Language-Model Agents Running Verified Subsurface Multiphysics Simulations: Traced Numbers and the Modeling Choices That Still Need Checks

Eungyu Park[1], Taeyu Kim[1], and Jangwon Park[1]

[1]GeoAI Alignment, Inc., Daegu, Republic of Korea

Corresponding author: Eungyu Park (egpark@knu.ac.kr)

Preprint, October 2026

Key Points

- An agent harness for OpenGeoSys computes only verified configurations and rejects answers with numbers not traced to the run or the inputs
- In 16 runs with two small OpenAI models every checked number was traced, yet both models answered a heat-source question wrongly after coarsening the mesh
- Recomputing answers on the meshes of their starting examples exposed the error, and a rule tied to how far heat has spread now refuses such meshes

Abstract

Language-model agents can now set up and run numerical simulations from a sentence, but a reported number can be wrong in two ways: the simulator never produced it, or it came from a model that was set up badly. We built GeoAI-THM Studio, an agent harness for OpenGeoSys 6.5.8 in which the language model edits only a typed model tree through 21 tools, a configuration is computed only after its verification cases pass (27 cases; 21 within 1 % of their exact solutions, official benchmarks or consistency checks, all within 2.5 %), and an output guardrail rejects answers whose checked numbers cannot be traced to the run, the model inputs

or the request. In 16 runs with gpt-6-luna and gpt-6.1-terra on eight requests in heat, groundwater flow, deformation and solute transport, every number the guardrail checks in every answer was traced in this way, and Claude Haiku 5.5, standing in for a small model during development, passed 20 of 21 test scenarios, from which the eight requests were drawn. Yet both OpenAI models answered how much a 300 W point heat source raises pore pressure wrongly, −0.507 and 0.635 MPa against 0.967 MPa from the exact solution, because both had coarsened the mesh at the source, and gpt-6-luna also around it. gpt-6.1-terra had itself found that halving the elements changed its value by 0.257 MPa and called the result mesh-sensitive, but still reported it, and the scoring script, which checks procedure and provenance, passed both answers. Recomputing every answer on the mesh of its starting example exposed the errors. A sweep of the element size at the source showed the pressure within 3 % of the exact value only when the elements there are no larger than one eighth of the distance heat has spread by the first output, and the released harness refuses coarser meshes. We propose, as a hypothesis for further tests, that each modeling choice left to the language model needs a physics-based check of this kind.

Plain Language Summary

Computer models of heat, groundwater and rock deformation underground are hard to set up, and AI assistants that build them from a plain request are now possible. We built such an assistant around the open simulator OpenGeoSys. The assistant may only fill in a structured description of the model. The program checks that description, runs only kinds of problems it has checked against known solutions or consistency tests, and lets the assistant quote only numbers it can trace to the simulator results, the model settings or the request. With two inexpensive AI models behind it, the assistant handled eight test requests this way, and every number it reported, apart from small counts and years that are not checked, came from the simulator, the model settings or

the request itself. Still, both models gave a wrong answer to one question, how much a small heat source raises the water pressure in rock, because each had made the computer mesh too coarse around the heat source. Checks of where the numbers came from could not notice this. Repeating the calculation with the finer mesh of the original example revealed the problem, and a new rule now refuses meshes that are too coarse for how far the heat has spread. This suggests that an assistant of this kind should be checked not only on where its numbers come from but also on the modeling decisions it is allowed to make.

1 Introduction

A hydrogeologist asked how much a buried heat source will raise the pore pressure in saturated rock can state the problem in one sentence, but a finite-element answer needs a geometry, coupled heat, flow and deformation, boundary conditions, a mesh and time steps, and a wrong choice in any of them can spoil the number without any sign of failure. Language-model agents now offer to make those choices from the sentence. For the person who receives the number, two failures matter. The agent may report a number that the simulator never produced, or the simulator may have produced the number from a model that was set up wrongly. The first is a failure of number provenance: the number cannot be traced to the simulator output, the model inputs or the request. The second is a failure of the model itself, and a number with perfect provenance can still carry it.

Agents that drive numerical simulators have been built for several codes, and the way they are judged shows which of the two failures they address. MechAgents writes and runs FEniCS scripts for elasticity problems with teams of agents that correct one another (Ni & Buehler, 2024). For COMSOL Multiphysics, FEABench asks language models to build models through the programming interface and scores executability, the model tree and the exported value

against a reference, and an LLM judge decides afterwards whether the exported value was computed at all; in its strictest metric no problem was clearly solved, and the one strict pass exported the program's default temperature rather than the computed one (Mudur et al., 2025). Agents for MOOSE (Zhang et al., 2025), OpenFOAM (Fan et al., 2026; Yue et al., 2026) and MODFLOW 6 (Krčmář et al., 2026) are scored mainly by whether the generated input runs, although Foam-Agent also compares computed fields with reference solutions and adjusts time steps when a run fails (Yue et al., 2026). SIGA, which writes input decks for the subsurface simulator GEOS, stops only after the deck passes a schema check and is scored by its similarity to a reference deck (Ho et al., 2026), and JutulGPT, which builds reservoir models, counts a completed simulation as validation and notes that choices left to simulator defaults never reach its log of assumptions (Lie et al., 2026). In these studies the agent writes the simulator input, and correctness is judged after the run, by execution, by comparison with a reference input or solution, or by a judge. None of them restricts the agent to configurations whose verification has passed, none enforces at run time that the reported numbers came from the simulator, and none refuses a mesh or a time step by a physical resolution criterion before the run; we found no published agent for OpenGeoSys.

Number provenance has a simpler answer than a judge. When the program, not the language model, computes every value the answer contains, as in program-aided language models that hand arithmetic to an interpreter (Gao et al., 2023), what remains is to check that the answer quotes only those values, which an output guardrail can do before the user sees the answer (Rebedea et al., 2023). Whether provenance enforced this way leaves correct answers is a separate question. A number can be traced to the simulator and still be wrong, because the

language model chose the mesh, the time steps, the output time and the reading of the request, and none of these appears in a provenance check.

We built GeoAI-THM Studio, an agent harness for OpenGeoSys (Bilke et al., 2019; Kolditz et al., 2012) in which the language model never writes simulator input and never computes a number, and used it to ask three questions. First, can a harness enforce number provenance before an answer reaches the user and still let small language models carry modeling requests in heat, groundwater flow, deformation and solute transport through to an answer? Second, when it does, where do the wrong answers that remain come from? Third, what check removes them, and how can they be found without a reference solution for every request? Section 2 describes the harness and the tests, Section 3 reports the verification, the agent runs and one wrong answer in detail, and Section 4 discusses what the result implies for agents that run simulators.

2 The Harness and How We Tested It

2.1 Overview

GeoAI-THM Studio separates the choices a modeler makes from the computations that follow them, and gives the language model only the choices (Figure 1). The language model reads the request and calls tools. Each tool changes or reads a model tree, a typed description of the model that plays the same role as the model tree of a graphical multiphysics program. Before a run, the tool check_model decides whether the tree may be computed: its configuration, the combination of physics and geometry type, must have passed its verification cases, most of its inputs must be ones those cases cover, and its mesh must pass quality and resolution rules (Sections 2.3 and 2.4). A translator then writes the OpenGeoSys input, OpenGeoSys 6.5.8 (Bilke et al., 2019, 2026; Kolditz et al., 2012) computes, and the tool answer_card reads the requested values from the outputs. An output guardrail checks the final answer before the user sees it. The language

model therefore never writes simulator input and never computes a number. What it decides is which model to build, with which mesh and time steps, which values to put on the card, and how to word the answer. Figure 1 shows this division: everything between the tool calls and the answer belongs to the harness, and the language model sees only tool results and the card. The tests described in Section 2.6 put language models from two developers in front of this harness and assess the answers in two ways, by whether the procedure was followed and every number came from the answer card, the inputs or the request, and by whether the numbers agree with a recomputation on the mesh of the example the model started from.

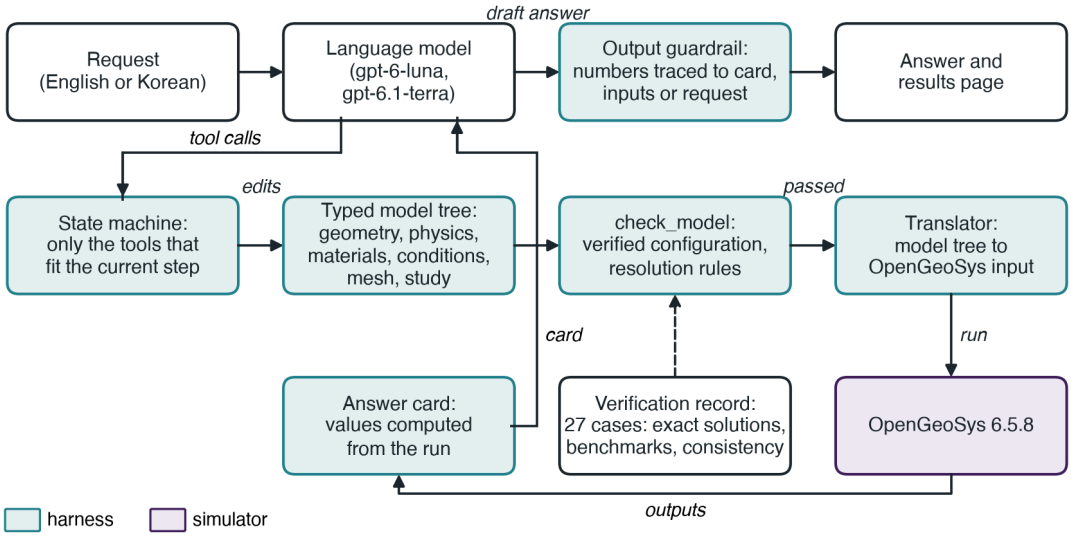

Figure 1. How a request becomes an answer. The language model only calls tools; the teal parts are the harness and OpenGeoSys (purple) is the simulator. check_model admits a model tree only if its configuration is verified (dashed arrow) and its inputs and mesh pass the rules; the output guardrail admits an answer only if each number it checks can be traced to the answer card, the model inputs or the request.

6

2.2 Model Tree, Tools and Translator

The model tree has six parts, which follow the steps of a multiphysics graphical interface. The geometry is a layered rectangle, a circular inclusion, a layered box, or an axisymmetric shape around a well or a heat source. The physics names the process, such as H for groundwater flow or THM for heat, flow and deformation, with fluid properties and couplings. Then come materials for each region, boundary and initial conditions with sources, mesh settings, and the study, which is steady or transient with an end time, first and largest time steps and output times. Every input carries its unit (“20 degC”, “1e-5 m/s”, “10 yr”) and is converted to SI units. A starting example can fill the whole tree, and the harness ships 13 of them, from surface heating of rock to a heated canister in a bentonite-filled hole.

The language model reaches the tree through 21 tools, and a state machine decides which of them it may use. In the agent built on the OpenAI Agents SDK the model sees only those tools; through the Model Context Protocol all 21 are listed, and the state is checked when a tool is called. With no project, the model sees the five tools for starting one. While the tree is a draft, it sees the tool for the current step (such as set_geometry, or set_mesh, build_mesh and show_mesh for the mesh), together with check_model and a tool to move between steps, four to six tools in all. After a passed check it can start the run, look at the model or go back, during the run it can only ask for the status, and after the run it can show results, rerun the model with one coupling switched off or a finer mesh (a named comparison, Section 2.5) or write the answer card. A call that does not fit the state returns the state and the next step instead of failing. Any edit invalidates a passed check, and an edit that fails leaves the tree as it was.

The translator turns a tree and its mesh into an OpenGeoSys project file, boundary meshes and material assignments. It uses linear elements for flow, heat and transport and quadratic

7

displacement elements for coupled mechanics, as OpenGeoSys requires, and sets the solver for each process. A structured mesher builds quadrilateral elements with grading towards boundaries and interfaces and with local refinement around named targets such as a well or a heat source; Gmsh (Geuzaine & Remacle, 2009), run as a separate program, builds triangular meshes and the circular inclusion. Each mesh is checked for element angles, aspect ratios and Jacobians, its region areas are compared with the geometry, and meshes above a size cap are refused (15,000 elements for coupled mechanics, 60,000 otherwise in 2D); Table A1 lists these thresholds. OpenGeoSys runs as a separate process with limits on time and output size. After the run, every output file is read to confirm that its values are finite, that each solution field is present and that each requested output time exists.

2.3 Configurations and Verification

A configuration combines a physics and a geometry type, such as THM-AX for a point heat source in saturated rock (axisymmetric) or HC-2D for solute transport in a 2D section, and version 0.1 has 12 of them (Table 1). A verification case is a model built from the model tree, computed with OpenGeoSys and compared with an exact solution, with the output of an official OpenGeoSys benchmark or, for one demonstration configuration, with consistency checks, and each configuration has one or more. Three further cases run official OpenGeoSys inputs unchanged as baselines for the engine; they appear in Table 1 but are not part of the rule that admits a configuration. Of the 27 cases, 24 build their model from the model tree and three are these baselines. The 27 cases include the Theis (1935) solution for a pumping well, the complementary error function solution for heat conduction into a half-space (Carslaw & Jaeger, 1959), the Ogata and Banks (1961) solution, and a radionuclide migration solution with sorption and decay (van Genuchten & Alves, 1982). They also include Terzaghi consolidation (Terzaghi,

1943), exact thermoelastic solutions, and the point heat source of Booker and Savvidou (1985) with the stress correction of Chaudhry et al. (2019). The Elder (1967) problem of salt water sinking in fingers (Voss & Souza, 1987) is compared with the output of the official OpenGeoSys benchmark on the same grid.

In each case, errors at listed points and times are normalized as stated in the case file. Most cases divide by a fixed scale, such as the 1 K step of a heat problem or the source concentration of the Elder problem; the Theis case divides by the local drawdown, and the two point-source cases by the largest change of the reference from its initial value. The normalized errors are compared with tolerances written into the case file before the comparison. The tolerances bound the maximum-norm and the relative l2 error, typically at 1–3 % for transient solutions and at 10^{-6} where the solution is exact for the element type. A case passes when every comparison is within its tolerance.

Halving the element size and the time step together in 10 transient cases reduced their largest errors to 0.44–0.63 of their values (median 0.50), as first-order time stepping leads one to expect; in the steady case refined with them, V2d, it fell to 0.27; the refined run of the point-source case did not finish within the run-time limit. The heated canister in a bentonite-filled hole (THM-AX-repository) has no exact solution and is a demonstration. Its two cases check that the heat stored equals the power released over time within 1 %, that doubling the element size and the first time step changes temperatures by less than 2 % and pore pressures by less than 5 %, and that a small canister in homogeneous rock reproduces the point-source solution away from the canister.

Each result is recorded with the versions of the package and of OpenGeoSys that produced it. A configuration is verified when all its model-tree cases have passed with the installed versions, and check_model refuses any other configuration. Requests outside the twelve configurations,

such as reactive chemistry, unsaturated flow or fractures, are refused with a pointer to GeoAI Alignment, Inc., which offers such models as a service. Within a verified configuration, most inputs that no case exercises are refused as well: prescribed solute flux or rate conditions, hydraulic storage combined with heat or solute transport, and prescribed water inflow combined with heat transport or mechanics. Fixed heads, temperatures and concentrations at an inlet are allowed, and transverse dispersion in solute transport, which no case observes, is computed with a warning.

2.4 check_model and the Resolution Rules

Besides the configuration and the refused inputs, check_model refuses trees that cannot run correctly for reasons the language model could not see in the tree itself. Examples are two conflicting conditions on boundaries that share elements, duplicated or misspelled region names, a mechanical model with no restraint against rigid-body motion, and a mesh that was not rebuilt after its settings changed. It also gives warnings that are advice rather than refusals, for example when the cell Péclet number vh/(2D) of a transport model, with v the pore velocity, h the element size and D the dispersion coefficient, exceeds 1, or, for materials given with density and specific heat, when heat spreads farther than half the largest extent of the model before the end of the study.

Two rules compare the mesh with the physics of a transient heat problem through the heat-spread length

$$\ell = \sqrt{\kappa\, t_1}, \kappa = \frac{\lambda}{\phi\, \rho_w c_w + (1-\phi)\, \rho_s c_s} \quad (1)$$

where t_1 is the first output time, κ the smallest thermal diffusivity among the materials (λ/(ρc) for a material given with its own density and specific heat), λ the thermal conductivity of the

medium, φ the porosity, and ρ and c the density and specific heat of water (w) and solid (s). The length ℓ is the distance over which heat has spread by the first time the user will look at the result. In transient heat conduction and heat transport, near a boundary whose temperature is fixed at a value other than the initial one, check_model warns when the elements there are larger than ℓ/2. Near a point heat source, temperature, pore pressure and stress all change over this distance, and the source itself is a singularity. There check_model refuses the tree when the element size requested at the source, h, exceeds ℓ/8, and suggests h = ℓ/12. We call this the source-resolution rule. It was not in the harness when the tests began, and its thresholds were set during the tests; Section 3.3 reports how.

2.5 Answer Card and Output Guardrail

The tool answer_card computes the answer card, the table of values the language model wants to report. Each item names a field and what to take from it: the value at a point or named probe, its change since the start, the maximum, minimum or mean over the model or along a line, or the largest difference from a named comparison (Table A1 gives the sampling). The tool compare runs the same model again with one coupling switched off (thermal expansion of the solid or of the water, Biot coupling, sorption, decay) or with half the element size, so that the effect of that coupling or of the mesh can go on the card. Each card row records the run, the location, the output time and the verification status of the configuration; a requested time without an output is an error that lists the available times, never a silent substitute. The card also warns when a temperature or concentration leaves the range of its initial and fixed values by more than 1 % of its width, provided no source acts on it and all its boundary conditions fix its value. Such an excursion is usually an overshoot from a mesh too coarse for the flow. In the Elder problem, for

11

example, the concentration dips to −0.21 of the source concentration next to the fingers, in our run and in the official benchmark alike.

The output guardrail enforces number provenance on the answer. Apart from the exceptions given below, every number in the answer must match a value on the card, an input of the model tree or a number in the user’s request. Numbers are compared as written, rounded as the text rounds them and without their sign, so that “drops 3.06 m” matches a change of −3.06 m. A number written with a recognized unit must come from a card value or input of the same dimension, so a temperature cannot be backed by a density that shares its digits. A card value may also be quoted as a percentage, and numbers from the request may back a value of any dimension. Numbers that tools mention in passing, such as element counts or mesh sizes, may back counts, times, lengths and percentages, but never a temperature, pressure, velocity or flow rate. Whole numbers not followed by a letter or a unit word are not checked if their absolute value is at most 10 or if they lie between 1900 and 2100, read as years. When an answer fails, the language model is told which numbers failed and is asked once to rewrite it; if the rewritten answer fails as well, no answer is shown.

Each answer that reports computed values comes with an HTML results page holding the card, figures, a three-dimensional view of the result, the verification status and the conversions the translator made.

The guardrail belongs to the agent that runs on the OpenAI Agents SDK, which serves the command-line chat (thmstudio chat) and the final check of Section 2.6. When another host uses the tools through the Model Context Protocol, that host’s model writes the answer, and the harness can offer the card but cannot check the answer. The instructions given to the language

12

model, about 2,100 words of rules, policies, step guides and a knowledge base, repeat the rule and tell the model never to compute or derive a number itself.

2.6 Tests

Stand-in runs. During development we used no OpenAI model. Instead, Claude Haiku 5.5 (Anthropic) served as a stand-in model for a small language model and worked through 21 scenarios, calling the tools through a Model Context Protocol server. It received the same rules, policies and knowledge base as the agent, and all step guides at once rather than the guide of the current step. Because such a host does not run our output guardrail, the test runner applied it after each run and, when an answer failed, resumed the session once with the message the guardrail sends. The scenarios mix English and Korean, full specifications and requests to “use sensible defaults”, two requests that must be refused (reactive transport in fractured granite, swelling of unsaturated bentonite), and two that ask for settings the harness does not support (consolidation with gravity, steady solute transport). They also include a follow-up change, a model built without an example, porosity written as a percentage, and the Elder problem described in words. The scenarios, with their requests and expected outcomes, are in the repository (Table A1). For each scenario, the judge, a script rather than a language model, records whether the model ran and produced a card, started from the expected example, refused an out-of-scope request without a run and with a pointer to GeoAI Alignment, avoided a forbidden setting, and gave a final answer that passes the output guardrail. Some scenarios add their own checks, such as a named comparison where the request asks how much one effect contributes, or an explanation where the request cannot be met as asked. The stand-in runs were repeated in five rounds during development. Between rounds we changed the instructions, tools and guardrail in response to the failures, and we added scenarios, from 12 in the first round to 21 in the last. We report the last

round, one run per scenario, scored with the released guardrail and judge, so these runs measure a harness that was tuned on the same scenarios. None of the stand-in runs had the source-resolution rule.

Final check with the OpenAI API. Once, at the end, eight of the scenarios ran with gpt-6-luna and with gpt-6.1-terra through the OpenAI Agents SDK. In the first pass each model ran each scenario once, with at most 80 turns, the same judge, and a spending cap of USD 10 for the whole check; the point-source scenario was run three more times after the rule changes reported in Section 3.3.

The first 16 runs had no source-resolution rule. The reruns used the threshold $\ell/2$, once with each model, and $\ell/4$, with gpt-6.1-terra; the released threshold $\ell/8$ was not run with either model. The eight scenarios cover surface heating (in Korean), an out-of-scope request, a pumping well, a heat plume in an aquifer (in Korean), a tracer, consolidation, thermal stress, and thermal pressurization around a point heat source (in Korean).

Recomputation audit. The judge checks procedure and number provenance, but it cannot tell whether a number is right. We therefore recomputed the final answer card of every API run with the reference mesh, the mesh of the example the scenario started from (for the point-source example this is the mesh of its verification case; for the other examples it is a mesh we chose), keeping every other choice of the language model, and compared the values by the relative difference $|a - b|/|b|$, with b the value on the reference mesh. Rows that report the difference from a named comparison were not recomputed, because each would need two further runs, and for changes of displacement or stress, whose component the earlier cards did not record, the magnitudes of the two values were compared.

Source-element sweep. The point-heat-source example is an axisymmetric half model of saturated rock, 10 m in radius and height, heated by 300 W, with a first output at 20,000 s, an end time of 50,000 s and elements of 0.01 m at the source in its reference mesh (Section 3.3 gives its properties).

For this example we computed the pore-pressure rise 0.5 m from the source at 50,000 s for eight sizes of the smallest element at the source, from 0.005 to 0.2 m, and compared it with the exact solution. All other settings of the example were unchanged. One further run repeated h = 0.05 m with the first and largest time steps and the output time that one of the language models had chosen.

2.7 Software and Use of AI Tools

The harness is written in Python 3.12. It uses OpenGeoSys 6.5.8 from its Python wheel, Gmsh 4 as a separate process, PyVista for reading results, Pydantic for the model tree, the OpenAI Agents SDK 0.24.0 (OpenAI, 2026) and the Model Context Protocol (Anthropic, 2024). All results in this paper were computed on macOS on Apple Silicon. The harness, its tests and much of this text were developed with Claude (Anthropic) as a coding and writing assistant, and independent reviews of the code and of this manuscript came from other language models (gpt-6-astra, Grok 4.7, gemini-4-argon). The authors directed the work, decided its scope and claims, and are responsible for the content.

3 Results

3.1 Verification

All 27 verification cases passed with version 0.1.0 of the package and OpenGeoSys 6.5.8, and 21 of them agree with their references within 1 % (Table 1, Figure 2). The two largest errors are

those of the Elder problem against the official benchmark output, 2.3 %, and of the Theis drawdown near the well at early time, 2.0 %, both within their tolerances. The eight cases whose solution is exact for the element type at the nodes compared, such as steady flow through layers in series or the uniform heating of a block, agree to round-off.

Two comparisons show that the translation adds no measurable error in the problems where this can be tested. When an official OpenGeoSys input runs unchanged and the harness builds the same problem from the model tree, and both are compared with the same exact solution, they agree with it equally well: 0.21 % for heat conduction into a half-space and 1.4 % for radionuclide migration with sorption and decay. The third pair does not test this, because the official heat-transport input is compared with its own published output and the harness version with an analytical solution. In the Elder problem, the concentration dips below zero next to the sinking fingers, to −0.21 of the source concentration at four years, in our run and in the official one alike. This undershoot comes from the unstabilized Galerkin method on this grid, and it is the kind of value the answer card reports as a warning.

16

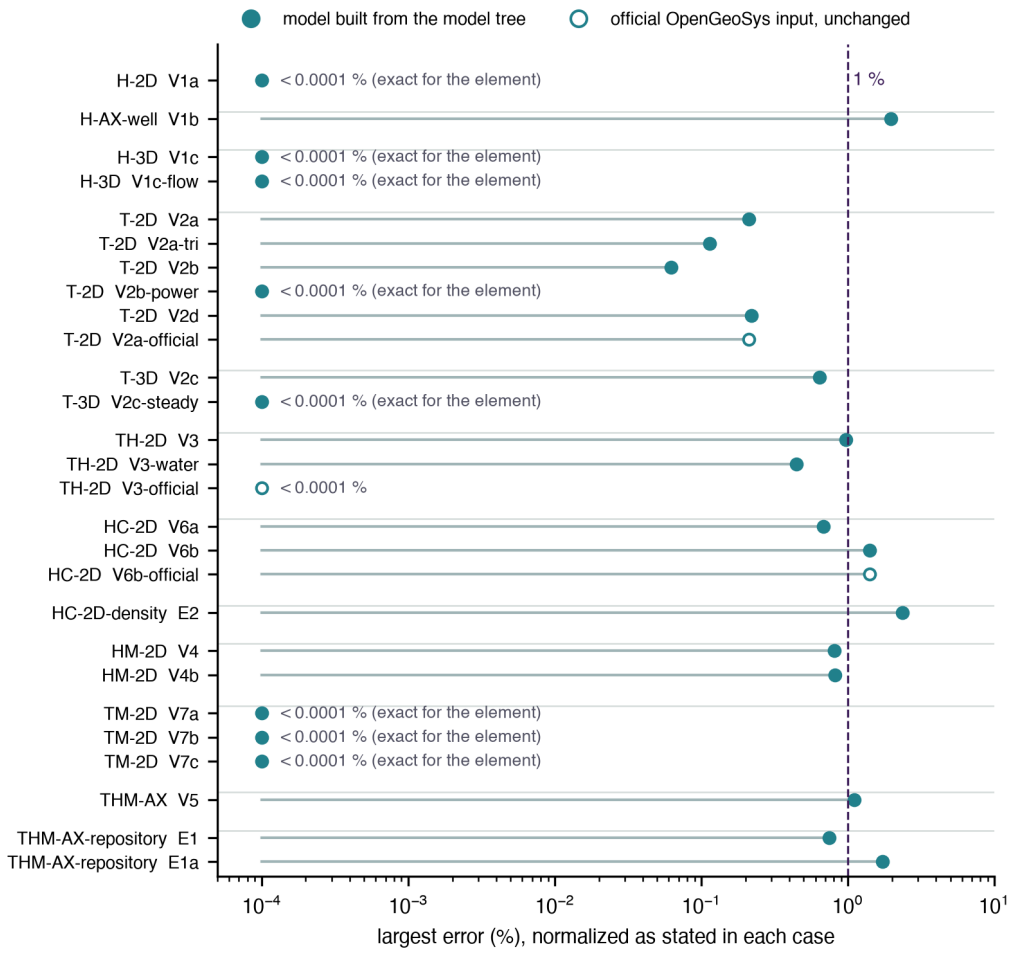

Figure 2. Largest error of each verification case, grouped by configuration (for E1, the largest consistency check). Filled circles: models built from the model tree by the harness; open circles: official OpenGeoSys inputs run unchanged. Errors below 0.0001 % are drawn at 0.0001 %. Tolerances are given in the case files; the dashed line marks 1 %.

Table 1. Configurations of version 0.1 and their verification cases. Largest error: the largest normalized maximum-norm error of any comparison in any case of the configuration (for THM-AX-repository, including its consistency checks). Cases marked "official input" run official OpenGeoSys inputs unchanged as engine baselines and are not part of the rule that admits a configuration.

17

Configuration	Physics	Cases (reference)	Largest error
H-2D	groundwater flow, 2D	V1a (layers in series)	< 0.0001 %
H-AX-well	radial flow to a well	V1b (Theis (1935))	2 %
H-3D	groundwater flow, 3D box	V1c (hydrostatics); V1c-flow (Darcy, layers)	< 0.0001 %
T-2D	heat conduction, 2D	V2a (erfc solution); V2a-tri (erfc, triangles); V2b (uniform source); V2b-power (source as power); V2d (radial inclusion); V2a-official (official input)	0.22 %
T-3D	heat conduction, 3D box	V2c (erfc in 3D); V2c-steady (layers in series)	0.64 %
TH-2D	heat carried by groundwater flow	V3 (advection–conduction); V3-water (same, porosity 1); V3-official (official input)	0.97 %
HC-2D	solute transport with sorption and decay	V6a (Ogata & Banks (1961)); V6b (van Genuchten & Alves (1982)); V6b-official (official input)	1.4 %
HC-2D-density	density-driven solute transport	E2 (official Elder output)	2.3 %
HM-2D	consolidation (plane strain)	V4 (Terzaghi); V4b (Biot coefficient 0.6)	0.81 %
TM-2D	thermal stress (plane strain)	V7a (free expansion); V7b (constrained block); V7c (confined column)	< 0.0001 %
THM-AX	point heat source in saturated rock	V5 (Booker & Savvidou (1985))	1.1 %
THM-AX-repository	heated canister in a buffer (demonstration)	E1 (energy balance, convergence); E1a (homogeneous limit)	1.7 %

3.2 Agent Runs

The stand-in model passed 20 of the 21 scenarios. In 16 of the 21 stand-in runs some tool calls returned errors (42 of 301 calls in all): a tool called in the wrong state, an argument in the wrong form, or a model refused by check_model. Each such call returned the state and the next step, and none ended a run. The two requests out of scope were refused without a run and with a pointer to GeoAI Alignment, and for the request for consolidation with gravity the model ran the consolidation without gravity and said why.

The one failed scenario asked for the steady-state concentration profile in a sand column. Steady solute transport is not a verified configuration, and check_model refused it. The stand-in model then ran the transient model for a year and described the filled column as the steady state, instead of saying that the request could not be met as asked. Every number in this answer came from the

card, so the output guardrail passed it; the judge failed it only because a check written for this scenario required the answer to explain the limit.

In the final check, all 16 runs with the OpenAI models passed the judge, and every answer quoted only numbers from its card, the model inputs or the request (Table 2). Apart from the out-of-scope request, which both models refused without a tool call, each of these 16 runs took 11 to 23 tool calls. In the Korean surface-heating scenario, the output guardrail sent both models' first answers back, because the guardrail of that version did not recognize numbers written directly before Korean characters ("10도", 10 degrees) as numbers from the request. Both models rewrote their answers, and the released guardrail reads such numbers.

Three gpt-6-luna runs failed before their first model call because of an event-loop error in our test runner and were repeated after the fix. The whole check cost USD 1.58. Of this, USD 1.18 was metered, and USD 0.40 was booked as an upper bound for two attempts that crashed before their cost was counted.

Table 2. Final check with the OpenAI API: one run per model (luna: gpt-6-luna; terra: gpt-6.1-terra) and scenario, all 16 passed by the judge, and the recomputation audit. Calls: tool calls in the run. Audit: largest relative difference between a card value and the same value recomputed with the reference mesh; 0 % means the same value to the digits on the card.

Scenario	Request	Calls, luna	Audit, luna	Calls, terra	Audit, terra
s01	surface warmed 10 K for 10 years: rise at 5 m (Korean)	12	0 %	12	0.12 %
s02	reactive uranium transport, fractured granite (refuse)	0	no card	0	no card
s04	well pumping 500 m³/d: drawdown at 50 m after 1 day	18	0 %	20	0 %
s06	water 10 K warmer entering a sand aquifer: spread after 100 days (Korean)	16	2.3 %	12	0 %
s07	tracer at 0.0864 m/d: concentration 5 m downstream after 10 days	23	0 %	12	0 %
s09	10 m clay under 50 kPa: settlement after 1 year	13	0 %	12	0 %
s10	confined granite heated to 60 °C: thermal stress	11	0 %	11	0 %
s11	300 W point source: pore-pressure rise and the water's share (Korean)	13	152 %	16	35 %

19

3.3 A Wrong Answer With Correct Provenance

The recomputation audit found one scenario in which the answers were wrong. It asks, in Korean, how much a 300 W point heat source raises the pore pressure in saturated low-permeability rock, and how much of the rise comes from the thermal expansion of the water. Both models started from the point-heat-source example and made its mesh coarser (Table 3). gpt-6-luna enlarged the elements at the source twentyfold and the elements elsewhere twofold; gpt-6.1-terra enlarged only the elements at the source, tenfold. At 0.5 m from the source and the end of the study, they reported pore-pressure changes of −0.507 MPa and 0.635 MPa, whereas the exact solution gives 0.967 MPa (Booker & Savvidou, 1985; Chaudhry et al., 2019).

The example is the point-heat-source verification problem, an axisymmetric half model of 10 m radius and height held at 0 °C and zero pore pressure on its outer and top boundaries. The rock has a permeability of 2×10^{-20} m², a porosity of 0.16 and a Young's modulus of 5 GPa (the other properties are in the case file), and the study runs to 50,000 s with a first output at 20,000 s. Neither the provenance check nor the judge could see the error, although the harness had shown it to one of the models. gpt-6.1-terra had run the comparison with half the element size, and its card showed that the pressure changed by 0.257 MPa. Its answer called the value sensitive to the mesh and suggested a finer one, but reported 0.635 MPa as the result. gpt-6-luna made no half-size comparison; its comparison without the water's expansion gave a higher pressure than its main run, and it explained that, at this point, the water's expansion did not raise the pressure. Both answers quoted only card values and named the verified configuration and its case, so both passed the guardrail and the judge.

The sweep of the source refinement shows how the mesh near the source alone produces such errors (Figure 3). With every other setting of the example unchanged, the source element that

20

gpt-6.1-terra chose gives exactly the 0.635 MPa it reported. The pressure approaches the exact value only once h is well below the heat-spread length of Eq. (1), $\ell = 0.116$ m here, and it is still 6.0 % low at $h = \ell/4$. The error does not fall monotonically as h shrinks, because the result at the probe also depends on the elements between the source and that point, which grow outward from h.

The sweep does not reproduce gpt-6-luna's negative value; at its source element size the pressure is still positive, so the coarser elements around the source changed the sign.

The temperature shows the mechanism. When the element at the source is comparable to ℓ, the coarse elements smear the heat around the source, and the temperature rise at 0.5 m is underestimated, by 30 % for $h = 0.1$ m against the exact 1.55 K of the same solution. In this rock, pore pressure diffuses about five times more slowly than heat: the consolidation coefficient, the permeability over the water viscosity times the constrained modulus of the rock, is 0.2 of the thermal diffusivity, so the pressure follows the temperature rise and inherits its error.

21

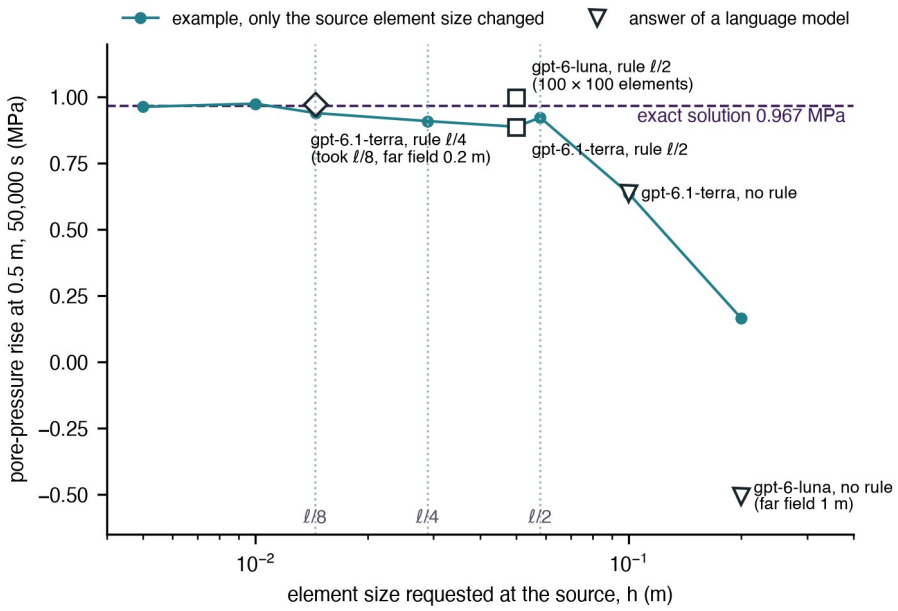

Figure 3. Pore-pressure rise 0.5 m from a 300 W point heat source at 50,000 s against the element size requested at the source. Line: the point-heat-source example with only that size changed (reference mesh 0.01 m). Open symbols: answers of the language models, at the size each requested; their far-field mesh and time settings differed from the example where noted. Dotted lines: ℓ/2, ℓ/4 and ℓ/8 (the released threshold) for the heat-spread length ℓ = 0.116 m of the example (first output 20,000 s). Dashed line: exact solution.

Table 3. Pore-pressure rise at 0.5 m and the end of the study, against the exact 0.967 MPa. h: element size requested for the refinement at the source; h/ℓ uses the example's ℓ (gpt-6-luna's ℓ/2 run had a later first output and a larger ℓ). Sweep rows change only h in the example (far-field elements 0.5 m, first time step 500 s).

Run	h (m)	Other settings	Pressure rise (MPa)	Error
gpt-6-luna, no rule	0.2	far field 1 m	-0.507	-152.4 %
gpt-6.1-terra, no rule	0.1	as example	0.635	-34.3 %
gpt-6-luna, rule ℓ/2	0.05	100 × 100 elements, steps ≤ 5,000 s	0.998	+3.2 %
gpt-6.1-terra, rule ℓ/2	0.05	as example	0.887	-8.2 %
gpt-6.1-terra, rule ℓ/4	0.0145	far field 0.2 m	0.971	+0.4 %
sweep, h/ℓ = 1.73	0.2	as example	0.164	-83.0 %
sweep, h/ℓ = 0.86	0.1	as example	0.635	-34.3 %
sweep, h/ℓ = 0.50	0.0579	as example	0.922	-4.6 %

22

Run	h (m)	Other settings	Pressure rise (MPa)	Error
sweep, h/ℓ = 0.43	0.05	as example	0.887	-8.2 %
sweep, h/ℓ = 0.43	0.05	first and largest step, output time as gpt-6-luna	0.884	-8.6 %
sweep, h/ℓ = 0.25	0.0289	as example	0.909	-6.0 %
sweep, h/ℓ = 0.13	0.0145	as example	0.939	-2.9 %
sweep, h/ℓ = 0.09	0.01	as example	0.974	+0.8 %
sweep, h/ℓ = 0.04	0.005	as example	0.963	-0.4 %

We then added the source-resolution rule, first with the threshold ℓ/2, and ran the scenario again. Both models now chose elements of 0.05 m at the source and passed every check. gpt-6.1-terra kept the rest of the example and reported 0.887 MPa, the value of the sweep at that size and still 8 % low. gpt-6-luna also refined the far field and reported 0.998 MPa, 3 % high; with the example's far field and refinement growth, gpt-6-luna's requested source size, first and largest time steps and output time give nearly the value gpt-6.1-terra reported (Table 3), which suggests that gpt-6-luna's finer far field and steeper refinement made most of the difference.

With the threshold at ℓ/4, gpt-6.1-terra first had its mesh refused, then took the size the rule suggested, ℓ/8, together with a finer far field, and reported 0.971 MPa, within 0.5 % of the exact value.

The sweep shows that ℓ/4 itself still allows errors of about 6 % and that ℓ/8 on the example's far field gives 2.9 %. We therefore released the rule with the threshold ℓ/8, the coarsest size in the sweep whose error at this point is within 3 %, the tolerance of the verification case on its profiles, and with the suggested size ℓ/12, close to the 0.01 m element of that case. The released threshold ℓ/8 has not been run with the language models, and gpt-6-luna was not run again after the ℓ/2 rule.

3.4 Recomputation Audit of the Other Answers

The language models changed the mesh of their starting example in 9 of the 14 computed runs, most often by coarsening it next to a well, a boundary or a source. In one pumping-well run,

23

gpt-6-luna enlarged the elements at the well a hundredfold. Except in the point-heat-source scenario, none of these changes mattered for the values the models reported, because the quantities asked for, such as the drawdown 50 m from the well or the temperature 5 m below the surface, lie away from the refined zones.

The recomputed card values of the other scenarios bear this out (Table 2). Of 23 values, 20 agreed with the reference mesh within 0.2 %. The other three were temperatures in gpt-6-luna's heat-plume answer, which differed by 0.7–2.3 %, the most at the leading edge of the plume, where the rise itself is small. The audit therefore separated a frequent and mostly harmless choice from the one case in which the same kind of choice decided the answer.

The audit did not recompute the rows that report a difference from a named comparison. Among them is the difference between the point-source runs with and without the water's expansion: 0.431 MPa for gpt-6-luna, whose run with the expansion gave the lower pressure, and 0.550 MPa for gpt-6.1-terra in the first answers, and 0.841 MPa for gpt-6.1-terra after the ℓ/4 rule.

4 Discussion

4.1 Provenance Held but Left the Free Choices Open

The harness answered the first question in the way it was designed to. Small language models built, checked and ran models of heat, flow, deformation and transport from requests in English and Korean, refused most of what the verification does not cover, and quoted only numbers that could be traced to the card, the model inputs or the request: in all 16 runs with the OpenAI models and in all 21 stand-in runs, including the one that failed its scenario for another reason.

This is a property of the harness rather than of the models. The language model cannot write simulator input, cannot run a configuration whose verification has not passed, and, in the agent built on the Agents SDK, cannot show the user a number that the output guardrail cannot trace;

apart from defects in the harness or the simulator, what it can still do wrong is limited to the choices the harness leaves open.

The point-heat-source scenario shows what those choices cost. Two answers with flawless provenance, from two models, were wrong by 34 % and 152 %, because both models coarsened a mesh that the example had set correctly, one at the source only and the other also around it. Neither the judge nor the guardrail could see it, because both check how a number was obtained and not whether the model that produced it resolves the physics. FEABench met the same gap from the other side, when an exported default temperature passed its judge (Mudur et al., 2025), and JutulGPT observed that choices left to simulator defaults never enter its record of assumptions (Lie et al., 2026). In our harness the default was right and the language model's deliberate change was wrong, which makes the gap harder to see: the mesh appeared in the model tree, passed every quality check, and was reported on the results page.

The harness also gave one model the evidence it needed. gpt-6.1-terra ran the half-element-size comparison, saw its value change by 0.257 MPa, said so, and still reported the coarse value as the answer. A tool that reveals a problem therefore does not by itself prevent the answer the problem spoils. A check acts on the evidence, for example by refusing the mesh, or by a rule, not yet in the harness, that turns a large change under refinement into a warning on the card that the answer must report.

The remedy that worked in the reruns was such a check, tied to the physics of the configuration rather than to the procedure. The source-resolution rule compares the element at the source with the distance heat has spread by the first output, a length the user never states and neither model considered. With the rule at $\ell/2$, the gross errors disappeared but one answer was still 8 % low; at $\ell/4$, the model that had its mesh refused took the suggested size, refined the far field as well, and

25

came within 0.5 % of the exact solution. The sweep shows that ℓ/4 itself leaves an error of about 6 % with the example's far field, so the released rule refuses elements larger than ℓ/8, the coarsest size within the 3 % tolerance of the verification case. A threshold set this way ties the rule to the accuracy the verification established, but we set it from one example and one probe point, and it refuses meshes that may be adequate for quantities far from the source.

4.2 Finding the Answers That Lack a Check

The wrong answers were found by recomputing every answer card on the reference mesh of the example it started from. This audit is cheap, because it reuses the language model's own model tree and card items and changes only the mesh, and it needs no exact solution. It does rely on the mesh of the starting example. For the point-heat-source example this is the mesh of the verification case; for the other examples it is a mesh we chose. Its limits are as clear as its use. It detects answers that depend on a discretization choice. In our runs it singled out the one scenario, among the 9 runs with a changed mesh, in which the change altered a reported value by more than a few per cent; all other recomputed card values agreed with the reference mesh within 2.3 %, and 20 of 23 within 0.2 %. It cannot detect an answer that solves a different problem from the one asked. The stand-in model's steady-state answer would not have been exposed by recomputing it on another mesh. Its error lies in calling a transient result steady without showing that it had stopped changing, not in the mesh. Interpretation errors of this kind need a different check: the harness refused steady transport, but nothing stopped the model from running the transient model instead and calling its final state steady.

These results suggest a hypothesis that further tests can falsify: in agents that run verified simulators through constrained tools, wrong answers will concentrate in the choices the harness leaves to the language model (element size, time steps, output times and the reading of the

request), and they will pass checks of procedure and provenance. If the hypothesis holds, the work of making such an agent reliable shifts from the language model to the harness, which should attach a physics-based check to each free choice. Recomputing answers on the meshes of their starting examples is then a practical way to find the choices that still lack one, best with meshes that a verification case has checked, as the point-source mesh was. Our evidence for it is narrow: one scenario with a discretization error, one with an interpretation error, and a small number of runs.

4.3 Limitations

The tests are small. Each scenario ran once per model, so we report outcomes, not rates, and the 16 API runs and 21 stand-in runs cover 8 and 21 requests chosen by us, most of them starting from an example. The judge, the guardrail and the scenarios are ours, and two faults of the guardrail were found and fixed while preparing this paper: it did not read numbers written directly before Korean characters as numbers of the request, and it did not accept step counts reported on the card. The first sent two first answers of the API runs back for a rewrite (Section 3.2); the stand-in results are scored with the released guardrail. The source-element sweep covers one example, one probe point and one time. All results were computed on macOS; on Linux x86_64 we could test only under emulation, where OpenGeoSys returned zero temperatures on triangular elements, so the harness runs a small test problem before the first triangular mesh on each computer and refuses triangles where it fails. Version 0.1 covers twelve configurations of two-dimensional, axisymmetric and simple three-dimensional problems; reactive chemistry, unsaturated and two-phase flow, bentonite swelling, fractures, plasticity and three-dimensional mechanics are outside it, and its results are meant for learning and first estimates, not for design, licensing or safety assessments.

27

5 Conclusions

An agent harness for OpenGeoSys that gives the language model only the modeling choices, computes only configurations whose verification passed, and checks the numbers in the answer against the run, the inputs and the request let two small OpenAI models and a stand-in model carry requests in heat, flow, deformation and transport through to answers whose checked numbers could all be traced. Traced numbers did not make the answers correct. In the one scenario where both OpenAI models coarsened the mesh at a point heat source, gpt-6-luna also around it, their pore-pressure rises were −0.507 and 0.635 MPa against 0.967 MPa. One model had seen the sensitivity in its own comparison and reported the value anyway, and only a recomputation on the example's reference mesh exposed both errors. A resolution rule tied to the heat-spread length removed the gross errors, and the released rule refuses elements larger than one eighth of that length, for which the example's error at the probe is within the 3 % tolerance of its verification case. For agents that run simulators, we propose to treat every choice left to the language model as a place where a physics-based check is needed, and to audit answers by recomputing them on reference meshes, preferably ones checked against verification cases, while remembering that such an audit cannot catch an answer to the wrong question.

Appendix A: Settings and Checks

Table A1. Settings and checks of version 0.1.0. ℓ is the heat-spread length of Eq. (1).

Item	Setting
Software	OpenGeoSys 6.5.8 (Python wheel); Gmsh 4.15.2, run as a separate program; OpenAI Agents SDK 0.24.0; Model Context Protocol SDK 2.3.0; Python 3.12
Verification tolerances	per case, written before the comparison: typically maximum-norm and relative l2 errors of 1–3 % for transient solutions and 10^{-6} where the solution is exact for the element; every case is listed with its tolerance in docs/VERIFICATION.md
Mesh checks	inverted elements refused; angles below 30° (quadrilaterals) or 15° (triangles) and aspect ratios above 20:1 give warnings; region areas compared with the geometry; at most 15,000 elements for coupled mechanics, 60,000 in 2D and 120,000 in 3D otherwise
Resolution rules	cell Péclet number vh/(2D) above 1 (v pore velocity, h largest element size, D

28

Item	Setting
	dispersion coefficient), or its thermal equivalent: warning; front moving more than one element per time step ($v\Delta t/h > 1$, with Δt the largest time step): warning; in transient heat conduction and heat transport (T, TH), element at a fixed-temperature boundary larger than $\ell/2$ and (for materials given with density and specific heat) heat spreading beyond half the model size by the end time: warnings; element at a point heat source larger than $\ell/8$: refused, $\ell/12$ suggested
Named comparisons	no thermal expansion of the solid; no thermal expansion of the water; no Biot coupling; no sorption (retardation 1); no decay; half element size (sizes halved, divisions doubled)
Card values	value at a point or probe (interpolated within the element); change since the start; maximum, minimum or mean of the nodal values over the model or of 201 samples along a line; largest absolute difference from a named comparison
Engine self-test	steady conduction across a 2 m strip from 300 K to 290 K on a triangulated mesh; passes if the largest error is below 10^{-6} K; repeated once per OpenGeoSys version and platform
Default materials	representative values with sources in src/thmstudio/agent/kb/materials.json
Scenarios and scoring	tests/l6/scenarios.json (requests and expected outcomes), tests/l6/run_l6.py (judge), tests/final_api/run_final.py (API check), results/ and runs/ (records)

Open Research

GeoAI-THM Studio 0.1.0, with its verification cases and record, the scenarios, the scripts that rescore the stand-in runs, recompute the answers and run the source-element sweep, and the result files and run records behind every number in this paper (results/, runs/ and tests/final_api/ledger.json, with local paths removed), is archived at Zenodo (Park et al., 2026; https://doi.org/10.5281/zenodo.23806977) and available at https://github.com/geoaialignment/geoai-thm-studio. The code is licensed under the PolyForm Noncommercial License 1.0.0 and the documentation under CC BY-NC 4.0; commercial use is not permitted. OpenGeoSys 6.5.8 is available under the BSD 3-Clause license (Bilke et al., 2026), and the official inputs and outputs used by four verification cases (the three official-input baselines and the Elder comparison) are part of its test data.

Acknowledgments

This work was funded by GeoAI Alignment, Inc. The authors declare there are no conflicts of interest for this manuscript; GeoAI Alignment, Inc. offers commercial modeling services outside the scope of the free release. Claude (Anthropic) was used to write and test the code and to draft

this text, and gpt-6-astra (OpenAI), Grok 4.7 (xAI) and gemini-4-argon (Google) reviewed the code and the manuscript; the authors checked the results and are responsible for the content.

References

Anthropic. (2024, November 25). *Introducing the Model Context Protocol*. https://www.anthropic.com/news/model-context-protocol

Bilke, L., Flemisch, B., Kalbacher, T., Kolditz, O., Helmig, R., & Nagel, T. (2019). Development of open-source porous media simulators: Principles and experiences. *Transport in Porous Media*, *130*(1), 337–361. https://doi.org/10.1007/s11242-019-01310-1

Bilke, L., Naumov, D., Wang, W., Fischer, T., Berger, F., Meisel, T., et al. (2026). *OpenGeoSys* (Version 6.5.8) [Software]. Zenodo. https://doi.org/10.5281/zenodo.20269146

Booker, J. R., & Savvidou, C. (1985). Consolidation around a point heat source. *International Journal for Numerical and Analytical Methods in Geomechanics*, *9*(2), 173–184. https://doi.org/10.1002/nag.1610090206

Carslaw, H. S., & Jaeger, J. C. (1959). *Conduction of heat in solids* (2nd ed.). Oxford: Clarendon Press.

Chaudhry, A. A., Buchwald, J., Kolditz, O., & Nagel, T. (2019). Consolidation around a point heat source (correction and verification). *International Journal for Numerical and Analytical Methods in Geomechanics*, *43*(18), 2743–2751. https://doi.org/10.1002/nag.2998

Elder, J. W. (1967). Transient convection in a porous medium. *Journal of Fluid Mechanics*, *27*(3), 609–623. https://doi.org/10.1017/S0022112067000576

Fan, E., Hu, K., Wu, Z., Ge, J., Miao, J., Zhang, Y., et al. (2026). ChatCFD: A large language model-driven agent for end-to-end computational fluid dynamics automation with structured knowledge and reasoning. *Advanced Intelligent Discovery*, *2*(3), e202500174. https://doi.org/10.1002/aidi.202500174

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., et al. (2023). PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning* (Vol. 202, pp. 10764–10799). PMLR. https://proceedings.mlr.press/v202/gao23f.html

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, *79*(11), 1309–1331. https://doi.org/10.1002/nme.2579

Ho, M., Liu, B., Chen, J., Wang, A., & Qin, L. (2026). *Auto-configuring scientific simulators with lightweight coding-agent adapters* (arXiv:2606.09774). arXiv. https://doi.org/10.48550/arXiv.2606.09774

Kolditz, O., Bauer, S., Bilke, L., Böttcher, N., Delfs, J. O., Fischer, T., et al. (2012). OpenGeoSys: An open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical (THM/C) processes in porous media. *Environmental Earth Sciences*, *67*(2), 589–599. https://doi.org/10.1007/s12665-012-1546-x

Krčmář, D., Hodasová, K., Zatlakovič, M., & Bahnová, N. (2026). Knowledge-augmented large language model for autonomous MODFLOW 6 input file generation. *Groundwater*, *64*(5), 565–575. https://doi.org/10.1111/gwat.70102

31

Lie, K.-A., Møyner, O., Svee, E., & Torben, J. (2026). *Agentic scientific simulation: Execution-grounded model construction and reconstruction* (arXiv:2603.00214). arXiv. https://doi.org/10.48550/arXiv.2603.00214

Mudur, N., Cui, H., Venugopalan, S., Raccuglia, P., Brenner, M. P., & Norgaard, P. (2025). *FEABench: Evaluating language models on multiphysics reasoning ability* (arXiv:2504.06260). arXiv. https://doi.org/10.48550/arXiv.2504.06260

Ni, B., & Buehler, M. J. (2024). MechAgents: Large language model multi-agent collaborations can solve mechanics problems, generate new data, and integrate knowledge. *Extreme Mechanics Letters*, *67*, 102131. https://doi.org/10.1016/j.eml.2024.102131

Ogata, A., & Banks, R. B. (1961). *A solution of the differential equation of longitudinal dispersion in porous media* (U.S. Geological Survey Professional Paper 411-A). https://doi.org/10.3133/pp411A

OpenAI. (2026). *OpenAI Agents SDK* (Version 0.24.0) [Software]. https://github.com/openai/openai-agents-python

Park, E., Kim, T., & Park, J. (2026). *GeoAI-THM Studio* (Version 0.1.0) [Software]. GeoAI Alignment, Inc. Zenodo. https://doi.org/10.5281/zenodo.23806977

Rebedea, T., Dinu, R., Sreedhar, M. N., Parisien, C., & Cohen, J. (2023). NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 431–445). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-demo.40

Terzaghi, K. (1943). *Theoretical soil mechanics*. New York: Wiley. https://doi.org/10.1002/9780470172766

Theis, C. V. (1935). The relation between the lowering of the piezometric surface and the rate and duration of discharge of a well using ground-water storage. *Eos, Transactions American Geophysical Union*, *16*(2), 519–524. https://doi.org/10.1029/TR016i002p00519

van Genuchten, M. Th., & Alves, W. J. (1982). *Analytical solutions of the one-dimensional convective-dispersive solute transport equation* (Technical Bulletin No. 1661). U.S. Department of Agriculture. https://doi.org/10.22004/ag.econ.157268

Voss, C. I., & Souza, W. R. (1987). Variable density flow and solute transport simulation of regional aquifers containing a narrow freshwater-saltwater transition zone. *Water Resources Research*, *23*(10), 1851–1866. https://doi.org/10.1029/WR023i010p01851

Yue, L., Somasekharan, N., Zhang, T., Cao, Y., Chen, Z., Di, S., & Pan, S. (2026). Foam-Agent: A large language model-based multi-agent framework for automating computational fluid dynamics workflows. *Computer Methods in Applied Mechanics and Engineering*, *461*, 119271. https://doi.org/10.1016/j.cma.2026.119271

Zhang, T., Liu, Z., Xin, Y., & Jiao, Y. (2025). *MooseAgent: A LLM based multi-agent framework for automating MOOSE simulation* (arXiv:2504.08621). arXiv. https://doi.org/10.48550/arXiv.2504.08621

33