

# Proposal: Integrating Quantum Chemistry, Molecular Dynamics, and AI for Small Molecule Simulation

LazyingArt

## 1 Overview and Objectives

We propose a comprehensive simulation workflow that **combines quantum chemistry, molecular dynamics (MD), and AI-driven tools** to study a small or biomolecular system. The goal is to **optimize the molecular structure** and **predict its spectra** (e.g. infrared and UV-Vis) with high accuracy, while leveraging automation to streamline the process. Specifically, this plan will:

- Use *first-principles quantum chemistry (DFT)* with **Gaussian** (or a similar package) for **geometry optimization** and **vibrational frequency analysis**, yielding an optimized structure and IR spectra wanglab.hosted.uark.edu wanglab.hosted.uark.edu. Optionally, excited-state calculations (TD-DFT) can predict UV-Vis spectra.
- Use *classical MD simulation* with **LAMMPS** to explore the molecule's conformational space or environment effects, employing appropriate force fields (or ML-based potentials) for efficiency. LAMMPS will run on our high-performance GPU hardware to accelerate sampling docs.lammps.org.
- Use *wavefunction analysis* with **Multiwfn** to gain deeper insight from the quantum chemistry results. Multiwfn can parse Gaussian outputs mattermodeling.stackexchange.com and compute derived properties (bonding analysis, population analysis, spectra plotting, etc.).
- **Automate** the workflow with Python scripts (using libraries like ASE, cclib, or custom scripts). This will allow running Gaussian and LAMMPS calculations, parsing outputs, and calling Multiwfn in batch mode. We will leverage our powerful system ( $2 \times$  RTX 4090 GPUs, 128 GB RAM) for parallel computation and possibly incorporate AI coding assistants to generate or optimize script code.

By integrating these components, we aim to demonstrate an **AI for Science** approach: bridging *first-principles accuracy* and *efficient simulation* through automation. This plan not only yields the target molecule's optimized structure and spectra, but also provides a re-usable **scripted pipeline** for future simulations.

## 2 Methodology and Tools

### 2.1 Quantum Chemistry (Gaussian) for Structure & Spectra

**Tool:** Gaussian (DFT quantum chemistry package) – to be run for geometry optimization and vibrational analysis.

- **Geometry Optimization:** We will perform a full geometry optimization using DFT (for example, B3LYP functional with an appropriate basis set for organic molecules). Gaussian's Opt keyword will adjust atomic coordinates to find a **minimum-energy structure** wanglab.hosted.uark.edu. This ensures we have a stable conformation (stationary point) before computing spectra.

- **Vibrational Frequency Analysis (IR Spectra):** After optimization, a frequency calculation (`Freq` keyword) will be run at the same theory level wanglab.hosted.uark.edu. Gaussian will compute the **force constants and vibrational frequencies**, and also output **IR intensities** for each normal mode wanglab.hosted.uark.edu. These results allow us to simulate an IR spectrum. (If Raman spectra are of interest, the `Freq=Raman` option could be added to compute Raman activities wanglab.hosted.uark.edu.) The frequency job confirms the absence of imaginary frequencies (ensuring a true minimum). We will obtain a list of vibrational modes with their frequencies ( $\text{cm}^{-1}$ ) and IR intensities.
- **UV-Vis Spectra (optional):** For electronic spectra, we can perform a **TD-DFT** calculation on the optimized structure to get excited state energies and oscillator strengths. Gaussian's TD (Time-Dependent DFT) keyword will provide excited-state transitions. This yields data to plot a UV-Vis absorption spectrum (peaks at wavelengths with given oscillator strengths).

*Why Gaussian:* It is a robust first-principles tool that provides reliable optimized geometries and spectral predictions for small molecules. By using DFT, we ensure high accuracy in structure and vibrational frequencies, which is crucial for trustworthy spectra.

## 2.2 Molecular Dynamics (LAMMPS) for Conformational Sampling

**Tool:** LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) – a classical MD engine to simulate the molecule's behavior over time.

- **Force Field Preparation:** We will choose an appropriate force field for the target molecule. For a small organic or biomolecule, this might involve using an all-atom force field (e.g., AMBER or OPLS for biomolecules, GAFF for drug-like molecules). Parameters can be assigned using tools like AmberTools or OpenBabel, generating a LAMMPS-compatible input (data file with masses, charges, bonded parameters, etc.). If an *ML potential* is preferred for higher accuracy, we could employ a pretrained neural network potential (for example, ANI-2x for small organic molecules) via an interface – though classical force fields are more straightforward for a first attempt.
- **Simulation Setup:** We'll set up the molecule in an appropriate environment. For gas-phase or vacuum simulations, a single molecule in a periodic box (with sufficient vacuum padding) will be used. For a biomolecule in solution, we would solvate it in a water box (using e.g. TIP3P water) and possibly add counterions if the molecule is charged. The initial coordinates can be the DFT-optimized structure (for a small molecule) or a known experimental structure (for a biomolecule, e.g. from PDB).
- **Running MD:** LAMMPS will be run to simulate the system's dynamics, using an NVT or NPT ensemble as appropriate (constant temperature dynamics to sample conformations, possibly at room temperature 300 K). We will exploit LAMMPS' parallelism and GPU acceleration to speed up the simulation docs.lammps.org. LAMMPS can be compiled with the GPU package or KOKKOS for CUDA support, allowing our RTX 4090s to significantly accelerate non-bonded force computations docs.lammps.org. We will run a sufficiently long simulation (e.g., several nanoseconds if feasible) to sample the conformational space. The output will be a trajectory of structures over time.
- **MD Analysis:** Using the trajectory, we can analyze structural fluctuations – e.g., RMSD, bond angle variations, etc. If the molecule is flexible, we'll identify distinct low-energy conformers from the trajectory. These conformers could then be re-evaluated with quantum chemistry to see

how they differ in energy or spectra. For example, we might take a few representative snapshots from MD and run single-point energy calculations or quick optimizations in Gaussian to see if multiple minima exist. If the target is a biomolecule (like a peptide), MD will give insight into its stable conformations in solution which a single DFT optimization (gas-phase) might miss.

*Why MD:* Classical MD provides a **time-dependent picture** that complements the static DFT result. It accounts for temperature effects and environmental interactions (especially if solvent or many-body context is included). By using MD, we can verify if the DFT-optimized structure is a global minimum and see how the molecule behaves dynamically. Moreover, MD can generate an **ensemble of structures** for which we might want to predict spectra (yielding, for instance, broadened IR peaks due to conformational averaging).

## 2.3 Wavefunction Analysis (Multiwfn) for Deeper Insights

**Tool:** Multiwfn – a versatile wavefunction analysis program by Tian Lu, compatible with many quantum chemistry outputs.

- **Integration with Gaussian:** After the Gaussian DFT calculation, we will save the **wavefunction output** (e.g., Gaussian checkpoint `.chk` file or formatted checkpoint `.fchk`). Multiwfn can read Gaussian outputs directly mattermodeling.stackexchange.com. In fact, Multiwfn supports almost all major QC programs' outputs (Gaussian, ORCA, GAMESS, etc.) mattermodeling.stackexchange.com. We will use the Gaussian output as input to Multiwfn to perform various analyses.
- **IR Spectrum Plotting:** Multiwfn can take the list of vibrational frequencies and intensities from Gaussian and **plot an IR spectrum** with peak broadening. This provides a visual spectrum to compare with experimental data. (There are tutorials for using Multiwfn to plot IR spectra from Gaussian results; essentially one loads the `.fchk`, selects the vibrational analysis module, and outputs a spectrum plot.)
- **Molecular Orbital and Electronic Analysis:** From the DFT wavefunction, Multiwfn can output molecular orbital information, e.g., HOMO/LUMO energies and compositions. It can perform **orbital composition (population) analysis**, showing contributions of atomic orbitals to each molecular orbital mattermodeling.stackexchange.com. This helps interpret UV-Vis transitions by identifying which orbitals are involved in major excitations.
- **Charge and Bonding Analysis:** Multiwfn offers many population analysis schemes (Mulliken, Hirshfeld, AIM, etc.) to compute atomic charges mattermodeling.stackexchange.com and bond critical points (AIM theory) mattermodeling.stackexchange.com. We can use these to understand the electron distribution in the optimized structure. For example, **Atoms-In-Molecules (QTAIM) analysis** could locate bond critical points and calculate electron density properties, giving insight into bonding character. **Hirshfeld or Mulliken charges** can be obtained to see charge distribution. These analyses deepen our chemical understanding of the molecule beyond just geometry.
- **Spectral Analysis (UV-Vis, PES):** If we performed a TD-DFT calculation, Multiwfn can also help plot the **UV-Vis spectrum** by broadening the transitions. Additionally, Multiwfn can simulate a photoelectron spectrum (PES) using Koopmans' theorem from orbital energies mattermodeling.stackexchange.com, if that is of interest. While PES is not explicitly requested, this demonstrates the breadth of spectral analysis possible.

In summary, Multiwfn acts as a **post-processing toolkit**, taking the high-level results from Gaussian and extracting detailed information and visualizations. It's a bridge between raw compu-

tational output and chemically meaningful data. (It is a command-line program but has interactive menus; we will use it in batch mode for automation as described later.)

## 2.4 Automation and AI Integration (Python Workflow)

To ensure the entire project is **reproducible and automated**, we will develop Python scripts that link all the steps. This is where we leverage coding tools (and potentially AI code assistants like OpenAI Codex or Copilot) to generate and manage simulation scripts.

- **Python as Glue:** Python will serve as the **workflow manager** to run and monitor simulations. We will use Python for tasks such as:
  - Preparing input files (Gaussian .gjf files, LAMMPS input scripts, Multiwfn control files).
  - Submitting jobs and waiting for completion.
  - Parsing output files to extract key results (energies, frequencies, etc.).
  - Calling analysis functions and possibly plotting results (using libraries like Matplotlib or Seaborn for spectra visualization).
- **Atomic Simulation Environment (ASE):** We can utilize ASE, an open-source Python toolkit for setting up and analyzing simulations abacus.deepmodeling.com. ASE has interfaces for many calculators; notably, it can run Gaussian if properly configured (by setting environment variables and using an ASE Gaussian calculator). ASE can also call LAMMPS through an interface or simply by writing LAMMPS input files and executing them. Using ASE:
  - We can create an `Atoms` object for our molecule, optimize it using a chosen calculator. (If Gaussian is not directly supported, we could use ASE with a simpler calculator for quick tasks, or just use it to manage coordinates.)
  - ASE can read Gaussian output geometries and convert them into other formats, which helps to pass the optimized structure to LAMMPS easily mlatom.com.
  - ASE's built-in optimization algorithms or workflows might assist in coordinating DFT and MD steps, though for our plan we will likely manage steps explicitly for clarity.
- **cclib for Parsing:** The **cclib library** in Python is very useful for parsing quantum chemistry outputs cclib.github.io. We will use `cclib` to parse the Gaussian log file for results such as the final optimized coordinates, the frequencies and IR intensities, and any orbital energies or TD-DFT results. `cclib` supports Gaussian output (including Gaussian16) and can extract vibrational modes, energies, etc., into Python data structures cclib.github.io. This will allow our script to automatically retrieve the IR spectrum data and perhaps compare it with MD-based results. For example, using `ccread` to parse `gaussian.log` yields a data object where `data.vibfreqs` and `data.vibirs` contain the frequencies and IR intensities respectively cclib.github.io. Similarly, `data.etenergies` and `data.etoscs` would hold excited state energies and oscillator strengths if TD-DFT was run.
- **Running LAMMPS via Python:** There are a couple of ways to run LAMMPS through Python. LAMMPS provides a Python library interface (`lammps.py`) which can send commands to the LAMMPS engine in-memory docs.lammps.org. Alternatively, one can simply have Python write a LAMMPS input script and call the LAMMPS executable via `subprocess`. We will likely do the latter for simplicity. The Python script can generate the LAMMPS input (text file specifying force field, integration parameters, etc.), then execute LAMMPS with that input (e.g., `subprocess.run(["lmp", "-in", "mysim.in"])`). Because LAMMPS is designed for

command-line operation, this approach is straightforward. After the run, the script can parse the LAMMPS log or data (using Python or even `MDTraj/pytraj` libraries) to extract, say, the final structure or some property of interest.

- **Automating Multiwfn:** Multiwfn is interactive by default, but it **supports batch mode** by reading commands from a text file (the Multiwfn manual describes a “task file” approach). We will create a text file with the sequence of menu options corresponding to the analysis we want (for example, to output an IR spectrum: load fchk, choose spectral analysis, output spectrum data). The Python script can write this Multiwfn command file, then call Multiwfn in batch mode (e.g., `subprocess.run(["Multiwfn", "file.fchk", "<", "commands.txt"])`). This will execute Multiwfn without user intervention and produce the desired output (like an IR spectrum XY data or an image). Given Multiwfn runs on both Windows and Linux and we have a Linux environment, we will compile or use the Linux version of Multiwfn [mattermodeling.stackexchange.com](https://mattermodeling.stackexchange.com).
- **Leveraging AI for Coding:** Throughout the development of these scripts, we can use AI coding assistants (like OpenAI’s Codex or ChatGPT) to help generate boilerplate code, troubleshoot errors, or optimize performance. For instance, AI can assist in writing a parsing routine or suggesting the correct syntax for using the LAMMPS Python interface. This will speed up development and ensure robustness. (All code will be reviewed and tested manually, but AI tools can significantly reduce the effort in scripting repetitive tasks.)
- **Workflow Structure:** We will likely implement the above in a **modular fashion**. For example:
  - `optimize_and_freq.py`: runs Gaussian optimization + frequency, using `cclib` to parse results.
  - `run_md.py`: sets up and runs LAMMPS MD, possibly returning a few sampled structures.
  - `analyze_wavefunction.py`: uses Multiwfn (called via Python) to analyze the outputs.
  - `main_workflow.py`: orchestrates the entire sequence (calls the above modules in order and handles data flow between them).By breaking it down, each part can be developed and tested independently (and AI helpers can be applied to each part). Ultimately, a **master script or Jupyter notebook** will run the full pipeline end-to-end.

## 2.5 System and Resource Considerations

- **Hardware Utilization:** Our dual NVIDIA RTX 4090 GPUs will be instrumental in speeding up **MD simulations** and any ML computations. LAMMPS GPU acceleration will offload force calculations to the GPUs [docs.lammps.org](https://docs.lammps.org), enabling longer or larger simulations within a given time. If we use machine learning potentials (like a neural network potential via TorchANI or similar), those computations will also be GPU-accelerated by design. The 128 GB RAM and 16 TB SSD ensure we can handle memory-intensive quantum calculations and store large trajectory files or volumetric data from wavefunction analysis. Gaussian will run on CPU (potentially multi-threaded across our CPU cores), while MD and ML will leverage GPUs – achieving a balanced use of our workstation.
- **Software Setup:** We need to have Gaussian installed (with appropriate license) on the machine, along with LAMMPS (which is open-source). Multiwfn is free; we will download the source and compile it on Linux, ensuring to link against required libraries (Lapack, FFTW as needed). Python environment should have `ase`, `cclib`, `numpy/pandas`, and possibly `mdtraj` for MD analysis, plus plotting libraries. We will create a conda environment to manage these tools. If

Gaussian is not available, we could substitute with ORCA (free) for DFT and it would still output files compatible with Multiwfn mattermodeling.stackexchange.com and cclib. However, using Gaussian as requested provides a familiar baseline.

- **AI/ML Enhancements (optional):** Given the project’s theme, we may incorporate some *machine learning* enhancements:
  - Use a **pre-trained neural network potential** for the molecule to accelerate conformational searches. For example, **TorchANI** provides ANI-2x, a neural potential trained to reproduce DFT energies for organic molecules (with elements H, C, N, O, etc.). We can use TorchANI via Python to quickly optimize the geometry or run a short MD, obtaining an initial guess for Gaussian. This **ML-FF (machine learning force field)** can bridge the accuracy-speed gap, providing DFT-approximate results in a fraction of the time cclib.github.io. Once a good conformation is found by ANI, we refine it with full DFT. (Our GPUs can run TorchANI extremely fast since it’s PyTorch-based acs.figshare.com/gganbumarketplace.com).
  - If time permits and data is available, we might *train a custom ML model* on a few DFT calculations (for instance, use active learning as in DeepMD or PhysNet frameworks) to see how AI can learn the potential energy surface of our target molecule. However, given the scope, using an existing model like ANI is the most straightforward path for ML integration.
  - We will also use **AI for data analysis** where suitable. For example, using regression or classification models to correlate MD features with spectral shifts (if we simulate spectra at different snapshots), or simply using AI to help interpret vibrational modes (though this is more speculative).

These enhancements align with the new paradigm of *AI for Science*, where **machine learning force fields (ML-FFs)** can achieve near first-principles accuracy at much lower cost. Recent developments show that ML-FFs can bridge the accuracy-efficiency gap between ab initio methods and classical force fields, enabling large-scale simulations with quantum-level precision. Our approach takes advantage of these advancements by optionally incorporating ML models into the simulation pipeline.

### 3 Implementation Plan (Step-by-Step)

Below is a detailed plan of execution, broken into phases. Each phase produces outputs that feed into the next.

#### Phase 1: Initial Setup and System Definition

- **Choose the Target Molecule:** Decide on a specific molecule or system to study. Since we want **structure optimization and spectral analysis**, a single molecule that is experimentally relevant would be ideal (e.g., a small drug molecule, an amino acid, or a small peptide). Ensure the system is of a size manageable by DFT (generally < 100 atoms for comfortable optimization). *For concreteness, assume we select an amino acid like alanine or a small peptide di-alanine.* This gives a flavor of a biomolecule (with peptide bond) but remains small. It has interesting conformational possibilities (rotamers) and measurable IR spectrum (amide vibrational bands).
- **Software Installation and Testing:**
  - Install Gaussian on the local machine (if not already). Verify that a test job (e.g., optimizing water) runs successfully.

- Install LAMMPS (with GPU support). We will compile LAMMPS with the GPU package enabled or use a pre-built binary for CUDA. Test run a small MD (like argon gas) to ensure GPU acceleration works.
- Compile or install Multiwfn. Ensure required libraries are present. After building, test by running an example analysis.
- Set up the Python environment with needed libraries: `ase`, `cclib`, `numpy`, `scipy`, `matplotlib`, `mdtraj`, `pandas`. Write a short test script to confirm each works.
- (Optional) Install TorchANI or other ML potential libraries if we plan to use them. Test loading a pretrained model and computing energy for a known molecule (e.g., benzene).

- **Obtain Initial Structure:**

- Draw the molecule in a builder (Avogadro, GaussView, etc.) and save as MOL or PDB; or obtain from PDB/literature.
- Do a quick pre-optimization (MMFF/OpenBabel or ASE MM) to aid DFT convergence.
- **Define Project Directory Structure:** Organize directories for each phase: e.g., `/project/geometry/`, `/project/md/`, `/project/analysis/`. Create a git repo to track scripts and settings.

## Phase 2: Quantum Chemistry Optimization & Spectra

- **Prepare Gaussian Input:** Write a `.gjf` including charge/multiplicity, starting coordinates, and route section (e.g., `# B3LYP/6-31G** Opt Freq` plus options like `EmpiricalDispersion=GD3`, `SCRF` if solvent). Combining `Opt Freq` optimizes then computes frequencies in one run `wanglab.hosted.uark.edu`.
- **Run Gaussian Calculation:** Execute and monitor for convergence. Ensure no imaginary frequencies; refine if needed.
- **Parse Results with cclib:** Example:

```
import cclib
data = cclib.ccread("alanine_opt_freq.log")
energies = data.scfenergies
final_energy = energies[-1]
frequencies = data.vibfreqs
ir_intens = data.vibirs
coords = data.atomcoords[-1]
atoms = data.atomnos
```

Save optimized structure, verify  $3N-6$  modes and no negatives.

- **Generate IR Spectrum Plot:** Broaden lines (e.g.,  $10\text{ cm}^{-1}$  HWHM) and plot. Or run Multiwfn to produce a polished spectrum.
- **(Optional) UV-Vis Calculation:** TD-DFT (`TD(NStates=N)`) on the optimized structure; parse `data.etenergies`, `data.etoscs` and simulate a UV-Vis spectrum.

## Phase 3: Classical Molecular Dynamics Simulation

- **Force Field Parameterization:** Use AMBER/GAFF (with charges) or similar; produce a LAMMPS data file. If solvated, prepare water box and ions.

- **Prepare LAMMPS Input:** Units (**real**), read data, choose pair styles (PME if periodic), thermostat (NVT/NPT), timestep (e.g., 1 fs), length (e.g., 1–2 ns), outputs (trajectory + thermo).
- **Run with GPU:** Use GPU or KOKKOS acceleration per docs to exploit RTX 4090 performance.
- **Monitor and Analyze:** Check stability; analyze RMSD, dihedrals, cluster conformers; extract representative structures.
- **(Optional) QM/MM or Refinement:** Single-point DFT on snapshots with implicit solvent to assess spectral shifts.

#### Phase 4: Multiwfn Analysis and Validation

- Load `.fchk` in Multiwfn (batch mode) to compute Mulliken/Hirshfeld charges, HOMO/LUMO, bond indices, QTAIM, etc.
- **Spectral Comparison:** Compare IR/UV-Vis with experiment; apply typical frequency scaling (~0.98) if desired; consider conformer averaging.
- **Consistency Checks:** ESP maps, polarizabilities, and other derived properties to interpret interactions and spectral features.

#### Phase 5: Consolidation, Automation, and Documentation

- Integrate scripts into a robust end-to-end pipeline with error handling and sanity checks.
- Tune parameters for efficiency (basis sets, MD lengths, temperatures).
- Use AI assistant(s) to review code for robustness and clarity.
- Synthesize results: optimized geometry, vibrational table, IR/UV plots, MO visualizations, MD analyses, and key properties.
- **Timeline:** Days 1–2 setup; 3–5 Gaussian; 6–8 MD; 9–10 Multiwfn; 11–14 automation; 15+ analysis/documentation.

## 4 Expected Outcomes and Significance

By following this plan, we will achieve a **comprehensive simulation study** of the chosen molecule:

- **Verified Optimized Structure:** DFT minimum providing a trustworthy reference.
- **Predicted IR/UV-Vis Spectra:** Directly comparable with experiment; informs accuracy and needed improvements.
- **Conformational Insight:** MD reveals finite-temperature behavior and conformer populations.
- **Wavefunction-Derived Data:** Charges, MO energies/compositions, bonding indicators from Multiwfn.
- **Automated Workflow:** Reusable pipeline for future studies, embodying best practices.
- **Use of AI in R&D:** Demonstrates productivity boost via AI coding assistance and optional ML potentials.

In conclusion, integrating **Gaussian** (quantum accuracy), **LAMMPS** (efficient sampling), and **Multiwfn** (in-depth analysis) under an automated, AI-augmented Python workflow aligns with the *AI for Science* paradigm. This project both achieves immediate goals (structure and spectra) and establishes a generalizable approach for future computational studies.

## 5 References and Supporting Resources

- Gaussian 16 User Manual – **Geometry Optimization and Frequency Calculation**: wanglab.hosted.uark.edu, wanglab.hosted.uark.edu.
- Gaussian 16 Manual – **Vibrational Analysis**: harmonic IR intensities and Raman options; recommended to use same level for Opt and Freq.
- LAMMPS Documentation – **Overview and GPU Acceleration**: docs.lammps.org, docs.lammps.org. Python library interface also available.
- Multiwfn – **Wavefunction Analysis**: mattermodeling.stackexchange.com (broad support; AIM, populations, spectra, etc.).
- cclib – **Parsing QC Outputs**: cclib.github.io; supports Gaussian 09/16; extracts vibrational and TD-DFT data.
- TorchANI and ML Potentials: pretrained ANI models (PyTorch) for fast PES; see gganbumarketplace.com; performance details at acs.figshare.com.
- ASE (Atomic Simulation Environment): high-level Python interfaces to codes and formats; abacus.deepmodeling.com.