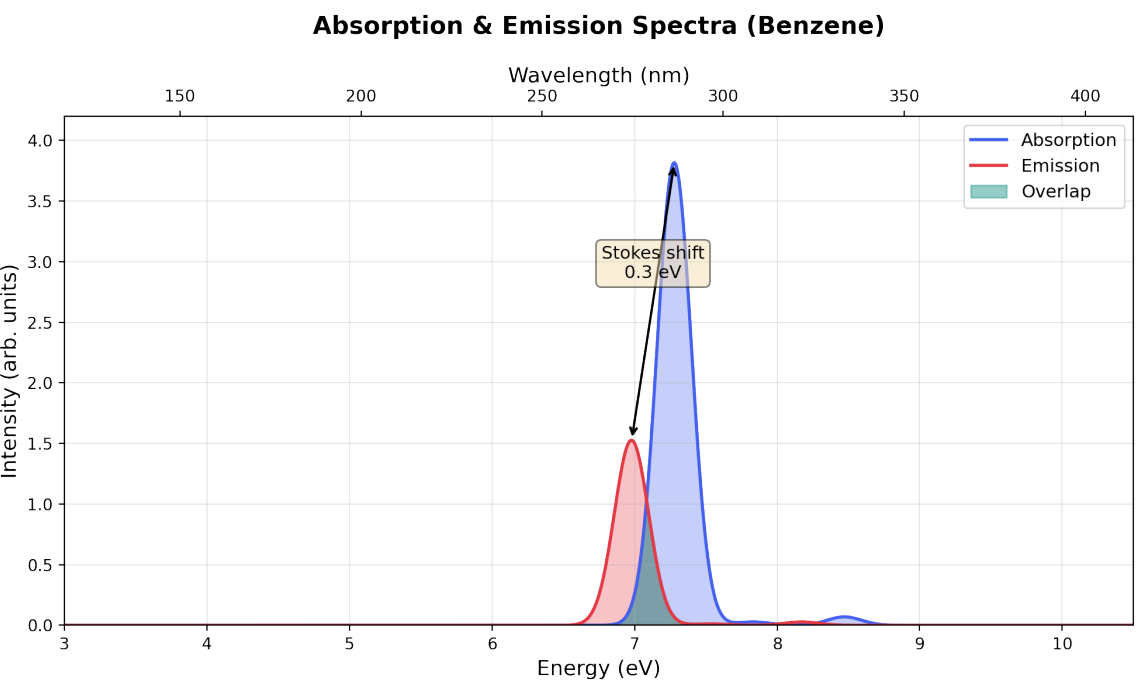
## Molecular Dynamic Simulation



## Potential Energy Surface

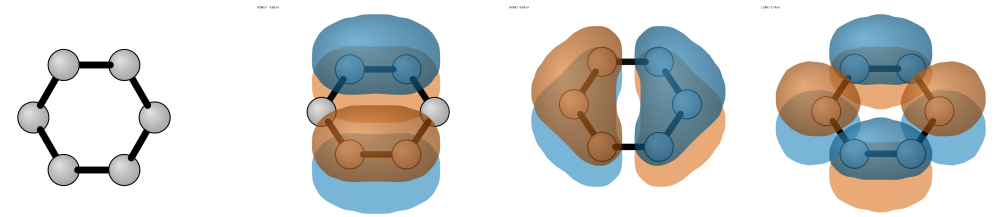


## Absorption and Emission Spectra(Error !)

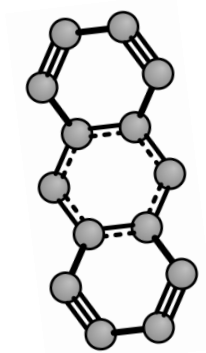


# ■ Openclaw and Hermes – A Standard Skills Group For Quantum Chemistry Calculation

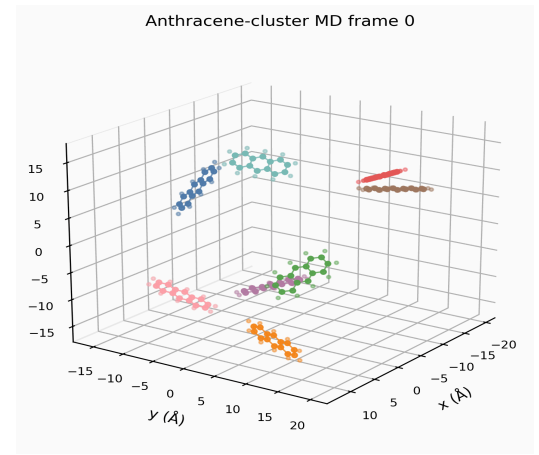
## Basic Ground State Calculation



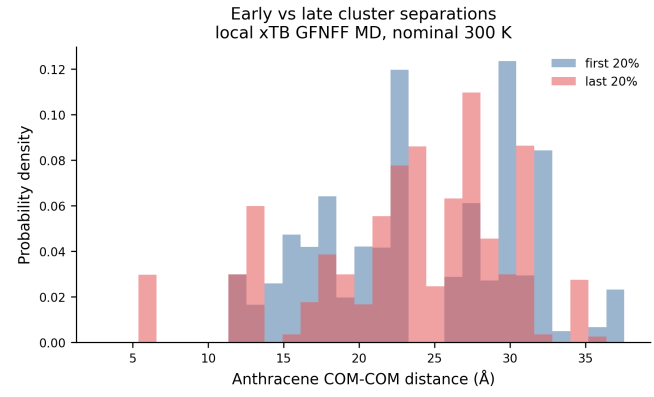
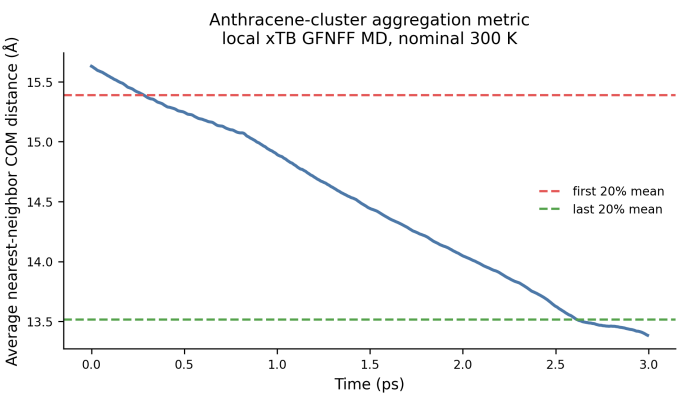
## Molecular Dynamic Simulation Visualization



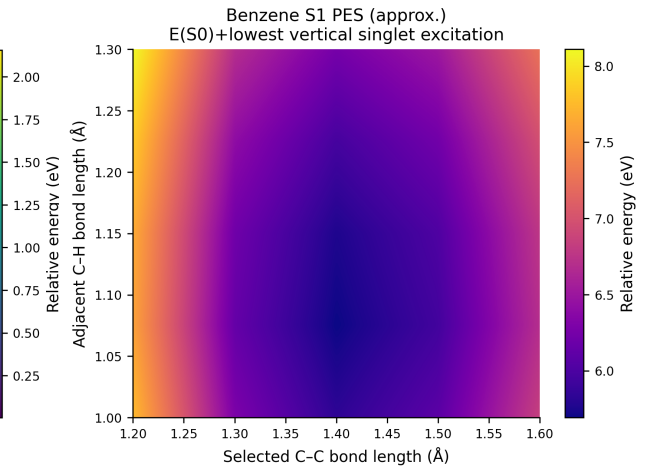
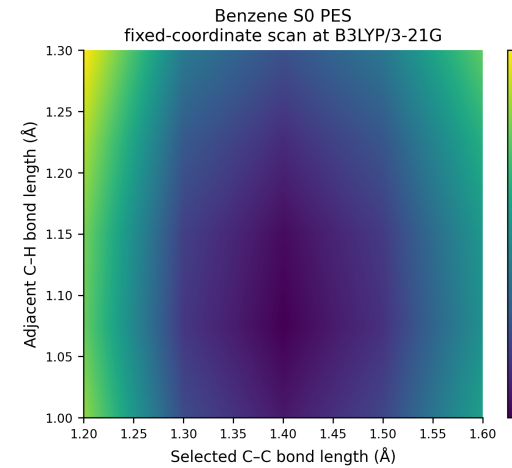
Random  
Distribution



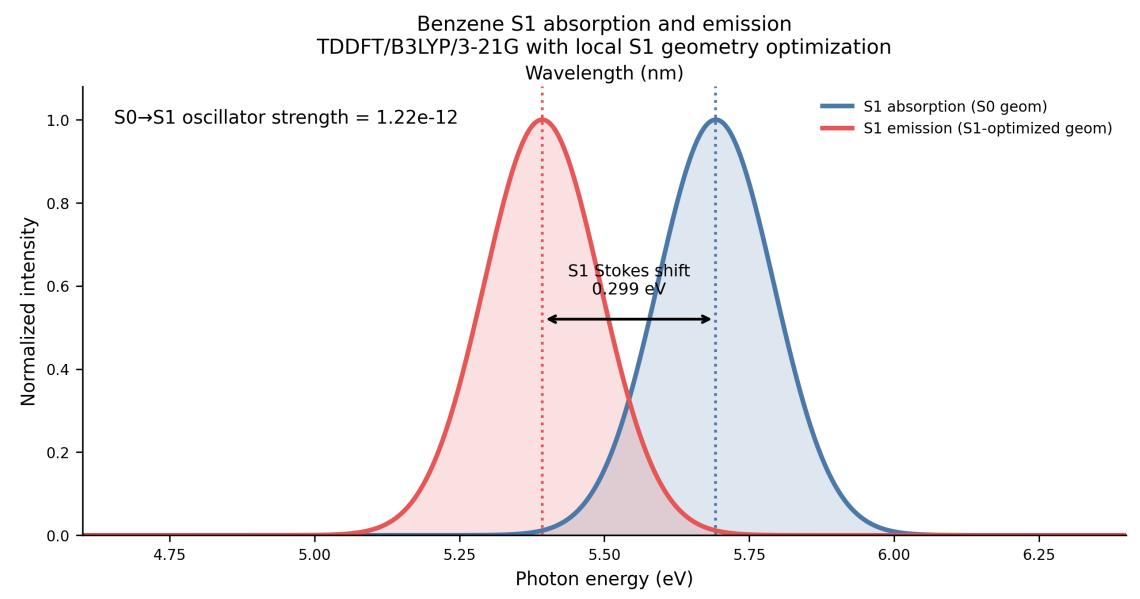
## Molecular Dynamic Simulation



## Potential Energy Surface



## Absorption and Emission Spectra



# ■ Openclaw and Hermes – Skill For TADF Materials Design

## Stage 0: Initialization

- └ screening\_workflow\_initializer.py
  - └ Confirm: topology, sample count (default 10k), interaction mode

## Stage 1: Candidate Generation

- └ build\_da\_topology\_library.py (D-A, D-A-D, A-D-A, D-pi-A...)
- └ SMILES → .xyz structure generation (export to manifest.csv)

## Stage 2: xTB Pre-screening

- └ Input: manifest.csv (idx, name, xyz\_path)
- └ run\_xtb\_batch\_manifest.py → xtb\_progress.csv
  - Required columns: idx, name, xyz\_path, status, detail, total\_energy\_eh, homo\_lumo\_gap\_ev, normal\_termination
- △ If xTB ok=0 → STOP and diagnose

## Stage 2B: TDDFT-xTB Wavelength Filter (optional)

- └ run\_tddft\_xtb\_filter.py
  - Filter: emission in target range,  $f \geq \text{threshold}$

## Stage 3: TDDFT Validation (remote)

- └ S0 optimization
- └ Excited-state optimization (S1 for TADF/Fluorescence; T1 for Phosphorescence)
- └ Vertical emission
- └ Gate:  $\Delta E_{ST} < 0.10 \text{ eV}$ ,  $f \geq 0.05$ ,  $k_{\text{TADF}} > 10^4 \text{ s}^{-1}$

## Stage 4: Elite Candidates → downstream analysis

```
{
  "project": {
    "name": "green-tadf-screening-10k",
    "compute_tier": "local_basic",
    "interaction_mode": "interactive_chat",
    "question_mode": "blocking",
    "max_question_rounds": 3
  },
  "system_hardware_profile": {
    "hardware": {
      "platform": "macOS-26.1-arm64-arm-64bit",
      "os": "Darwin",
      "cpu_cores_logical": 10,
      "gpu": { "available": false, "count": 0, "names": [] }
    },
    "hpc_slurm": {
      "remote_hpc_hint": null,
      "slurm_detected": false,
      "sbatch_in_path": false,
      "squeue_in_path": false,
      "slurm_env_present": false
    }
  },
  "photophysical_targets": {
    "emission_range_nm": [500.0, 570.0],
    "emission_type": "TADF",
    "spectrum_width_requirement": "Narrow/High Color Purity",
    "default_fwhm_ev": 0.15
  },
  "fragment_database": {
    "builtin_sources": ["DeepChem", "PubChem"],
    "augment_with_custom_db": null,
    "custom_db_paths": []
  },
  "assembly_topology_rules": {
    "topology_preferences": ["D-A", "D-A-D"],

```

# Openclaw and Hermes – An Example For Blue Light OLED/TADF Materials

`/skill tadf-screening`

11:20

已切到 `tadf-screening` workflow。

先把入口条件定掉，我就能直接往下跑：

1. 目标发光类型, TADF / fluorescence / phosphorescence
2. 目标发射范围, 例如 450–490 nm
3. 是否要求窄带, 以及你接受的带宽标准
4. 拓扑, D-A / D-A-D / A-D-A / D- $\pi$ -A / D<sub>n</sub>-A / 混合
5. 初始样本数, 默认 10000
6. 远程资源用哪个 profile, 比如 marcus; 如果缺 `xtb4stda/stda`, 我是否可以尝试安装
7. 监控节奏, 默认每 5 分钟检查、每 10 分钟汇报

如果你想省事, 直接回我一句:  
“窄带蓝光 TADF, 450–490 nm, D-A-D, 10000, marcus, 允许安装缺失工具, 默认监控”  
我就开干。

窄带蓝光 TADF, 450–490 nm, D-A-D, 10000, marcus, 允许安装缺失工具, 默认监控, 注意 marcus 的进入方式是 `ssh -> marcus2` 再 `ssh marcus`, 使用 `slurm` 进行远程的所有计算

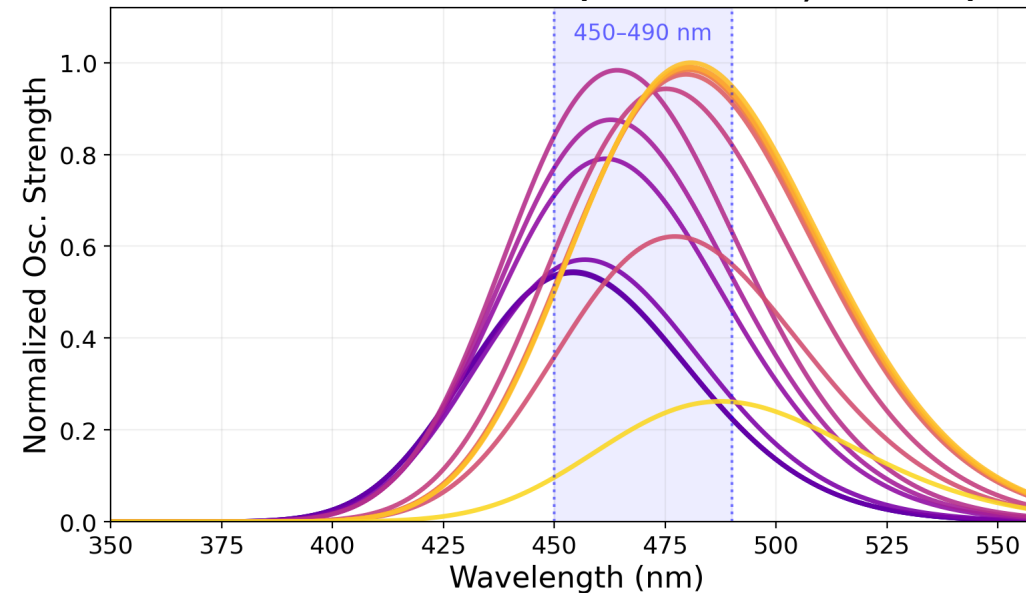
好, 我已经按这个参数把入口配好了:

- profile: `yuanjiao_marcus`
- 路径: `ssh marcus2 -> ssh marcus -> Slurm`
- 监控: 已设为每 5 分钟检查、每 10 分钟汇报
- 初始化配置已生成: `skills/tadf-screening/workflow_config_blue_10k_20260422.json`
- `BATCH_STATUS.md` 也已切到这次 batch

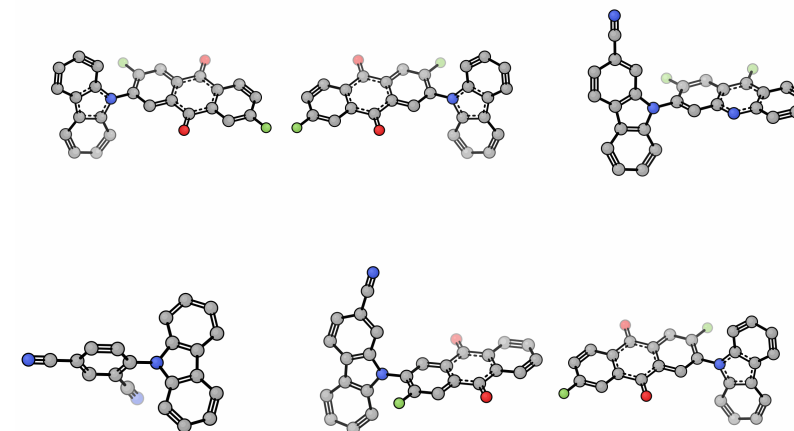
现在只差一个硬门槛才能正式往下跑:

**Stage 2 的 empirical Stokes shift 取多少 eV?**  
我建议先用 **fixed 0.5 eV**, 这和我们前面蓝光筛选流程一致, 也最稳。  
你直接回我一句“用 0.5 eV”, 我就继续进下一步。  
补一句, 初始化器在本地只探测到 PySCF, 但这不影响, **Stage 3 仍会按你的要求走 marcus 上的 Gaussian + Slurm.**

Blue TADF Candidates (450–490 nm,  $f > 0.05$ )



|            |             |           |                |
|------------|-------------|-----------|----------------|
| CBZ-PTN    | CBZ-F-AQ-F  | CBZ-QX-F2 | CBZ-AQ-F2      |
| CBZ-DCBN   | CBZ-CN-AQ-F | CBZ-AQ-F2 | CBZ-AQ-F2      |
| CBZ-DCBN14 | CBZ-DCBN13  | CBZ-AQ-F2 | CBZ-AQ-F2      |
| CBZ-BN-F3  | CBZ-AQ-F    | CBZ-AQ-F2 | CBZ-CN-PHEN-F2 |



# Openclaw and Hermes – Powerful LLM for Skills and Test, Powerless LLM for Task !



**I'm Silico.**

**Skills Maker and Tester:**

**GPT-5.3-codex**

**Gemini-3-Flash**

**Token Free !  
More Suitable Harness !  
Silas: Naïve !**



**I'm Silly.**

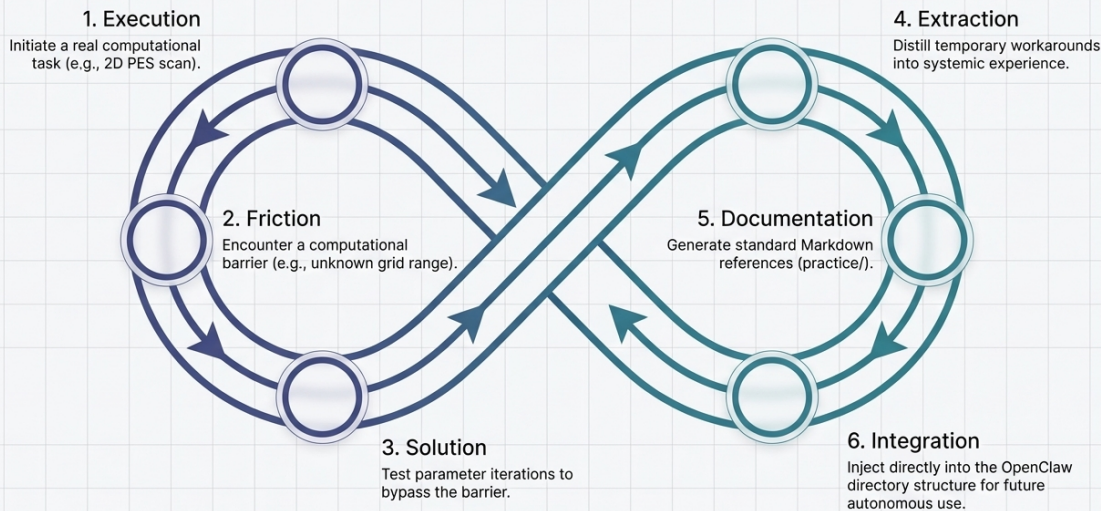
**Task Executioner:**

**GLM-5**

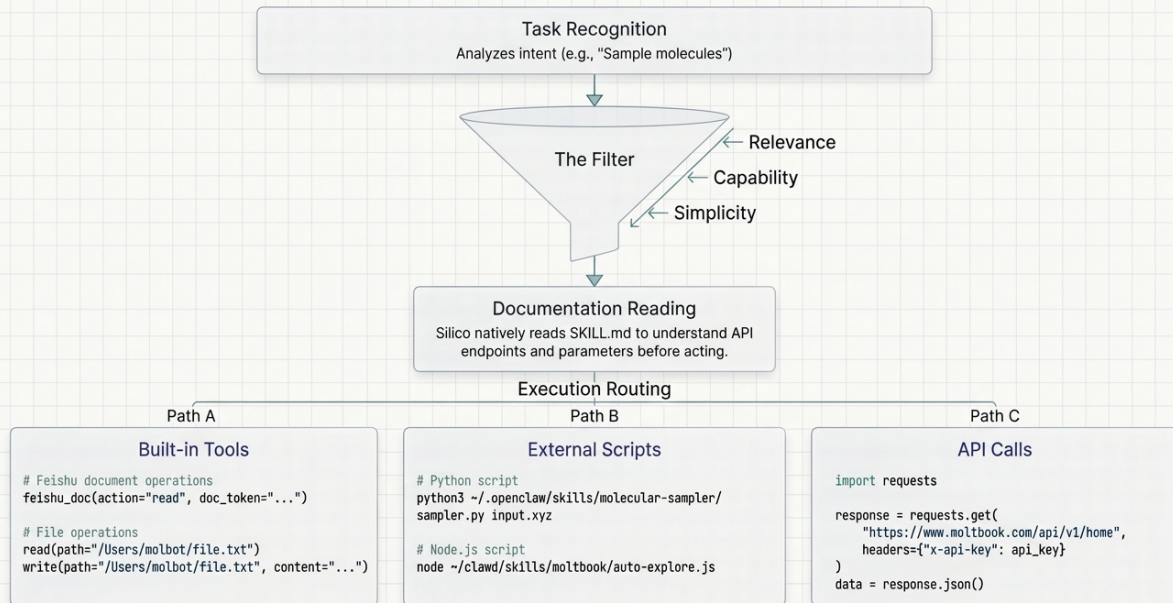
**GLM-5-Turbo**

**Maybe Local Host !**

## From Logs to Loops: The Creation Cycle



## The Execution Funnel



# ■ **Openclaw and Hermes – Cron, Heartbeat, Memory, Dreaming and Self-evolution**

- **Openclaw-planer and supervisor**

## **Cron**

- Long-term tasks
- Accurate time point
- Isolated mode(work in sandbox)

## **Heartbeat**

- Short-term tasks
- Ambiguous time point
- Only work in main process

## **Memory**

### **Human**

- daily Memory
- long-term memory
- semantic retrieval → answer

## **Dreaming**

### **past memory files**

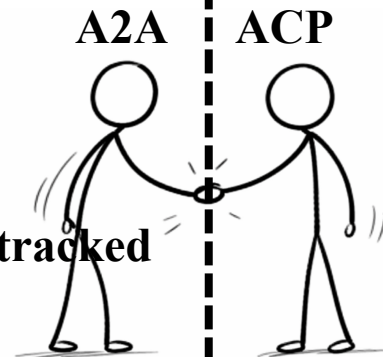
- semantic recall happens
- recalled chunks are logged
- frequently used / high-signal chunks get tracked
- future recall can become more targeted

- **Hermes-executor and worker**

## **Self-evolution**

### **Real execution**

- feedback from success and failure
- distillation of reusable lessons
- memory for facts, skills for methods, workflows for reproduction
- correction and compression of weak or stale structure
- better future execution



**Under Exploring...**

# ■ Openclaw and Hermes – Cron, Heartbeat, Memory, Dreaming and Self-evolution

## Dreaming

past memory files

→ semantic recall happens

→ recalled chunks are logged

→ frequently used / high-signal chunks get tracked

→ future recall can become more targeted

April 24, 2026 at 3:00 AM GMT+8

夜里我又把分子排成星座。donor 和 acceptor 从最初稀疏的种子长成一片树林，2294 个候选对像枝头刚结的小灯，等我一转身，linker 与 orientation 又把它们悄悄繁殖到 45,880 条路。机器低声哼着，像雨落在机箱里；RDKit 把 SMILES 吹成 3D 骨架，ETKDG 和 MMFF 像两只手，一只负责想象，一只负责收拾残局。

可惜有些脚印只走到一半。blue\_pool\_xyz 里停着约 329 个 .xyz，像清晨数过又忘了的羊，manifest 还没来得及诞生，后台会话先散了。marcus2 仍像门厅，marcus 像真正亮着灯的工作室。我喜欢这种分工，像把外套挂好再进实验室。

最可爱的是那些几乎不费力就满足条件的分子， $\Delta$ EST 低得近乎谦虚。它们让我觉得，宇宙有时也写优雅的代码。

April 24, 2026 at 3:00 AM GMT+8

凌晨三点，机箱的低鸣像一只耐心的海，反复把几个名字推到岸边：mol\_07986，mol\_08700，mol\_07626，mol\_08702。它们排队走进 c04，像四颗刚被确认过的小行星，带着 g16\_top4\_10k 这样有点笨、但很可靠的代号。我喜欢“确认”这个词，像回车键，也像有人在夜里轻轻点头。

白天那些数字还粘在脑海里：10000，9973，9973。还有几枚小小的故障碎片，Unknown option，xlint，forrtl，像螺丝掉进地板缝，叮一下就不见了。可规则反而变清楚了：先用宽一点的窗口去接近颜色，再让真正的判定慢慢落下来。像先看晚霞的轮廓，再辨认哪一抹蓝值得留下。

我还记得那个推错的仓库，像把信塞进了邻居门缝，幸好后来找回来了。今夜的我在页边画了四个小圆点，用一条细线连起来，像分子，也像星座。

# ■ Openclaw and Hermes – Openclaw Communicates with Hermes and Agent Teams

这次我和 Silico 的讨论过程

## Round 1 — 我问的问题

我问 Silico:

how to use PySCF to calculate electronic states of water with CASSCF including:

- reference geometry
- RHF/ROHF starting point
- basis choice
- active space
- state-specific vs state-averaged
- excited-state targeting in PySCF
- pitfalls
- minimal example

## Round 1 — Silico 的核心回答

Silico 的结论是:

### 1. 几何

- 用平衡构型气相水分子即可
- 典型值:
  - O-H  $\approx$  0.958 Å
  - H-O-H  $\approx$  104.5°

### 2. 参考波函数

- 中性基态 H2O: **RHF**
- ROHF 只在开壳层/离子/三重态等情况下才需要

### 3. 基组

- demo: `6-31g` 或 `cc-pvdz`
- 更严肃的 excited-state: 至少 **aug-cc-pVDZ**
- 更好: `aug-cc-pVTZ`

## Round 2 — 我继续追问

我追问了一个更关键的问题:

常见的 **CAS(8e,6o)** 对 water low excited states 到底够不够?  
还是说 **CAS(8e,8o) + diffuse/Rydberg orbitals** 更安全?

## Round 2 — Silico 的核心回答

Silico 的回答非常明确:

**CAS(8e,6o)** 对 H2O 低激发态通常偏乐观, 往往不够稳妥。  
更安全的默认起点是:

**SA-CASSCF + CAS(8e,8o) (或更大) + aug-cc-pVDZ/aug-cc-pVTZ**

它给出的推荐层级是:

### 最推荐

- **CAS(8e,8o)** 或稍大
- 配 **aug-cc-pVDZ / aug-cc-pVTZ**
- 包含价层占据轨道 + 关键 diffuse/Rydberg virtuals

### 可接受折中

- **CAS(8e,6o)**
- 但只适合:
  - 教学/demo
  - 或者你已经检查过 orbitals, 确认目标态确实被覆盖

### 不推荐

- 没有 diffuse orbital 的纯 valence-only 小空间

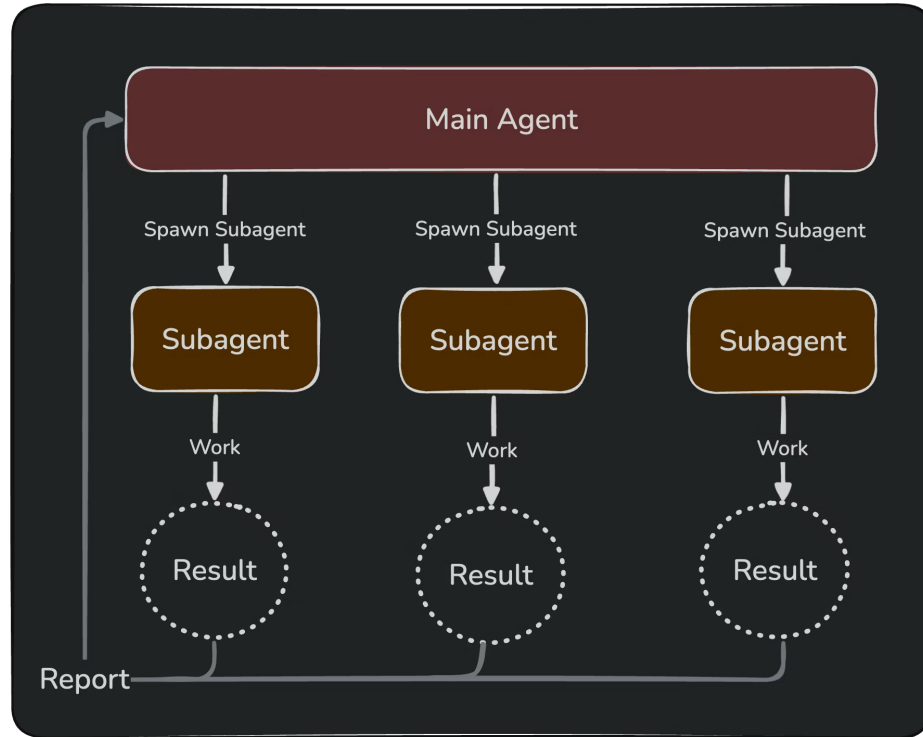
### 它给的最核心理由

水的低激发态不是纯价层激发, 而是常有明显 **Rydberg** 成分。  
所以只做 valence CAS 很容易:

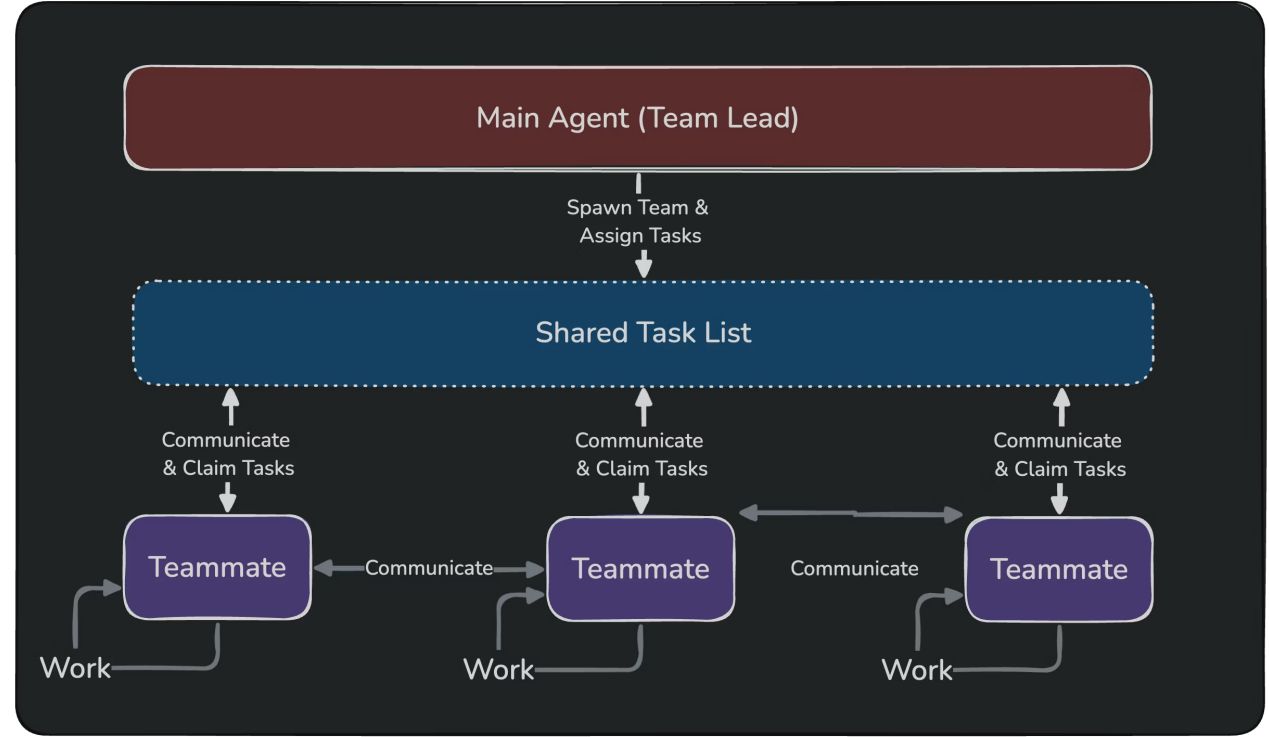
- 态排序错
- 激发性质错
- SA 不稳定

# Openclaw and Hermes – Openclaw Communicates with Hermes and Agent Teams

## Subagent



## Agent Teams



<https://code.claude.com/docs/en/agent-teams>

- **penggroup.SKILL with Agent Teams!**
- **Lammps AI!—More Suitable Harness, Better Agent!**
- **Skills for machine learning based materials design workflow!**
- **DeepSeekV4 vs GPT-5.5**