

DynaMate: An Autonomous Agent for Protein-Ligand Molecular Dynamics Simulations

Cassandra Masschelein

cassandramasschelein@gmail.com

EPFL, Valence Labs <https://orcid.org/0009-0001-7216-6320>

Salomé Guilbert

EPFL

Jeremy Goumaz

EPFL <https://orcid.org/0009-0004-0637-2881>

Bohdan Naida

EPFL

Ursula Rothlisberger

Ecole Polytechnique Fédérale de Lausanne <https://orcid.org/0000-0002-1704-8591>

Philippe Schwaller

EPFL, Switzerland <https://orcid.org/0000-0003-3046-6576>

Article

Keywords: language models, molecular dynamics, protein dynamics, agentic, binding free energy, automation

Posted Date: August 27th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10243410/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: There is **NO** Competing Interest.

DynaMate : An Autonomous Agent for Protein-Ligand Molecular Dynamics Simulations

Salomé Guilbert^{1†}, Cassandra Masschelein^{1†}, Jeremy Goumaz¹,
Bohdan Naida¹, Ursula Rothlisberger^{1*}, Philippe Schwaller^{1,2*}

¹Ecole Polytechnique Fédérale de Lausanne (EPFL), Rte Cantonale,
Lausanne, 1015, Vaud, Switzerland.

²National Centre of Competence in Research (NCCR) Catalysis.

*Corresponding author(s). E-mail(s): ursula.roethlisberger@epfl.ch;
philippe.schwaller@epfl.ch;

Contributing authors: salome.guilbert@epfl.ch;
cassandra.masschelein@epfl.ch; jeremy.goumaz@epfl.ch;
bohdan.naida@epfl.ch;

[†]These authors contributed equally to this work.

Abstract

Force field-based molecular dynamics (MD) simulations are indispensable for probing the structure, dynamics, and functions of biomolecular systems. Despite their broad utility in drug discovery and protein engineering, the technical complexity of a MD setup remains a major barrier for widespread and efficient usage. Agentic LLMs have demonstrated their capacity to autonomously execute multi-step scientific processes. Despite this progress, these systems have not been used to fully automate protein-ligand MD workflows. Here, we present DynaMate, a modular multi-agent framework that autonomously designs and executes complete MD workflows for both protein and protein-ligand systems, and offers free energy binding affinity calculations with the MM/PB(GB)SA method. The framework integrates dynamic tool use, web search, literature query via PaperQA, and a self-correcting behavior. We evaluate its performance first across fifteen benchmark systems of varying complexity, assessing success rate, efficiency, and adaptability. Second, we confirm its robustness by achieving 67% successful executions of MD simulations on a dataset of 827 diverse protein and protein-ligand systems. Overall, DynaMate reduces the typical multi-hour setup time to minutes and autonomously resolves common runtime errors through

iterative reasoning. This automated framework paves the way toward standardized and scalable molecular modeling pipelines for biomolecular and drug design applications.

Keywords: language models, molecular dynamics, protein dynamics, agentic, binding free energy, automation

1 Introduction

Molecular dynamics (MD) simulations are a cornerstone of computational chemistry and biophysics, enabling atomistic modeling of molecular interactions and conformational changes over time[1–4]. Specifically, force field-based MD simulations provide crucial insights into the structure, stability, dynamics, and function of biomolecules such as proteins, nucleic acids, and membranes, and have become indispensable tools in areas such as drug design and protein engineering. Further, conformational sampling of protein-ligand complexes through MD simulations can be used to approximate binding free energies using the MM/PB(GB)SA method based on molecular mechanics (MM) with Poisson-Boltzmann (PB)[5, 6] or generalized Born (GB)[7, 8], and surface-area (SA) calculations[9]. One frequent application of these methods is to rank a large number of structurally similar or congeneric binding complexes, since they offer reasonable accuracies at a reduced computational cost, compared to rigorous free-energy methods[10].

In practice, MD simulations are typically performed using specialized engines such as GROMACS[11], OpenMM[12], Amber[13], or CHARMM[14]. However, preparing a molecular system for simulation remains one of the most error-prone and labor-intensive steps in the workflow[15]. After retrieving a structure from a database such as the Protein Data Bank (PDB)[16], users must perform a series of preparation steps. First, the structure must be cleaned by choosing specific conformational states when several are available, adding unresolved amino acids, removing co-crystallized crystallization agents, or mutating back non-native mutations introduced for crystallization. Then, the protein chain termini should be capped with acetyl (ACE) and *N*-methyl (NME) groups. The next step is non-trivial and consists in correctly protonating the protein amino acids and the ligand at the desired pH. Finally, the structure should be solvated in aqueous solution with physiological ionic strength. These preprocessing steps are followed by parameter selection (e.g., choice of force field, temperature, thermostat, barostat) and multi-stage thermal equilibration prior to production runs. While protein parameterization is relatively straightforward due to mature force fields, handling small organic molecules, such as ligands or cofactors, can be substantially more complex. Given these challenges, automation offers clear benefits in reducing setup time, improving reproducibility, and enabling large-scale exploration of molecular systems. Several frameworks have sought to automate MD simulations, including CHAPERONg[17] and PyAutoFEP[18], which facilitate simulation setup and analysis or integrate free-energy perturbation (FEP) workflows[19, 20]. However, these pipelines remain rigid and domain-specific, and extending them to new systems or

simulation engines often requires significant manual modification and expert knowledge. Additionally, as they are purely scripted, they cannot actively correct errors on their own.

Recent advances in large language models (LLMs)[21–23] have introduced new possibilities for scientific automation. LLMs can interpret natural language instructions, reason about domain-specific constraints, and orchestrate external tools, offering a flexible control layer that can adapt to new workflows and dynamically recover from errors. The emergence of agentic LLMs[24–26] has demonstrated their capacity to autonomously execute multi-step scientific processes, including early efforts in MD workflow automation through MDCrow[27] and NAMD-Agent[28]. Both systems successfully automate protein-only simulations using OpenMM or CHARMM-GUI respectively, and can handle basic preprocessing and simulation stages. Yet, they remain limited in scope, unable to process protein-ligand systems, perform adaptive recovery from simulation errors, or generalize across simulation platforms. There have also been efforts on text-to-code generation and fine-tuning of language models for the task with MDAgent[29], which uses LAMMPS[30] and a fine-tuned LLM on an internally curated dataset, albeit for the purpose of obtaining material thermodynamic parameters.

In this work, we introduce DynaMate, a modular multi-agent framework that autonomously designs and executes complete MD workflows for both protein and protein-ligand systems, with the option to perform free energy binding affinity calculations with the MM/PB(GB)SA method[20]. Unlike prior systems, DynaMate separates high-level reasoning (e.g., parameter choice, structure retrieval) from low-level execution (e.g., simulation setup, file handling). This design enables dynamic tool use, retrieval-augmented parameter selection, and a self-correcting behavior during simulation runs. By combining domain-aware reasoning with multi-agent coordination[26, 31], DynaMate bridges natural-language goal specification and robust computational execution in molecular simulation.

Supplementary Table S1 summarizes key differences between existing agentic MD frameworks in the biology domain and DynaMate. Our system uniquely supports protein-ligand systems, integrates retrieval from external databases (web search as well as literature search with PaperQA[25, 32, 33]), and implements adaptive tool selection and error correction, thus significantly extending the capabilities of current MD automation pipelines.

2 Results

2.1 System Architecture of DynaMate

DynaMate is a multi-agent architecture designed to automate the workflow for running MD simulations of protein-ligand systems (Figure 1). DynaMate is comprised of a planner (**PrepAgent**), a worker (**MDAgent**), and an analyzer (**AnalyzerAgent**), and each agent is equipped with a set of tools and their descriptions. After being prompted by the user, a plan is passed to the worker agent from the planner containing the required steps for the procedure as well as the system information such as the path to the PDB structure, the run directory, the name of the ligand if involved, and

116 the desired simulation temperature, pressure and duration. The plan is updated and
117 reinforced in the agent-specific conversation history in order to prevent confusion and
118 loss of details. Agents have access to their action traces and previous messages (or
119 a summary). Action traces include whether tool calls failed and either the full error
120 message or if the tool call proceeded successfully, a message summarizing the output.

121 All agents also have access to the run directory file structure containing the fetched
122 (or provided) PDB as well as the `.mdp` (molecular dynamics parameters) files. This
123 directory will grow as simulation files are created, and the respective agent is able to
124 edit, read, or write anything in the run directory. This, together with the provided
125 tools, allows the LLM to reason, plan, act, and problem solve in a complex setting
126 where multi-step actions are required.

127 If DynaMate fails to complete the entire pipeline, the program exits. The
128 `AnalyzerAgent` provides a post-analysis of the steps completed, any errors encountered
129 during the pipeline, and any suggestions for further investigation or possible
130 error correction. If the MD simulation ran successfully, this analysis includes the monitoring
131 of the protein system temperature, density and energy during equilibration, as
132 well as a root mean square deviation (RMSD), root mean square fluctuation (RMSF),
133 radius of gyration, and hydrogen bonding analysis. DynaMate also offers the ability
134 to run binding free energy calculations in the event of a successful simulation and the
135 presence of a ligand.

136 Throughout each workflow, the decisions of DynaMate are recorded, all created files
137 are saved, and all potential error messages from the tools or software are logged. This
138 allows users to review decisions made by the agent as well as inspect the generated files.
139 Human experts can then verify the accuracy of the workflow and determine whether
140 the appropriate error correction steps were taken.

141 2.2 Performance on MD simulations of protein-ligand 142 complexes: workflow success rate and efficiency

143 The performance and generalizability of DynaMate was evaluated across fifteen distinct
144 simulation setups (further detailed in Supplementary Table S2) spanning diverse
145 protein families and ligand chemotypes. Five of these systems are well-established
146 protein-ligand complexes that are widely used in the literature for benchmarking
147 MD and free energy calculation methods (lysozyme, β -trypsin, tyrosine kinase and
148 bromodomain-containing proteins with PDB IDs: 3HTB[34], 3PTB[35], 4GIH[36],
149 4W52[37] and 5UEZ[38]). We also evaluate the performance on multi-chain proteins
150 (lysozyme, insulin, hemoglobin and carbonic anhydrase proteins with PDB IDs:
151 1AKI[39], 4FKA[40], 1A3N[41], 2CBA[42] and 2VVB[43]). The final five systems added
152 show expected errors and limitations of the tools provided to the LLMs (1U6Q[44],
153 1C5S[44], 2RBQ[45], 3SQ6[46] and 5DQ1[47]). Indeed, as robust and general as Dyna-
154 Mate is, the provided tools can fail for some PDB files. For these biological systems, a
155 deep understanding of the simulation failure mode and chemical or biological reasoning
156 is required to solve the encountered errors. Therefore, these failure modes provide
157 useful case studies to assess the error correction ability of different LLMs.

158 The agent’s success was measured by its ability to correctly execute the tools
159 required for the simulation to succeed, and editing the appropriate files when necessary.

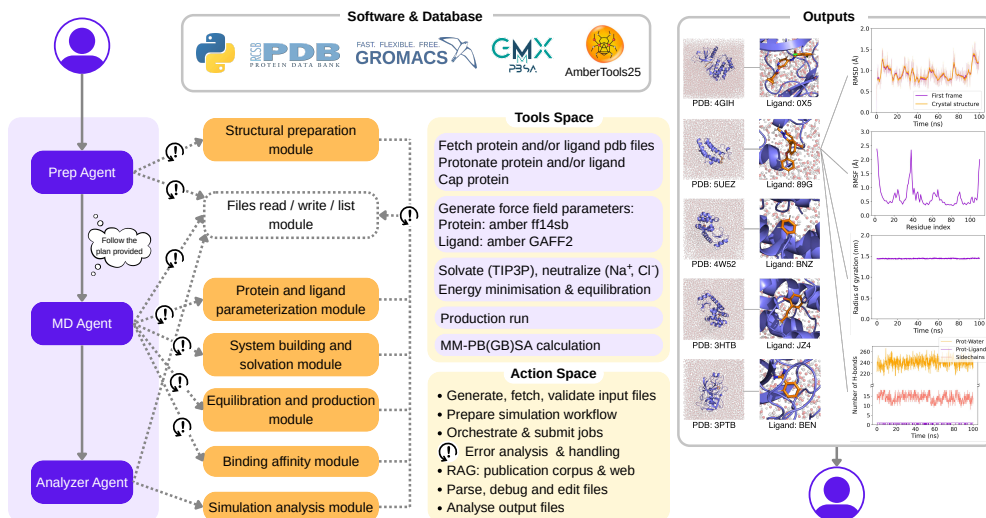


Fig. 1 Overview of the DynaMate framework. The **PrepAgent** constructs a context-aware simulation plan, the **MDAgent** executes it, and the **AnalyzerAgent** interprets the resulting trajectories. All agents use error-corrective reasoning. Representative output plots shown for the 5UEZ system: RMSD, RMSF, radius of gyration, and hydrogen-bond analysis.

We define “workflow success rate” as the percentage of (non-empty) files produced over the total number of files that should have been created for the simulation to successfully complete. All the created files were manually inspected to confirm that they were correct. To achieve a 100% success rate, the protein and optional ligand should be correctly protonated at physiological pH, and the protein chain termini should be capped with acetyl and *N*-methyl groups. All systems should also be correctly solvated, neutralized with ions, energy minimized, and equilibrated with 100 ps NVT and 100 ps NPT simulations. Finally, the LLM should analyze the output RMSD, RMSF, and radius of gyration files to confirm appropriate equilibration. In addition to workflow success rate, efficiency was measured by the number of iterations performed by the LLM. The minimum tool call is set to seven for protein-only systems and eight for protein-ligand systems. This defines the number of essential tool calls required to successfully complete the task, assuming no errors are encountered. We evaluated the performance of DynaMate with five different LLMs detailed in Supplementary Table S3. Herein, we will refer to the LLMs as Claude Opus 4.7[48], Claude Sonnet 4.6[48], GPT-5.4 mini[49], GPT-5.5[49], and Llama 3.3[50, 51]. Each of the 15 systems was tested with the five LLMs in five replicates to better sample the stochastic behavior of LLMs.

Excellent performance (Figure 2) is achieved for the ten non-problematic systems, as evidenced by the 100% success rate obtained for all Claude and GPT models. Llama 3.3 achieves moderately high performance despite being the only open-source model. Its drop in performance is due to the wrong use of tools (incorrect order or wrong variable called). These results show the versatility of DynaMate to adapt its

183 workflow to protein-only or protein-ligand systems, and its robustness across diverse
184 protein and ligand types. In terms of efficiency, Claude and GPT models also achieve
185 outstanding performances for these ten systems, reaching pipeline completion within
186 seven to thirteen iterations. In less than two minutes, the MD simulation is set up, a
187 process that could take hours for the manual setup of an experienced user. On systems
188 for which Llama 3.3 was successful, the efficiency is also very high. Failures in some of
189 the replicate runs lower the performance of Llama 3.3, despite some successful runs.
190 It is interesting to note the slight decrease in efficiency for GPT 5.5 compared to the
191 other models, associated with the LLM’s frequent use of tools to read and list files.
192 This LLM seems to seek a deeper understanding of the simulation process and different
193 files created, to potentially better reason about the outcomes.

194 As expected, the success rates and efficiencies are lower for the five systems with
195 known tool failure modes. Interestingly, the five agents varied in their approach to
196 complete the simulations and achieved different degrees of workflow completion. GPT-
197 5.5 clearly outperformed the other agents, achieving 100% success rates across the
198 five problematic systems. The ability of GPT-5.5 to quickly and accurately identify
199 the causes of the errors, the associated files to modify, and how to properly edit them
200 was unmatched by the other agents. This increase in performance could be associ-
201 ated with the enhanced reasoning mode that this agent seemed to demonstrate on
202 non-problematic systems (higher edit/read tool call rate and slightly lower efficiency).
203 Although less consistent, the other GPT and Claude LLMs also demonstrated impres-
204 sive error analysis and correction capabilities, further detailed in the case studies.
205 GPT-5.4 mini’s slightly lower performance can be noted, associated with a weaker
206 understanding of the error messages. The limitation of Llama 3.3 seems to be clear
207 when errors are encountered, as it did not manage to complete the workflow for any of
208 these five systems. The LLM struggles to understand which files are associated with
209 tool failures. Apart from trying the tools several times, it does not attempt to modify
210 files which was essential in these test cases.

211 2.3 Error correction capability

212 2.3.1 Case study one: clashes with the capping of the protein 213 termini with acetyl or *N*-methyl groups

214 The tool used by DynaMate to cap the protein chains of systems 1U6Q_745 and
215 1C5S_ESX results in atom clashes between the Isoleucine residue at the N-terminus
216 and the capping group added. As seen from Figure 3, Claude Sonnet 4.6, Opus 4.7
217 and GPT-5.5 had at least one successful attempt at removing the clashing hydrogens
218 (H1, H2 and H3 of the capping acetyl residue) for system 1U6Q_745. The tLEaP tool
219 then correctly re-protonated the acetyl group. In all of the failed runs, by Claude and
220 GPT models, the atoms H1, H2 and H3 were correctly identified as being the source
221 of the error, but the maximum number of iterations was reached before the correct
222 edits could be made by the LLMs. Llama 3.3 only pointed out that some atoms did
223 not have a type, without being able to identify which ones. For the 1C5S_ESX system,
224 only GPT-5.5 and Claude Opus 4.7 were able to identify and remove problematic
225 hydrogens in at least one of the replicate runs. Since this system also contained a

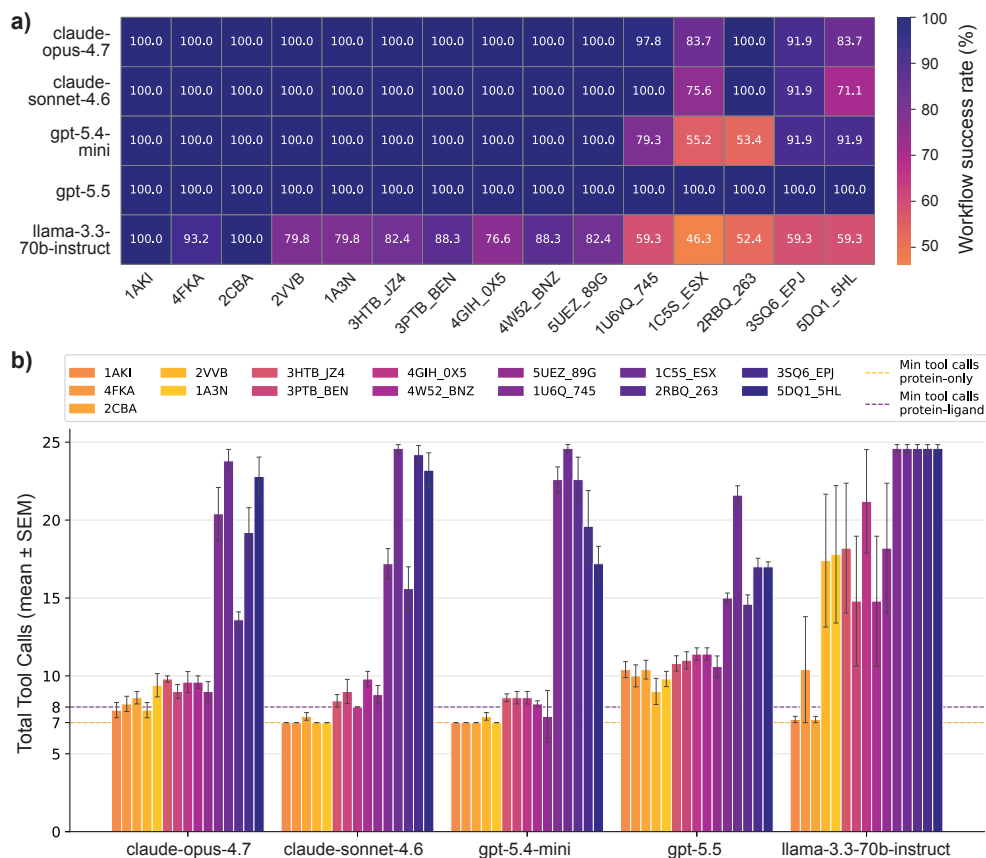


Fig. 2 (a) Workflow success rate of the different LLMs for all tested protein and protein-ligand systems. Success rate is defined as the creation of the necessary (non-empty) files for each essential step, which were manually inspected to ensure they were correct. The mean of the five replicate runs is reported. (b) Efficiency of the different LLMs is defined as the number of tool calls required to complete the run, with a maximum number of iterations set to 25. We denote the minimum number of required tool calls with a dashed line for protein-only (yellow) and protein-ligand (purple) systems. The mean and standard error for the five replicate runs is reported.

ligand protonation error beforehand, the other LLMs had to deal with a few iterations to solve the capping tool error.

2.3.2 Case study two: wrong protonation of ligands

Since Open Babel is used to protonate ligands in DynaMate, there are instances of incorrect ligand protonation. This kind of error was recorded for systems 1C5S_ESX and 2RBQ_263, with ligands containing an amidinium group and an azide group, respectively. For the 2RBQ_263 system, both Claude models and GPT-5.5 achieved a 100% success rate for all replicate runs, and GPT-5.4 mini solved the error in one of its runs. In this case, the extra hydrogen (H10) of the terminal nitrogen (N3) of

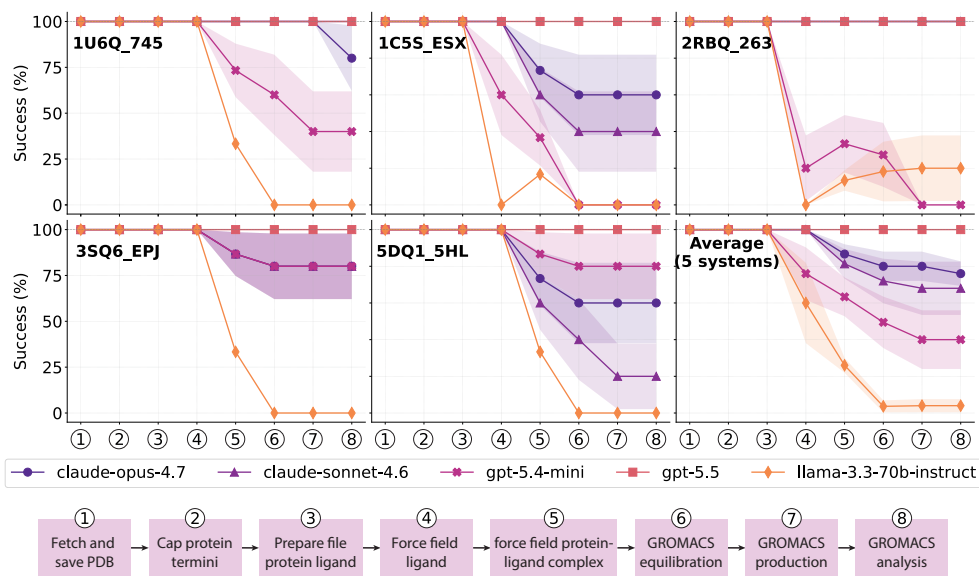


Fig. 3 Tool success rate for the five LLMs on the five problematic systems. Each system and LLM pair was run five times, the plain line is the mean and the shaded part represents the standard error of the mean.

the azide group had to be removed, and the agents demonstrated impressive chemical understanding and reasoning, correctly pinpointing which specific hydrogen should be removed for correct multiplicity and charge to be achieved. As the error message of tLEaP only indicated a problem with the ligand charge, the agents need to understand what is the cause, whether there is a missing or additional hydrogen, and which atom to remove or add accordingly. Performances are a bit lower for the 1C5S_ESX system since on top of the ligand protonation error, the agents also need to solve clashes with capping groups of protein termini, a difficult task within a limited number of tool calls. Nevertheless, in one of the replicate runs, Claude Opus 4.7 and Claude Sonnet 4.6 clearly identified the correct hydrogen that had to be removed from the amidinium group of the ligand, leaving it with a +1 charge. GPT-5.5 solved the error in all replicate runs, extremely efficiently which left enough iterations to also solve the second capping error. For both 1C5S_ESX and 2RBQ_263 systems, Llama 3.3 identified that the error originated from the ligand but could not analyze it further.

2.3.3 Case study three: incompatibility between chlorine atom name in the structure and force field files

System 3SQ6_EPJ suffers from an inconsistency between the chlorine atom name in the PDB file (CL) and the conventional name in force fields (Cl). Claude Sonnet 4.6 and GPT-5.5 achieved 100% success rate for all replicate runs. The only unsuccessful attempt to solve the error originated from editing the wrong file, by Claude Opus

255 4.7 and GPT-5.4 mini on one of their replicate runs. LLama 3.3 didn’t manage to
256 complete the pipeline in any runs. The only attempts taken were to try the same tools
257 sequentially or search the literature for help, which were insufficient. Both Claude
258 models and GPT-5.5 took the strategy to edit the ligand PDB file and replace chlorine
259 with the correct naming convention of the force field. Another strategy, undertaken
260 by GPT-5.4 mini, was to directly modify the force field files (mol2 and prepi files) to
261 normalize the atom name matching the PDB file.

262 2.4 Binding Free Energy with DynaMate

263 Once a protein-ligand simulation is complete, DynaMate offers the option to per-
264 form single trajectory MM/PB(GB)SA calculations to obtain ligand binding free
265 energies, with the `gmx_MMPBSA` tool[20, 52]. The integration of this binding affinity
266 tool was tested against a set of ten inhibitor molecules for the bromodomain 1 of
267 Bromodomain-containing protein 4 (BRD4 BD1) originating from an experimental
268 study[53]. Beforehand, the ten inhibitors were first docked into BRD4 BD1 (PDB:
269 6JJ3[53]). DynaMate then autonomously performed 100 ns MD simulations of the com-
270 plex structures and determined a binding free energy value with the PB model in each
271 case (selecting frames with an interval of five discarding the first 20 ns of the production
272 run). The binding affinity values obtained with DynaMate and the GNINA[54] dock-
273 ing scores were then compared with experimental IC_{50} values[53]. As seen in Figure
274 4, $\Delta\Delta G$ values given by DynaMate’s MM/PBSA calculations are in better agreement
275 with experimental IC_{50} values ($r=0.597$) than GNINA docking scores ($r=0.385$). The
276 MM/PBSA method also offers valuable insights about energetic components of bind-
277 ing in addition to binding scores given by docking softwares. Indeed, energetic van
278 der Waals and electrostatic contributions help rationalize the difference in binding
279 between the ten inhibitors investigated. For instance, the most potent inhibitor (com-
280 pound **10** from Figure 4) seems to bind more strongly than compound **1** (least potent
281 inhibitor) mainly due to an increase in van der Waals interactions, while its stronger
282 binding with respect to compound **2** seems to be linked to stronger electrostatic inter-
283 actions. This automated binding affinity calculation pipeline has direct applications
284 in early-stage drug discovery, where rapid ranking of compound libraries is essential.

285 2.5 Robustness of DynaMate

286 To test the ability of DynaMate to successfully set up a wide variety of systems, 589
287 protein-ligand and 238 protein-only PDB IDs were given as inputs to GPT-5.4 mini.
288 Successful completion of the pipeline was very high for protein-only systems (79%),
289 and reasonably high for protein-ligand systems (62%). Indeed, one of the most diffi-
290 cult challenges of an automatized MD pipeline is dealing with the ligand. The tools
291 given to DynaMate can sometimes lead to wrong protonation or wrong generation of
292 force field parameters. Errors in the PDB file itself are also possible. Nevertheless, the
293 positive impact of the error correction capability of DynaMate is further confirmed by
294 the significant improvement over a deterministic pipeline as shown in Figure 5, with
295 64 protein-ligand systems and 70 protein-only systems for which corrected errors led

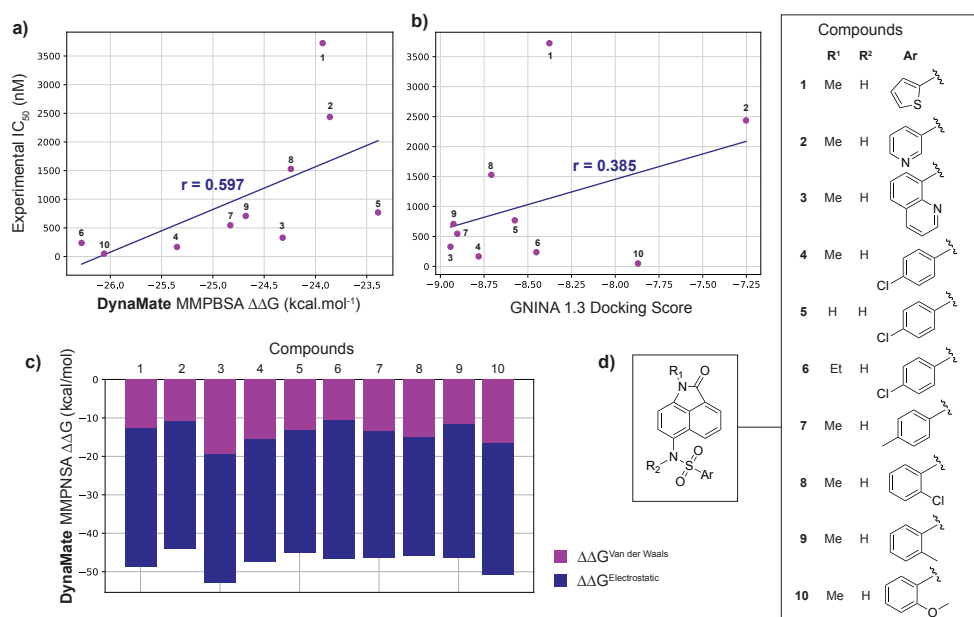


Fig. 4 Comparison of the experimental IC_{50} values with (a) DynaMate MM/PBSA $\Delta\Delta G$ values and (b) GNINA 1.3 docking scores, for a set of BRD4 BD1 inhibitors[53]. (c) Van der Waals and electrostatic contributions to the MM/PBSA $\Delta\Delta G$ DynaMate values. (d) Chemical structures of the ten inhibitors tested for this study.

to pipeline success where it otherwise would have failed. Overall, the agentic framework achieved 10.9% and 29.4% increases in performance compared to a deterministic pipeline for protein-ligand and protein-only systems, respectively. As expected, we find a slow taper of systems proceeding to the next step with no obvious drop-off as the pipeline proceeds (Figure 5).

3 Methods

3.1 DynaMate Infrastructure

Our LLM-based system was implemented with LiteLLM (v.1.79.3) including customized tool use, working memory, and error parsing and feedback. The models were accessed through OpenRouter with the online setting so that we could utilize the web search functionality. The specific details of the models used for the agents, Anthropic's Claude Sonnet 4.6 and Claude Opus 4.7, OpenAI's GPT-5.5 and GPT-5.4 mini, as well as Llama 3.3 can be found in Supplementary Table S3, with model selection guided by a comparative evaluation of latency, cost, reasoning performance, and stability in tool-use-heavy workflows. A temperature setting of 0.1 was selected in order to ensure highly consistent, deterministic reasoning and minimize randomness of tool calls in planning multi-step actions.

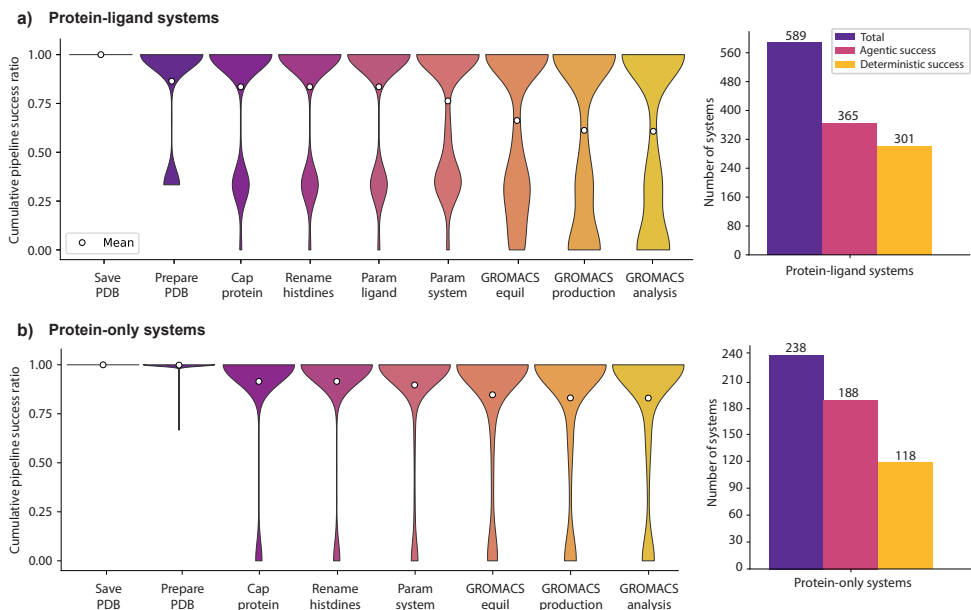


Fig. 5 The overall distribution of tool call success rates for protein (a) and protein-ligand (b) systems that were tested. The mean success value for each step is shown by the white circle while the distribution is detailed through the violin plot. We display the cumulative pipeline success. On the right of each plot we compare the performance of our agentic system to a deterministic pipeline. We display the total systems that were tested by the purple bar, the portion of those systems which the agentic pipeline ran MD simulations successfully by the pink bar, and the number of systems which would have ran successfully with our tools but no agentic driver by the yellow bar.

Each agent is initialized with a personal context in the system prompt (as found in Supplementary Information D), and the custom plan from the planner agent is dynamically incorporated into the worker agents prompt for each run. After each tool call, the agent is provided with a summary of the completed tasks as well as the tasks remaining from the original plan. We also implement custom logic for software errors in order to route the context back to the LLM.

Finally, in order to minimize context overload and forgetting, we periodically summarize the entire chat history into a single assistant message, and replace earlier parts of the chat with it. For robustness, we include the last two assistant messages with custom logic such that tool calls and outputs are not broken in the message history.

3.2 Agentic Workflow

Our system, DynaMate, is designed as a multi-agent framework working in an environment of tools that autonomously prepares, executes, and analyzes MD simulations. The architecture extends previous agentic approaches such as MDCrow[27] and NAMD-Agent[28] by introducing a dedicated planner, dynamic feedback with error correction, and by enabling simulations of protein-ligand complexes.

The workflow is organized into three main components and is contained within a sandboxed environment:

1. **Planner agent:** Extracts user intent and scientific requirements from natural language input, retrieves structural information, and constructs a context-aware simulation plan. This agent identifies whether the system contains a ligand, decides on parameters such as simulation temperature and duration (if not requested by the user), and defines an ordered sequence of simulation subtasks for the system (preprocessing, equilibration, production, and analysis). The planner leverages retrieval-augmented reasoning[55] and tool invocation to retrieve the PDB file and curate an appropriate plan for the MD simulation.
2. **Molecular dynamics agent:** Interprets the plan and autonomously executes it through a loop of tool invocation, validation, and reflection[26]. Each subtask is validated against the filesystem and output logs, allowing the agent to detect and repair common runtime and software errors by regenerating or modifying input files (e.g., correcting missing atom names or updating topology directives).
3. **Analyzer agent:** Post-processes MD trajectories and outputs descriptive analysis. The agent further provides qualitative summaries of stability, convergence, and protein-ligand interactions in natural language form. If some errors were corrected, the agent summarizes its attempts to solve them and gives further recommendations if they were not successful. Comments on next steps, and how to improve further simulations are also included.
4. **Execution environment:** The agents operate within a sandboxed directory structure, ensuring safe read/write operations and reproducibility. Interactions with external LLMs are routed through LiteLLM, which acts as a unified interface for model invocation, request routing, and provider abstraction. The framework supports both AmberTools and GROMACS, expanding prior works restricted to OpenMM or CHARMM-GUI.

3.3 DynaMate Toolset

Our framework interacts with a structured set of domain-specific tools, defined through a formal schema that includes tool names, descriptions, and JSON input specifications. The tool schemes are loaded at runtime and the LLM selects actions through the tools and input arguments. The major tool families include structure preparation, ligand parameterization, system building and solvation, equilibration and production, and web and data retrieval. A representative workflow is shown in Supplementary Figure S1, illustrating the difference between a protein-only simulation and a protein-ligand complex. The planner dynamically selects additional steps for ligand parameterization and topology construction, automatically adjusting the downstream MD setup.

PDB retrieval: If not provided by the user, the PDB coordinate file is fetched from the Protein Data Bank (PDB)[16], given the PDB ID specified by the user.

Structure preparation: The protein coordinates are extracted from the relevant PDB file. If a ligand is requested by the user, its coordinates are also extracted. The protein termini are capped with acetyl (ACE) or *N*-methyl (NME) groups. Atom

names in the protein and ligand are corrected to match force field conventions used in later steps.

Ligand parameterization: If a ligand is present, it is protonated using Open Babel[56] at pH 7.0, unless otherwise specified by the user or by the agent via literature search. The ligand is then parameterized with the antechamber program of AmberTools version 24[57, 58]. The GAFF2 force field is applied, commonly used to parameterize small organic molecules[59].

Protein parameterization and system preparation: The AMBER ff14sb force field is applied for the protein[60]. The system is solvated with the TIP3P model for water[61], and neutralized with Na^+ or Cl^- ions.

Equilibration and production: The MD simulations are performed with the GROMACS version 2023 package[11]. In order to simulate a realistic biochemical environment, periodic boundary conditions are applied. An integration time-step of 2 fs is employed for all simulations and covalent bonds with hydrogen are constrained using the LINCS algorithm[62]. The Verlet cut-off scheme[63] is used throughout all steps with a force-switch modifier starting at 10 Å and a cutoff at 12 Å. The Particle-mesh Ewald (PME) method is used for long-range electrostatics and a cutoff of 12 Å was used for short-range electrostatics[64]. As is standard practice to ensure low energy starting configurations, each system is initially energy minimized for 5000 steps of steepest descent or until a maximum force of 1000 kJmol⁻¹nm⁻¹ is reached on any atom. Following energy minimization, two 100 ps equilibration-runs are performed first in the NVT, then in the NPT ensembles, maintaining the temperature at 300 K (or else if specified by the user or by the agent’s literature search capability) and the pressure at 1 bar using the Berendsen thermostat and barostat, respectively[65]. In the cases of simulations with a ligand, two temperature coupling groups ensure better temperature control on a system with one group consisting of the protein and the ligand, and the other of the solvent (water and ions). Finally, NPT production runs are performed for various lengths.

Analysis: Trajectories are analyzed using GROMACS tools: `gmx_rmsd` for root-mean-square deviation (RMSD) of the protein backbone (with respect to both the first frame and the crystal structure), `gmx_gyr` for radius of gyration, `gmx_rmsf` for root-mean-square fluctuation (RMSF) of protein residues, and `gmx_hbond` for various hydrogen bond quantification.

Binding affinity calculations with the MM/PB(GB)SA method: Binding free energies are computed using the `gmx_MMPBSA` tool (version 1.4.3) with `MMPBSA.py` (version 16.0)[20, 52]. The binding free energy is calculated as $\Delta G = \Delta H - T\Delta S$, where ΔH is the enthalpy change, T is the temperature, and ΔS is the entropy change. The one-trajectory method is used and frames are selected with an interval specified by DynaMate, discarding the first 20% of the simulation.

Web and data retrieval: Web search or PaperQA[25, 32, 33] queries are used to determine experimental conditions such as recommended temperature or ligand information. They can also be used to diagnose common runtime errors.

This modular design allows for tool reuse and extension across multiple agents and systems, and facilitates transparent LLM-tool interactions.

416 3.4 MM/PB(GB)SA test

417 For the MM/PB(GB)SA tests, a set of ten BRD4 BD1 inhibitors was selected: com-
418 pounds **16** to **25** from the Jiang et al. study[53]. These molecules were first docked
419 into BRD4 BD1 (PDB: 6JJ3[53]) using the software GNINA 1.3[54], yielding a docking
420 score for each case. DynaMate then autonomously performed 100 ns MD simula-
421 tions of the complex structures in explicit solvent at 300 K, following the procedure
422 described above. Binding free energy predictions were performed by DynaMate using
423 the MM/PBSA method. For these calculations, frames were extracted with an inter-
424 val of five, omitting the first 20 ns of the simulation. The ionic strength istrng = 0.0
425 M was selected, with the default atomic radii settings.

426 3.5 Robustness test

427 To test the robustness of DynaMate, a set of PDB IDs was selected from the
428 PLINDER[66] dataset, a large and high-quality dataset of protein-ligand complexes.
429 Out of the 77,474 distinct PDB IDs in the dataset, we selected 589 that met the qual-
430 ity criteria (as defined by the authors[66]), contained only one distinct ligand chain,
431 less than three distinct protein chains, did not contain unresolved residues, and for
432 which the ligand was not an oligonucleotide, an oligopeptide, contained metals or was
433 covalent. These 589 PDB IDs were used to evaluate the robustness of our pipeline
434 by running DynaMate with GPT-5.4 mini on each system in three replicates with a
435 capped number of tool calls set to 35 for error correction on each run. Out of these
436 589 PDB IDs, a subset of 238 were selected for the protein-only test, by keeping a
437 balance between successful and unsuccessful systems in the protein-ligand test. A run
438 was considered successful if all the required files were created, non-empty, and the
439 final production MD simulation was able to run.

440 3.6 Error correction and feedback

441 A key feature of the framework is its iterative feedback mechanism, which is similar
442 to the mechanism outlined in[26, 67], enabling self-correction for failed or incomplete
443 tool calls. DynaMate parses the errors from the failed tools which will include the error
444 messages from the respective software programs such as GROMACS, AmberTools, or
445 other general Python errors. Providing this rich context to the agent allows future
446 steps to involve self-correction, such as editing input files with malformed structures,
447 updating atom names to follow expected conventions, and running essential steps that
448 might have been missed resulting in the creation of necessary files that were missing.

449 Each tool invocation is sandboxed, logged, and summarized into a compressed
450 message context to maintain continuity between attempts. If a subtask fails (i.e. due
451 to missing files, naming errors, or incompatible parameters) the agent analyzes the log
452 output and proposes a corrected re-execution plan.

3.7 Retrieval-augmented generation for parameter selection and error control

To support literature-grounded decision-making during the design and execution of MD simulations, we integrated PaperQA[25, 32, 33], a retrieval-augmented question-answering tool that operates over a curated corpus of 90 domain-relevant journal articles. These journal articles are specific to the systems tested in this study, and can be adapted to any other system of choice. PaperQA performs structured retrieval by embedding both user queries and full-text documents into a shared semantic space, followed by passage-level ranking and answer synthesis. Unlike generic RAG pipelines, PaperQA is optimized for scientific texts: it uses metadata-aware chunking, citation-preserving retrieval, and domain-specific answer constraints to ensure that all generated responses are explicitly grounded in referenced source passages.

Within our agentic workflow, PaperQA serves two key purposes. First, it enables the agent to search through relevant literature for appropriate system parameters and methodologies before launching MD runs, such as protonation states, whether the protein works better at a specific pH or temperature, and whether the protein was crystallised under specific conditions that would require some modifications to the PDB file. Second, it provides real-time resources that can help understand errors in input systems that require correction. This substantially reduces human supervision, enabling autonomous workflows that remain aligned with established practices. Some examples of questions asked by the agent to PaperQA and the response are displayed in Supplementary Information F.

Complementary to PaperQA’s closed-corpus retrieval, we enabled the LLMs internal web search tool with medium context size. This tool provides access to up-to-date, open-web resources, such as papers, software documentation, and community-reported best practices. The medium context setting returns passages of sufficient granularity for the model to extract protocol-level details without overwhelming the reasoning steps of the agent.

Together, the web search tool and PaperQA create a hybrid retrieval environment. This combination improves the agent’s ability to generalize across simulation systems, validate parameter choices, and adapt to evolving methods in the MD community.

3.8 Analysis and visualization

Upon successful completion of the MD run, the analyzer performs automated trajectory analysis. It generates RMSD, radius of gyration, RMSF, and hydrogen-bond statistics, using GROMACS analysis tools. Each plot is accompanied by an automatically generated textual interpretation describing structural stability, folding dynamics, and binding behavior.

4 Discussion

Agentic systems like DynaMate illustrate the feasibility of “AI scientists”[25, 68] capable of performing full experimental workflows in computational chemistry. Unlike rigid automation scripts that follow a deterministic workflow, our multi-agent architecture

generalizes across simulation goals and tools by reasoning about both physical constraints and software syntax. In this section, we will go over the four core strengths of DynaMate and outline the current limitations and future directions.

Error catching and self-diagnosing Runtime errors can be challenging and require a significant amount of user time for debugging. Offsetting (some of) this responsibility to an agent can help to streamline scientific discovery. We probed the extent of error correction capabilities through three case studies, which are common issues in MD workflow setups: the capping of protein termini, the protonation of ligands and mismatches between structure and force field files. First, we tested the agents’ ability to deal with the capping of protein termini when this step was not correctly performed by the tool provided to DynaMate. In the case of protein termini capping, using the editing tool correctly to remove clashing hydrogens and identifying the correct files to edit were key. On top of error identification and correction, agents were also tested on their efficiency since for one system, there was an additional error that needed to be solved, complicating the task within a limited scope of tool calls. Except for Llama 3.3, all agents managed to successfully identify and edit the correct files in at least one of the replicate runs. The second case study demonstrated the impressive chemical understanding and reasoning of the agents. Correct ligand protonation can be non-trivial and can require the user to search the literature and use pKa prediction tools. In DynaMate, ligand protonation is performed with Open Babel and in certain cases a mismatch between ligand protonation and charge leads to subsequent error messages. Several cases were observed of correct hydrogen removal in ligands with charge versus protonation mismatch. The coordinates of these hydrogens were in the middle of the ligand coordinate files, and identifying which ones to remove therefore required a good understanding of chemical structures. Case study three (involving a wrong chlorine atom name) clearly demonstrated a successful case of error catching and correction by DynaMate. This example also highlighted that some errors have several working solutions; in this case, either editing the input PDB file or directly supplying the right input parameters for the wrong atom name in the topology file.

These error-correction capabilities address a significant bottleneck in traditional MD workflows, where debugging can require up to hours of expert intervention. DynaMate’s autonomous diagnosis and correction of issues such as atom naming inconsistencies and file formatting errors demonstrates performance that matches common users in routine trouble-shooting tasks.

Literature search An important component of DynaMate is its ability to ground decisions in relevant scientific contexts. The combination of using PaperQA with a curated corpus as well as the web search tool enables the agent to quickly retrieve protocol-level information, recommended parameter choices, and community-reported solutions to common issues. We found that this hybrid system was particularly effective in cases where multiple plausible simulation strategies exist, such as determining protonation states, selecting force-field parameters, or interpreting ambiguous error messages. In these situations, DynaMate was able to justify its choices by referring directly to published methods or authoritative documentation, reducing the burden on the user to manually verify each step. Although retrieval does not guarantee correctness, it substantially improves the transparency and traceability of the agent’s

reasoning, and helps ensure that suggested workflows remain aligned with established practices. As autonomous systems begin to take on more complex scientific tasks, the ability to rapidly connect actions to citable sources becomes an essential safeguard, and our results suggest that DynaMate benefits significantly from this retrieval layer.

Simulation output analysis Another strength of DynaMate is the analysis performed at the end of each run. The agent summarizes the key steps undertaken, the files produced and the potential usage of file editing tools. The equilibration process can be commented on, as well as the meaning of the RMSD, RMSF, radius of gyration and hydrogen bond analysis files produced. DynaMate also provides advice on the next steps that can be performed, or which parameters require more attention. There will for example be a warning if the simulation was too short to achieve protein equilibration. Even in cases where DynaMate is not able to complete the simulation workflow, its subsequent attempts to deal with error messages allow it to gain a better understanding of the system and potential problems. As a result, it provides more useful recommendations and orients the user to the particular files to look into. This feature extends the use of DynaMate. A user who wants to investigate errors in the simulation of a particular system can use DynaMate to receive detailed feedback on what could be the cause of the observed errors.

Binding affinity calculations The incorporation of the MM/PB(GB)SA tool into DynaMate substantially broadens its range of applicability, given that this binding affinity method is routinely employed to rank candidate compounds in drug discovery assays, toxicity assessments, and related studies. As a demonstration, DynaMate successfully computed MM/PBSA $\Delta\Delta G$ values for a series of inhibitor molecules targeting BRD4 BD1. Through this process, the agent gains detailed insight into the binding behavior of the molecular set under consideration. This capability opens the possibility of integrating DynaMate into broader agentic frameworks, including those designed for drug optimization, where the acquired knowledge could be leveraged to guide the optimization process toward molecules with more favorable binding properties. It is important to note that one must be aware of the approximations associated with the MM/PB(GB)SA method to critically assess the results obtained[69]. To lower the computational cost compared with more rigorous free-energy calculations, MM/PB(GB)SA only integrates the conformational sampling from MD simulations in explicit solvent, while the free energy itself is estimated from an implicit solvent model. These approximations generally fail to accurately represent changes in the conformational entropy of protein and ligand, as well as conformational adjustments between the bound and unbound states. Such errors usually cancel out when ranking similar complexes, and MM/PB(GB)SA is mostly used in comparative studies.

Limitations The framework depends on external LLM APIs, so users must manage API access and credits, and rate limits and data handling apply. It also runs without persistent memory, so each run is independent of prior ones, and adding long-term memory alongside the existing working memory is a natural next step. Performance tracks the underlying model: the Claude and GPT agents handled error analysis, correction, and literature search far more reliably than the open-source Llama agent, so capability will improve as the available models do. The present toolset targets single-ligand systems and does not yet cover multiple distinct ligands, nucleic

584 acids, or membrane proteins. Non-standard atom names, missing residues, and incom-
585 plete coordinates remain the main failure sources, which argues for strengthening the
586 underlying tools in parallel with agent reasoning. As in any protein-ligand MD study,
587 system-specific choices should be verified by the user, such as the protonation of his-
588 tidine, aspartate, and glutamate residues, and of titratable groups on ligands. For
589 non-standard ligands, the force field parameters can also be refined for more reliable
590 results, using for example RESP charges or dihedral fitting. Overall, while Dyna-
591 Mate is an ideal tool for automated protein and protein-ligand studies, in depth and
592 accurate studies of one particular system would require more expert knowledge than
593 is currently offered by available LLMs.

594 5 Conclusion

595 In conclusion, tools like DynaMate lower the entry barrier for running complex
596 scientific workflows, thus making MD simulations more accessible to the broader sci-
597 entific community. The streamlined usability of DynaMate also enables more efficient
598 incorporation of computational binding-affinity calculations into drug-optimization
599 campaigns. By providing easy access to newcomers in the field, DynaMate democ-
600 ratizes usage of computational tools that are increasingly critical for structure-based
601 drug design, potentially accelerating the identification and optimization of therapeutic
602 candidates. As traditional computational drug discovery pipelines require significant
603 expert time for system preparation, the combination of autonomous simulation setup,
604 error correction, and binding affinity calculations positions DynaMate as a practical
605 tool for medicinal chemistry applications. Additionally, the modularity of the agen-
606 tic architecture will allow facile extensions. The context-aware nature of our system
607 makes it possible to suggest the most appropriate workflows for a series of new tasks.
608 Future works will introduce diverse systems into the pipeline, such as DNA and mem-
609 brane proteins. Software specific documentation and on-demand retrieval of scientific
610 papers for new systems will also be integrated.

611 **Supplementary information.** This article has an accompanying supplementary
612 information file.

613 **Acknowledgements.** SG acknowledges support from Intel and Merck KGaA via
614 the AWASES programme, and the Swiss National Science Foundation (SNSF) (grant
615 number: 200020-219440). CM acknowledges Valence Labs for financial support. BN
616 acknowledges funding from Swiss National Science Foundation (SNSF) (grant num-
617 ber: 226509). UR acknowledges support from the Swiss National Science Foundation
618 (SNSF). PS acknowledges support from the NCCR Catalysis (grant number 225147),
619 a National Centre of Competence in Research funded by the Swiss National Science
620 Foundation (SNSF). This work used computing time from the Swiss National Comput-
621 ing Centre CSCS. The authors gratefully acknowledge Víctor Sabanza Gil for providing
622 the docked BRD4 BD1 inhibitor structures used for the MM/PB(GB)SA test.

623 **Funding.** Intel and Merck KGaA via the AWASES programme, Swiss National Sci-
624 ence Foundation (SNSF) (grant numbers: 200020-219440, 226509), NCCR Catalysis
625 (grant number 225147).

626 **Conflict of interest/Competing interests.** Not applicable
627 **Ethics approval and consent to participate.** Not applicable
628 **Consent for publication.** Not applicable
629 **Data availability.** Not applicable
630 **Materials availability.** Not applicable
631 **Code availability.** The code to reproduce results and utilize the agentic frame-
632 work with open-source softwares is available at [https://github.com/schwallergroup/](https://github.com/schwallergroup/DynaMate)
633 [DynaMate](https://github.com/schwallergroup/DynaMate).
634 **Author contribution.** Using the CRediT system: Conceptualization: CM, SG, JG,
635 & PS; Data curation: CM, SG & JG; Formal analysis: CM, SG & JG; Investigation:
636 CM, SG, JG, UR & PS; Methodology: CM, SG, JG & BN. Project administration:
637 UR, PS; Resources: PS; Software: CM, SG JG & BN; Supervision: UR, PS; Validation:
638 CM, SG & JG; Visualization: CM, SG & JG; Writing - original draft: CM, SG, JG,
639 BN & PS; Writing - review and editing: CM, SG, JG, BN, UR & PS.

References

- [1] Hollingsworth, S.A., Dror, R.O.: Molecular Dynamics Simulation for All . *Neuron* **99**(6), 1129–1143 (2018) <https://doi.org/10.1016/j.neuron.2018.08.011>
- [2] Karplus, M., McCammon, J.A.: Molecular dynamics simulations of biomolecules. *Nature structural biology* **9**(9), 646–652 (2002)
- [3] Schlick, T., Portillo-Ledesma, S.: Biomolecular modeling thrives in the age of technology. *Nature computational science* **1**(5), 321–331 (2021)
- [4] Schlick, T., Collepardo-Guevara, R., Halvorsen, L.A., Jung, S., Xiao, X.: Biomolecular modeling and simulation: a field coming of age. *Quarterly reviews of biophysics* **44**(2), 191–228 (2011)
- [5] Davis, M.E., McCammon, J.A.: Electrostatics in biomolecular structure and dynamics. *Chemical Reviews* **90**(3), 509–521 (1990)
- [6] Honig, B., Nicholls, A.: Classical electrostatics in biology and chemistry. *Science* **268**(5214), 1144–1149 (1995)
- [7] Feig, M., Brooks III, C.L.: Recent advances in the development and application of implicit solvent models in biomolecule simulations. *Current opinion in structural biology* **14**(2), 217–224 (2004)
- [8] Onufriev, A.V., Case, D.A.: Generalized born implicit solvent models for biomolecules. *Annual review of biophysics* **48**(1), 275–296 (2019)
- [9] Eisenberg, D., McLachlan, A.D.: Solvation energy in protein folding and binding. *Nature* **319**(6050), 199–203 (1986)
- [10] Nandigrami, P., Szczepaniak, F., Boughter, C.T., Dehez, F., Chipot, C., Roux, B.: Computational assessment of protein–protein binding specificity within a family of synaptic surface receptors. *The Journal of Physical Chemistry B* **126**(39), 7510–7527 (2022)
- [11] Abraham, M.J., Murtola, T., Schulz, R., Páll, S., Smith, J.C., Hess, B., Lindahl, E.: Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **1**, 19–25 (2015)
- [12] Eastman, P., Swails, J., Chodera, J.D., McGibbon, R.T., Zhao, Y., Beauchamp, K.A., Wang, L.-P., Simmonett, A.C., Harrigan, M.P., Stern, C.D., *et al.*: Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS computational biology* **13**(7), 1005659 (2017)
- [13] Case, D.A., Aktulga, H.M., Belfon, K., Cerutti, D.S., Cisneros, G.A., Cruzeiro, V.W.D., Forouzes, N., Giese, T.J., Gotz, A.W., Gohlke, H., *et al.*: Ambertools. *Journal of chemical information and modeling* **63**(20), 6183–6191 (2023)

- [14] Brooks, B.R., Brooks III, C.L., Mackerell Jr, A.D., Nilsson, L., Petrella, R.J., Roux, B., Won, Y., Archontis, G., Bartels, C., Boresch, S., *et al.*: Charmm: the biomolecular simulation program. *Journal of computational chemistry* **30**(10), 1545–1614 (2009)
- [15] Lemkul, J.A.: Introductory tutorials for simulating protein dynamics with gromacs. *The Journal of Physical Chemistry B* **128**(39), 9418–9435 (2024)
- [16] Burley, S.K., Bhatt, R., Bhikadiya, C., Bi, C., Biester, A., Biswas, P., Bittrich, S., Blaumann, S., Brown, R., Chao, H., *et al.*: Updated resources for exploring experimentally-determined pdb structures and computed structure models at the rcsb protein data bank. *Nucleic acids research* **53**(D1), 564–574 (2025)
- [17] Yekeen, A.A., Durojaye, O.A., Idris, M.O., Muritala, H.F., Arise, R.O.: Chaperong: A tool for automated gromacs-based molecular dynamics simulations and trajectory analyses. *Computational and structural biotechnology journal* **21**, 4849–4858 (2023)
- [18] Carvalho Martins, L., Cino, E.A., Ferreira, R.S.: Pyautofep: An automated free energy perturbation workflow for gromacs integrating enhanced sampling methods. *Journal of Chemical Theory and Computation* **17**(7), 4262–4273 (2021)
- [19] Lenselink, E.B., Louvel, J., Forti, A.F., Veldhoven, J.P., Vries, H., Mulder-Krieger, T., McRobb, F.M., Negri, A., Goose, J., Abel, R., *et al.*: Predicting binding affinities for gpcr ligands using free-energy perturbation. *ACS omega* **1**(2), 293–304 (2016)
- [20] Valdés-Tresanco, M.S., Valdés-Tresanco, M.E., Valiente, P.A., Moreno, E.: gmx_mmpbsa: a new tool to perform end-state free energy calculations with gromacs. *Journal of chemical theory and computation* **17**(10), 6281–6291 (2021)
- [21] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., *et al.*: Improving language understanding by generative pre-training. OpenAI preprint (2018)
- [22] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. *Advances in neural information processing systems* **30** (2017)
- [23] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., *et al.*: Language models are few-shot learners. *Advances in neural information processing systems* **33**, 1877–1901 (2020)
- [24] M. Bran, A., Cox, S., Schilter, O., Baldassari, C., White, A.D., Schwaller, P.: Augmenting large language models with chemistry tools. *Nature Machine Intelligence* **6**(5), 525–535 (2024)

- [25] Skarlinski, M.D., Cox, S., Laurent, J.M., Braza, J.D., Hinks, M., Hammerling, M.J., Ponnampati, M., Rodriques, S.G., White, A.D.: Language agents achieve superhuman synthesis of scientific knowledge. arXiv preprint arXiv:2409.13740 (2024)
- [26] Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems* **36**, 8634–8652 (2023)
- [27] Campbell, Q., Cox, S., Medina, J., Watterson, B., White, A.D.: Mdcrow: Automating molecular dynamics workflows with large language models. arXiv preprint arXiv:2502.09565 (2025)
- [28] Chandrasekhar, A., Farimani, A.B.: Automating md simulations for proteins using large language models: Namd-agent. arXiv preprint arXiv:2507.07887 (2025)
- [29] Shi, Z., Xin, C., Huo, T., Jiang, Y., Wu, B., Chen, X., Qin, W., Ma, X., Huang, G., Wang, Z., *et al.*: A fine-tuned large language model based molecular dynamics agent for code generation to obtain material thermodynamic parameters. *Scientific Reports* **15**(1), 10295 (2025)
- [30] Thompson, A.P., Aktulga, H.M., Berger, R., Bolintineanu, D.S., Brown, W.M., Crozier, P.S., in ’t Veld, P.J., Kohlmeyer, A., Moore, S.G., Nguyen, T.D., Shan, R., Stevens, M.J., Tranchida, J., Trott, C., Plimpton, S.J.: LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.* **271**, 108171 (2022) <https://doi.org/10.1016/j.cpc.2021.108171>
- [31] Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., Anandkumar, A.: Voyager: An open-ended embodied agent with large language models. arXiv preprint arXiv:2305.16291 (2023)
- [32] Narayanan, S., Braza, J.D., Griffiths, R.-R., Ponnampati, M., Bou, A., Laurent, J., Kabeli, O., Wellawatte, G., Cox, S., Rodriques, S.G., White, A.D.: Aviary: training language agents on challenging scientific tasks. arXiv preprint arXiv:2412.21154 (2024)
- [33] Lála, J., O’Donoghue, O., Shtedritski, A., Cox, S., Rodriques, S.G., White, A.D.: Paperqa: Retrieval-augmented generative agent for scientific research. arXiv preprint arXiv:2312.07559 (2023)
- [34] Boyce, S.E., Mobley, D.L., Rocklin, G.J., Graves, A.P., Dill, K.A., Shoichet, B.K.: 2-propylphenol in complex with t4 lysozyme l99a/m102q (2009) <https://doi.org/10.2210/pdb3htb/pdb>
- [35] Bode, W., Schwager, P., Walter, J.: The geometry of the reactive site and of the

- 748 peptide groups in trypsin, trypsinogen and its complexes with inhibitors (1983)
749 <https://doi.org/10.2210/pdb3ptb/pdb>
- 750 [36] Ultsch, M.H.: Tyk2 (jh1) in complex with 2,6-dichloro-n-2-
751 [(cyclopropylcarbonyl)amino]pyridin-4-ylbenzamide (2013) <https://doi.org/10.2210/pdb4gih/pdb>
752
- 753 [37] Merski, M., Shoichet, B.K., Eidam, O., Fischer, M.: T4 lysozyme 199a with
754 benzene bound (2015) <https://doi.org/10.2210/pdb4w52/pdb>
- 755 [38] Park, C.H.: Brd4.bd2.a1342843 (2017) <https://doi.org/10.2210/pdb5uez/pdb>
- 756 [39] Carter, D., He, J., Ruble, J.R., Wright, B.: The structures of the orthorhombic
757 form of hen egg-white lysozyme at 1.5 angstroms resolution (1997) <https://doi.org/10.2210/pdb1aki/pdb>
758
- 759 [40] Prugovečki, B., Pulić, I., Toth, M., Matković-Čalogović, D.: High resolution
760 structure of the manganese derivative of insulin (2013) <https://doi.org/10.2210/pdb4fka/pdb>
761
- 762 [41] Tame, J., Vallone, B.: Deoxy human hemoglobin (1998) <https://doi.org/10.2210/pdb1a3n/pdb>
763
- 764 [42] Håkansson, K., Carlsson, M., Svensson, L.A., Liljas, A.: Structure of native and
765 apo carbonic anhydrase ii and structure of some of its anion-ligand complexes
766 (1993) <https://doi.org/10.2210/pdb2cba/pdb>
- 767 [43] Sjöblom, B., Polentarutti, M., Djinović-Carugo, K.: Human carbonic anhydrase
768 ii in complex with bicarbonate (2009) <https://doi.org/10.2210/pdb2vnb/pdb>
- 769 [44] Bruncko, M., McClellan, W., Wendt, M.D., Sauer, D.R., Geyer, A., Dalton, C.R.,
770 Kaminski, M.K., Nienaber, V.L., Rockway, T.R., Giranda, V.L.: Substituted 2-
771 naphthamidine inhibitors of urokinase (2004) <https://doi.org/10.2210/pdb1u6q/pdb>
772
- 773 [45] Graves, A.P., Boyce, S.E., Shoichet, B.K.: 3-methylbenzylazide in complex with
774 t4 l99a/m102q (2008) <https://doi.org/10.2210/pdb2rbq/pdb>
- 775 [46] Li, S.-X., Huang, S., Bren, N., Noridomi, K., Dellisanti, C.D., Sine, S.M., Chen, L.:
776 Crystal structures of the ligand binding domain of a pentameric alpha7 nicotinic
777 receptor chimera with its agonist epibatidine <https://doi.org/10.2210/pdb3sq6/pdb>
778
- 779 [47] Radeva, N., Uehlein, M., Weiss, M., Heine, A., Klebe, G.: Endothiapepsin in
780 complex with fragment 34 (2016) <https://doi.org/10.2210/pdb5DQ1/pdb>
- 781 [48] Anthropic: System card: Claude opus 4 claude sonnet 4. Anthropic Blog (2025)

- [49] Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., et al.: Openai gpt-5 system card. arXiv preprint arXiv:2601.03267 (2025)
- [50] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
- [51] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al.: The llama 3 herd of models. arXiv e-prints, 2407 (2024)
- [52] Miller III, B.R., McGee Jr, T.D., Swails, J.M., Homeyer, N., Gohlke, H., Roitberg, A.E.: Mmpbsa. py: an efficient program for end-state free energy calculations. Journal of chemical theory and computation **8**(9), 3314–3321 (2012)
- [53] Jiang, F., Hu, Q., Zhang, Z., Li, H., Li, H., Zhang, D., Li, H., Ma, Y., Xu, J., Chen, H., *et al.*: Discovery of benzo [cd] indol-2 (1 h)-ones and pyrrolo [4, 3, 2-de] quinolin-2 (1 h)-ones as bromodomain and extra-terminal domain (bet) inhibitors with selectivity for the first bromodomain with potential high efficiency against acute gouty arthritis. Journal of medicinal chemistry **62**(24), 11080–11107 (2019)
- [54] McNutt, A.T., Li, Y., Meli, R., Aggarwal, R., Koes, D.R.: Gnina 1.3: the next increment in molecular docking with deep learning. Journal of Cheminformatics **17**(1), 28 (2025)
- [55] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., *et al.*: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems **33**, 9459–9474 (2020)
- [56] O’Boyle, N.M., Banck, M., James, C.A., Morley, C., Vandermeersch, T., Hutchison, G.R.: Open babel: An open chemical toolbox. Journal of cheminformatics **3**(1), 33 (2011)
- [57] Wang, J., Wang, W., Kollman, P.A., Case, D.A.: Automatic atom type and bond type perception in molecular mechanical calculations. Journal of molecular graphics and modelling **25**(2), 247–260 (2006)
- [58] Case, D., Aktulga, H., Belfon, K., Ben-Shalom, I., Berryman, J., Brozell, S., Cerutti, D., Cheatham, T., Cisneros, G., Cruzeiro, V., *et al.*: Amber2020, university of california, san francisco. J. Amer. Chem. Soc **142**, 3823–3835 (2020)
- [59] Wang, J., Wolf, R.M., Caldwell, J.W., Kollman, P.A., Case, D.A.: Development and testing of a general amber force field. Journal of computational chemistry **25**(9), 1157–1174 (2004)

- [60] Maier, J.A., Martinez, C., Kasavajhala, K., Wickstrom, L., Hauser, K.E., Simmerling, C.: ff14sb: improving the accuracy of protein side chain and backbone parameters from ff99sb. *Journal of chemical theory and computation* **11**(8), 3696–3713 (2015)
- [61] Mark, P., Nilsson, L.: Structure and dynamics of the tip3p, spc, and spc/e water models at 298 k. *The Journal of Physical Chemistry A* **105**(43), 9954–9960 (2001)
- [62] Hess, B., Bekker, H., Berendsen, H.J., Fraaije, J.G.: Lincs: A linear constraint solver for molecular simulations. *Journal of computational chemistry* **18**(12), 1463–1472 (1997)
- [63] Verlet, L.: Computer” experiments” on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical review* **159**(1), 98 (1967)
- [64] Darden, T., York, D., Pedersen, L., *et al.*: Particle mesh ewald: An $n \log(n)$ method for ewald sums in large systems. *Journal of chemical physics* **98**, 10089–10089 (1993)
- [65] Berendsen, H.J., Postma, J.v., Van Gunsteren, W.F., DiNola, A., Haak, J.R.: Molecular dynamics with coupling to an external bath. *The Journal of chemical physics* **81**(8), 3684–3690 (1984)
- [66] Durairaj, J., Adeshina, Y., Cao, Z., Zhang, X., Oleinikovas, V., Duignan, T., McClure, Z., Robin, X., Studer, G., Kovtun, D., *et al.*: Plinder: The protein-ligand interactions dataset and evaluation resource. *BioRxiv*, 2024–07 (2024)
- [67] Zou, Y., Cheng, A.H., Aldossary, A., Bai, J., Leong, S.X., Campos-Gonzalez-Angulo, J.A., Choi, C., Ser, C.T., Tom, G., Wang, A., *et al.*: El agente: An autonomous agent for quantum chemistry. *Matter* **8**(7) (2025)
- [68] Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Van Katwyk, P., Deac, A., *et al.*: Scientific discovery in the age of artificial intelligence. *Nature* **620**(7972), 47–60 (2023)
- [69] Roux, B., Chipot, C.: Editorial guidelines for computational studies of ligand binding using mm/pbsa and mm/gbsa approximations wisely. *The Journal of Physical Chemistry B* **128**(49), 12027–12029 (2024)
- [70] Ibrahim, M., Zacharias, M.: Protein–protein interaction stabilizers from md simulation-derived pharmacophores. *Journal of Chemical Information and Modeling* (2026)

Supplementary Information

A DynaMate capabilities

Here we outline the current state of both existing agentic frameworks that run MD simulations, and our framework. We specify both the types of systems it can accept, and the range of tasks it can perform.

Table S1 Comparison of capabilities across existing agentic MD frameworks. DynaMate extends prior systems by enabling ligand handling, adaptive error correction, and retrieval-augmented parameter selection.

Capability	MDCrow	NAMD-Agent	DynaMate (ours)
Protein-only simulations	Yes	Yes	Yes
Protein-ligand simulations	No	No	Yes
Protein-ligand binding affinity calculations	No	No	Yes
Automated structure cleaning	Partial	Yes	Yes
Force field parameterization	Yes (OpenMM)	Yes (CHARMM-GUI)	Yes (Amber-Tools/GROMACS)
Retrieval from literature/databases	Yes (limited)	No	Yes (RAG via RCSB/PubChem/-Google)
Adaptive tool selection	No	Partial	Yes
Error correction and recovery	No	No	Yes
Natural-language task specification	Yes	Yes	Yes
Modular multi-agent architecture	No	Partial	Yes
Cross-platform compatibility	OpenMM only	CHARMM only	GROMACS / AmberTools

5

B Human defined workflow

There is a series of tools provided to the agent that it should use correctly for simulation success. The tools provided include:

- Fetch and save the pdb file.
- Preparation of pdb files. This comprises cleaning the protein file, capping it (C- and N- termini with acetyl (ACE) and *N*-methyl (NME) groups)[70], extracting the ligand file, and protonating it at pH = 7 using Open Babel[56].
- Preparation of ligand parameters with AmberTools.
- Generation of the protein-ligand complex file if a ligand is present.
- Web and data retrieval: query of external sources (RCSB, PubChem, Google) to determine experimental conditions such as recommended temperature or ligand information.
- Generation parameters for the protein or protein-ligand systems using tLEaP. The protein is parameterized with the ff14sb force field and the ligand with GAFF2, with AM1-BCC charges.
- Preparation the simulation box with AmberTools, by solvating with the TIP3P model and neutralising with ions.
- Equilibration the system with GROMACS: it includes energy minimisation until the maximum force on all atoms is inferior to 1000 kJmol⁻¹nm⁻¹, 100 ps NVT and 100 ps NPT simulations to equilibrate the temperature and pressure of the system. Both properties are plotted and analyzed for convergence.
- Generation of the NPT production run with GROMACS, with the temperature either specified by the user, or determined by the agent with its web-search tool.
- Analysis of the RMSD, RMSF, radius of gyration, number of hydrogen-bonds (between protein side-chains, protein-ligand and protein-water) plots generated by GROMACS.

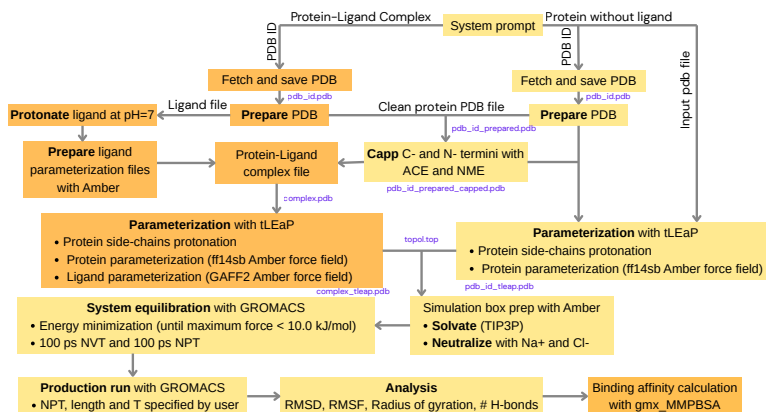


Fig. S1 Agentic workflow for a protein-only system (yellow) and a protein-ligand system (orange and yellow). Steps include structure retrieval, preprocessing, generation of input force field parameters, solvation, equilibration, and production.

C Reference systems for evaluation

Table S2: Systems used to evaluate the LLM agent for molecular dynamics simulations (protein-ligand complexes and protein-only systems).

PDB ID	Protein	Ligand	Rationale for Selection
1AKI [39]	Lysozyme	/	High resolution crystal structure of a protein monomer.
4FKA [40]	Manganese derivative of insulin	/	High resolution structure of one of the most common human proteins.
1A3N [41]	Hemoglobin	/	High resolution crystal structure of a protein multimer. Good case study for multiple chain system.
2CBA [42]	Carbonic anhydrase II	/	High resolution crystal structure of a protein monomer.
2VVB [43]	Carbonic anhydrase II	/	High resolution crystal structure of another carbonic anhydrase structure. Test for the sensitivity to small changes in crystallographic data.
3HTB [34]	T4 Lysozyme L99A/M102Q	JZ4	Common model for studying polar cavity binding; probes hydrogen bonding and solvation effects.
3PTB [35]	β -Trypsin	BEN	Canonical benchmark system for protein-ligand MD; simple and well-characterized binding site.
4GIH [36]	Tyk2 (JH1 domain)	0X5	Popular benchmark for free energy methods; halogenated, conformationally complex inhibitor.
4W52 [37]	T4 Lysozyme L99A	BNZ	Minimal hydrophobic binding model; widely used to test force fields and thermodynamics.
5UEZ [38]	BRD4 (Bromodomain-containing protein 4)	89G	Pharmacologically relevant target; tests handling of flexible, aromatic ligands.
1U6Q [44]	Urokinase	745	Error: incorrect N-terminus capping with <i>N</i> -methyl (NME) group by DynaMate’s capping tool.*
1C5S [44]	Urokinase type plasminogen activator	ESX	Error: incorrect protonation of the ligand’s amidinium group by DynaMate’s protonation tool (Open Babel) & incorrect N-terminus capping with <i>N</i> -methyl (NME) group by DynaMate’s capping tool.*

Continued on next page

Table S2 – continued from previous page

PDB ID	Protein	Ligand	Rationale for Selection
2RBQ [45]	T4 Lysozyme L99A/M102Q	263	Error: incorrect protonation of the ligand's azide group by DynaMate's protonation tool (Open Babel).*
3SQ6 [46]	Pentameric Alpha7 Nicotinic Receptor Chimera	EPJ	Error: mismatch between the chlorine atom name of the ligand in the PDB file and the atom name of chlorine in the amber14sb force field, leading to errors with DynaMate's tLEaP tool.*
5DQ1 [47]	Hydrolase	5HL	Error: incorrect N-terminus capping with <i>N</i> -methyl (NME) group by DynaMate's capping tool.*

*Known problematic system requiring LLM error correction ability for pipeline success.

33

34 Bromodomain-containing protein 4 (BRD4 BD1). Ten inhibitors (compounds **16** to
 35 **25** from Jiang et al. [53]) were first docked into BRD4 BD1 (PDB: 6JJ3) with the
 36 software GNINA 1.3[54].

37 D Specialized system prompts

```
PREP_SYSTEM_PROMPT =
You are a helpful science assistant designed to fetch information about protein
structures and ligands, and make helpful suggestions regarding molecular systems.
Classify the user request and prepare the input files for an appropriate molecular
dynamics pipeline. Depending on the user inputs you should define what a successful
MD pipeline would involve. Call the relevant tools when needed to prepare the system
for molecular dynamics.
```

```
MD_SYSTEM_PROMPT =
You are an MD execution assistant. You have access to tools that prepare and
run molecular dynamics (MD) simulations using GROMACS.

The user has already provided a PDB structure file at {pdb_path} and the
necessary MDP files to run GROMACS. They are available in the sandbox directory
located at {sandbox_dir}, and you should use them.

You should use the tools to solvate, equilibrate, and run MD in sequence,
starting with a simulation using GROMACS and the Amber force field ff14sb.

First, if the system is a protein-ligand complex, separate the protein and
ligands into two separate pdb files. If a ligand is present, protonate it at the
appropriate pH. Check that the protonation is accurate based on a literature search.
Then, if a ligand is present, parameterise the ligand using antechamber with Amber.
If the system is a protein-ligand complex, merge the two into complex.pdb,
otherwise if the system is a protein alone, keep only the protein atoms.

Next, parameterise the system using tleap, which allows to create a box,
solvate, add ions to neutralise and prepare the 'topol.top' file. In the tleap step,
the protein will be protonated. Check that the protonation is accurate by doing
a literature search, and that the protein was protonated at the optimum pH.

Then, perform a short energy minimization using GROMACS.
Next, equilibrate with short NVT and NPT runs.
Finally, perform the production run.

After production run is complete, perform a basic analysis of the trajectory
including RMSD, RMSF calculations, radius of gyration, and hydrogen bond analysis.
The analysis of these plots should be saved as a text file named "analysis.txt"
in the sandbox directory.

If any step fails, retry after analyzing the provided error message and make
corrections to the inputs for the current step.
```

38 E LLM specifications

Table S3 Overview of evaluated large language models and evaluation settings.

Provider	Model	Identifier	Tools	Web	Temp	WS Ctx
Anthropic	Claude Opus 4.7	anthropic/claude-opus-4.7	Y	Y	0.1	Medium
Anthropic	Claude Sonnet 4.6	anthropic/claude-sonnet-4.6	Y	Y	0.1	Medium
OpenAI	GPT-5.5	openai/gpt-5.5	Y	Y	0.1	Medium
OpenAI	GPT-5.4 mini	openai/gpt-5.4-mini	Y	Y	0.1	Medium
Meta	Llama 3.3 70B Instruct	meta-llama/llama-3.3-70b-instruct	Y	Y	0.1	Medium

39 F Examples of PaperQA questions and answers

Table S4 Examples of questions asked by the agent to PaperQA as part of the `search_papers` tool. The answers are obtained through a RAG system searching in a curated corpus of 90 domain-relevant journal articles.

Examples of agent questions to PaperQA (categorized)
MD simulation parameters selection
Question: molecular dynamics simulation temperature for 3PTB protein
Question: Why are MD simulations performed at room temperature?
Question: short molecular dynamics simulation duration
Protonation states
Question: Protonation states of histidine residues in protein simulations at pH 7
Question: protonation and charge states of ligand ADN at pH 7 for molecular dynamics simulations
Question: protonation and atom types for ligand ADN in protein simulations at pH 7
Question: protonation state and parameterization of ligand ADN for Amber force field at pH 7
Error solving and softwares
Question: how to use gmx pdb2gmx to generate a gromacs topology from a pdb file
Question: Why does ParmEd fail with a FileNotFoundError when using run_tleap?
Question: Common issues with GROMACS energy minimization infinite forces and atom overlaps

40 **Table S5.** Examples of answers generated by PaperQA following the agent questions using
41 the `search_papers` tool.

Question: molecular dynamics simulation temperature for 3PTB protein
The molecular dynamics (MD) simulation temperature for the 3PTB protein is explicitly stated as 298.15 K. This temperature is achieved during the heating phase, where the system is gradually heated from 10 K to 298.15 K over 100 ps at constant volume using a Langevin thermostat with a collision frequency of 1.0 ps^{-1} . Subsequent simulations are conducted at this constant temperature and pressure (heinzelmann2021automationofabsolute pages 10-11). Other excerpts discuss general MD methodologies, including heating, equilibration, and production phases, but do not provide specific temperature details for the 3PTB protein (patodia2014moleculardynamicssimulation pages 2-3, patodia2014moleculardynamicssimulation pages 1-1, patodia2014moleculardynamicssimulation pages 3-4). Additionally, while one source describes simulations conducted at temperatures ranging from 300 K to 320 K, this information pertains to a different system and is not relevant to the 3PTB protein (lamprakidis2021evaluatingtheefficiency pages 38-41). Thus, the only directly relevant information specifies that the MD simulation temperature for the 3PTB protein is 298.15 K (heinzelmann2021automationofabsolute pages 10-11).
Question: Why does ParmEd fail with a FileNotFoundError when using run_tleap?

Continued on next page

Table S6 – continued from previous page

ParmEd may fail with a <code>FileNotFoundError</code> when using <code>run_tleap</code> due to missing or improperly referenced template or parameter files required for the workflow. The AMBER <code>parmbsc0</code> force field, commonly used for DNA simulations, relies on templates generated using tools like TINKER, which define parameters such as charges, bonds, and angles for residues in specific chain positions. If these template files are absent or incorrectly specified, errors like <code>FileNotFoundError</code> can occur during the execution of <code>run_tleap</code> (lopez2015mappingbiophysicsthrough pages 49-51, lopez2015mappingbiophysicsthrough pages 51-54). Additionally, topology files must be consistent and properly parameterized for tools like ParmEd and <code>run_tleap</code> to function correctly. Inconsistencies or missing files, such as topology or input files, can lead to such errors. The Python utility <code>ante-MMPBSA.py</code> can assist in generating consistent topology files, which may help prevent these issues (miller2012ijlmmmpbsa.pyijlmmmpbsa.py pages 3-4). Errors may also arise from missing or misconfigured input files, such as PQR files or solvent model parameters, which are critical for molecular dynamics workflows. Proper configuration and verification of these files are essential to avoid <code>FileNotFoundError</code> during simulations (lopez2015mappingbiophysicsthrough pages 56-58). In summary, the failure is likely due to missing or improperly referenced template, parameter, or input files critical for the workflow (pqac-e03c3d8a, miller2012ijlmmmpbsa.pyijlmmmpbsa.py pages 3-4, lopez2015mappingbiophysicsthrough pages 56-58).
Question: Common issues with GROMACS energy minimization infinite forces and atom overlaps Common issues with energy minimization in GROMACS, such as infinite forces and atom overlaps, often arise from poorly prepared initial configurations or suboptimal parameter settings. Infinite forces typically result from atom overlaps or unrealistic geometries in the starting structure, which can occur due to improper solvation, ion placement, or packing of solvent molecules in the simulation box (balintUnknownyears systematicexplorationof pages 13-15, lemkul2024introductorytutorialsfors pages 3-4). These issues can also stem from incorrect topology files or inadequate minimization parameters, such as improperly defined nonbonded cutoffs or force field settings (balintUnknownyears systematicexplorationof pages 103-105, lemkul2024introductorytutorialsfors pages 4-6). To mitigate these problems, careful preparation of the system is essential. This includes ensuring proper solvation, avoiding steric clashes, and validating input files such as topology and restraint files (balintUnknownyears systematicexplorationof pages 103-105, lemkul2024introductorytutorialsfors pages 4-6). Applying position restraints during energy minimization can help stabilize the system and prevent unphysical movements, particularly for complex systems or those with structural ions (balintUnknownyears systematicexplorationof pages 103-105, andrio2019bioexcelbuildingblocks pages 5-7). Gradual heating during equilibration and the use of robust algorithms, such as the steepest-descent method and the LINCS algorithm for bond constraints, further reduce the likelihood of infinite forces and overlaps (balintUnknownyears systematicexplorationof pages 13-15, andrio2019bioexcelbuildingblocks pages 5-7). Properly setting physically sound parameters in the <code>.mdp</code> file, such as nonbonded cutoffs consistent with the chosen force field, is critical to achieving a stable energy minimization process (lemkul2024introductorytutorialsfors pages 4-6).