Student Project Proposal

Agent-Driven Adaptive Multiscale Simulation

Master area	AI for Drug Discovery; molecular simulation; multiscale modelling; active learning
Project type	Computational prototype with Python implementation and simulation analysis
Duration	2-3 months
Supervision	One tutor or co-tutor; weekly or bi-weekly progress meetings
Recommended level	Basic Python and numerical modelling; basic knowledge of molecular simulation is useful
Main output	Documented workflow/notebook, short technical report and final presentation
Possible data	Synthetic 2D agent systems; optional tutor-provided local MD examples or simplified molecular configurations

Short description. The project asks the student to build a simplified adaptive multiscale simulation prototype in which coarse molecular agents can keep, merge, or split their representation according to local order, uncertainty and computational need.

1. Background and rationale

Many systems relevant to molecular modelling, formulation science and drug delivery show strong spatial and temporal heterogeneity. Some regions are stable and can be described coarsely, whereas others undergo rapid rearrangements, coalescence, local disorder or rare events that require a finer description.

Classical multiscale approaches often rely on fixed coarse-graining rules or predefined regions of high resolution. This is efficient, but it may become inadequate when the relevant region changes over time. The central idea of this project is that the simulation should adapt its own resolution according to the evolving local state of the system.

This concept is relevant to AI for drug discovery because adaptive simulation, uncertainty-driven refinement and local surrogate models can reduce computational cost while preserving information where it matters most.

2. Project objective

The objective of the project is to implement a minimal agent-driven adaptive simulation framework, initially in two dimensions, in which agents represent coarse molecular regions and dynamically decide whether to remain unchanged, merge into larger entities, or split into finer units.

The project should remain a proof of concept. It is not expected to deliver a production molecular dynamics engine. The expected outcome is a clear and testable prototype that illustrates the computational logic of adaptive resolution, local uncertainty and optional micro-solver refinement.

3. Specific aims

- Define the state variables of each agent, including position, size, composition, orientation, local uncertainty and time step.
- Implement neighborhood detection and simple effective interaction rules based on local free-energy or PMF-like terms.
- Design merge, keep and split rules driven by local order, overlap, composition and uncertainty.
- Introduce an asynchronous scheduler in which stable and unstable agents may evolve with different time steps.
- Create validation metrics such as coverage, overlap, composition conservation, order and computational cost.
- Prepare a simple interface for future on-demand micro-solvers based on short MD or DFT calculations.

4. Proposed methodology

4.1 Minimal system representation

The student will implement a simplified 2D periodic box containing agents with variable radius and local properties. Each agent will store its coarse state, neighborhood, uncertainty score and current update time.

4.2 Effective interactions and free-energy bookkeeping

Instead of defining a full atomistic force field, the prototype will use simple effective interactions or PMF-like terms. These may include attraction, repulsion, alignment and composition-dependent interaction terms.

4.3 Adaptive merge and split policies

Local rules will decide when neighboring agents should merge and when large agents should split. The rules should be transparent and interpretable rather than highly optimized.

4.4 Uncertainty-driven refinement

A local uncertainty metric will be computed from fluctuations, overlap, instability or disagreement between simple surrogate estimates. When uncertainty increases, the simulation may reduce the time step, split an agent or request a micro-solver update.

4.5 Visualization and validation

The student will produce visual outputs showing how the adaptive representation evolves. Quantitative metrics will be used to verify that the simulation remains stable and conserves relevant quantities.

5. Indicative work plan

Period	Main task	Expected output
Weeks 1-2	Model definition and repository setup	Design document, data structures and minimal visualization of an initial agent population.
Weeks 3-4	2D simulation prototype	Basic update loop with neighborhoods, effective interactions and periodic boundary handling.
Weeks 5-6	Merge/split rules and uncertainty	Adaptive coarse-graining rules and first validation plots.
Weeks 7-8	Asynchronous scheduling	Event queue or agent-specific time-step mechanism.
Weeks 9-10	Micro-solver placeholder and metrics	Prototype interface for local refinement and quantitative validation metrics.
Weeks 11-12	Final integration and reporting	Final notebook/code, technical report and presentation.

6. Expected deliverables

- A documented Python prototype of the adaptive agent simulation.
- A short technical report describing model assumptions, rules and validation results.
- Plots or animations showing merge, split and adaptive refinement events.
- Quantitative metrics for stability, overlap, coverage, composition conservation and computational cost.
- A final presentation of approximately 10-15 minutes.

7. Skills acquired by the student

- Design of simplified multiscale simulation models.
- Implementation of agent-based computational systems in Python.
- Use of uncertainty indicators to guide adaptive computation.
- Development of reproducible simulation metrics and visual diagnostics.
- Critical assessment of the relationship between coarse-graining, accuracy and computational cost.

8. Suggested software and resources

The project can be implemented with NumPy, pandas, scipy, matplotlib and optionally numba for acceleration. NetworkX may be useful for neighborhood and connectivity analysis.

The tutor should provide an initial conceptual example and define whether the student should remain at the synthetic 2D prototype level or attempt a small connection to local MD data.

9. Optional extensions

Possible extensions include testing different uncertainty metrics, comparing synchronous and asynchronous integration, adding simple learning-based rules for merge/split decisions, or connecting the prototype to a very small molecular dynamics example.

10. Evaluation criteria

Criterion	What should be assessed
Scientific clarity	The scientific question, assumptions and expected use case are clearly stated.
Technical implementation	The code is readable, documented and reproducible on the selected data.
AI relevance	The project produces representations, models or workflows that are suitable for AI-based analysis.
Interpretability	The student explains what the results mean in a drug discovery or molecular science context.
Critical discussion	The final report discusses limits, uncertainty and possible future extensions.

Note for tutors. The minimal successful version is a stable and interpretable 2D prototype. Coupling to real MD or DFT calculations should be treated as an optional extension for stronger students.